Helion

SMASHING MAGAZINE

Thord Daniel Hedengren

# PODRĘCZNIK WordPressa

SMASHING MAGAZINE

Najlepszy podręcznik o WordPressie!

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Tytuł oryginału: Smashing WordPress: Beyond the Blog

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-6681-2

Translation copyright © 2013 by Helion S.A.

This edition first published 2012
© 2012 John Wiley & Sons, Ltd.

All Rights Reserved. Authorised translation from the English language edition published by John Wiley & Sons Limitd. Responsibility for the accuracy of the translation rests solely with Helion S.A. and is not the responsibility of John Wiley & Sons Limited.

Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley and Sons, Inc. and/ or its affiliates in the United States and/or other countries, and may not be used without written permission. WordPress is a registered trademark of Automattic, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Ltd. is not associated with any product or vendor mentioned in the book.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
*ftp://ftp.helion.pl/przyklady/podwsm.zip*

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
*http://helion.pl/user/opinie/podwsm_ebook*
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Poleć książkę na Facebook.com
- Kup w wersji papierowej
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Komu dedykować książkę o WordPressie? Rodzicom? Swojemu zwierzakowi? Kochance? A może swojemu lekko upośledzonemu alter ego, które nosi pończochy i dziwnie interpretuje znaczenie litery W w logo WordPressa?

Nie, żadna z wymienionych postaci przecież nic nie wie o tym systemie. Może jedynie ten dziwny gość od W. W pewnym sensie jest on geniuszem.

Książkę tę dedykuję społeczności skupionej wokół WordPressa, zespołowi rozwijającemu rdzeń systemu oraz wszystkim programistom, którzy piszą motywy i wtyczki oraz udzielają się na różnych blogach i forach. Dzięki tym wszystkim ludziom nasza platforma — tak, nasza — idzie do przodu, a to jest najważniejsze. To, o czym tu mówimy, nie jest kolejnym systemem mającym umożliwić wielkim firmom zaoszczędzenie jakiejś niewielkiej sumy pieniędzy. Nie chodzi też o tworzenie nowych blogów, portali czy projektów tanim kosztem. Nie chodzi nawet o udostępnienie wszystkim najnowocześniejszych technologii sieciowych.

Tu chodzi przede wszystkim o możliwość wyrażenia swoich myśli i umożliwienie tego samego innym. Esencją WordPressa i ogólnie ruchu open source jest wolność wypowiedzi. To właśnie to sprawia, że inwestuję w tę platformę swój czas i nigdy mi się to nie nudzi.

Zatem książka ta jest o możliwościach, jakie WordPress daje każdemu z nas. Używaj tego systemu, wyrażaj swoje opinie i próbuj zmieniać świat. A przy okazji dobrze się baw!

*Thord Daniel Hedengren*

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# Spis treści

**O autorze** **13**

**Wprowadzenie** **15**

**CZĘŚĆ I PODSTAWY WORDPRESSA** **19**

**Rozdział 1. Anatomia instalatora WordPressa** **21**
Podstawowa instalacja 22
Instalacja z kreatorem 22
Instalacja ręczna 23
Korzystanie z zewnętrznego serwera baz danych 26
Inne ustawienia bazy danych 26
Przydatne funkcje pliku wp-config.php 27
Kilka słów na temat instalatorów 28
Przenoszenie instalacji WordPressa do nowego katalogu 29
Modyfikowanie bazy danych 30
Struktura bazy danych WordPressa 31
Usuwanie problemów bezpośrednio w bazie danych 31
Robienie kopii zapasowej 32
Zmienianie hosta 34
Narzędzia eksportu i importu 34
Problemy z importowaniem i eksportowaniem danych 36
Zabezpieczanie WordPressa 38
Użytkownicy i hasła 39
Zabezpieczenia po stronie serwera 39
W następnym rozdziale 40

**Rozdział 2. Składnia WordPressa** **43**
WordPress i PHP 44
Dokumentacja WordPressa 44
Rdzeń WordPressa 44
Motywy i szablony 46
Tagi szablonowe 49
Tagi dołączania plików 49
Przekazywanie kilku parametrów do tagów szablonowych 51
Argumenty w stylu funkcji i łańcuchów zapytań 52
Typy danych 54

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Tagi warunkowe 55
Co w nich takiego wyjątkowego 56
W następnym rozdziale: pętla 57

**Rozdział 3. Pętla 59**
Zasada działania pętli WordPressa 60
Najprostsza pętla 60
Zapisywanie pętli w pliku szablonowym loop.php 61
Kilka słów o WP_Query 62
Używanie pętli 63
Przyklejanie wpisów 69
Formaty wpisów 72
Tag get_template_part() i formaty wpisów 74
Funkcja query_posts() 75
Co zamiast pętli 78
Tworzenie wielu pętli 79
Wyświetlanie proponowanych artykułów 80
To było niezłe, ale cztery pętle to dopiero coś 82
Oswajanie się z pętlą 86

**CZĘŚĆ II PROJEKTOWANIE I PROGRAMOWANIE MOTYWÓW WORDPRESSA 89**

**Rozdział 4. Motywy do WordPressa — wiadomości podstawowe 91**
Podstawy budowy motywu 92
Podstawowe elementy motywu 92
Co będziemy robić 93
Kilka słów na temat języka HTML5 94
Tworzenie plików szablonowych 95
Deklaracja motywu w pliku style.css 95
Plik header.php 96
Plik footer.php 100
Prawa kolumna: plik sidebar.php 102
Treść główna: plik index.php 103
Przenoszenie pętli do osobnego pliku 109
Pojedyncze wpisy i strony 111
Szablony archiwów 114
Błędy 404, wyszukiwarka i zrzuty ekranu 116
Plik functions.php 117
Pliki szablonowe 118
Kiedy używane są poszczególne pliki szablonowe 120
Hierarchia szablonów 121
Szablony stron 121
Korzystanie z pliku functions.php 124
Ustawianie domyślnej szerokości 125
Dodawanie elementów promocyjnych za pomocą pliku functions.php 126



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Widżety — czym są i do czego służą 127
Deklarowanie obszarów na widżety 128
Deklarowanie wielu obszarów na widżety 128
Dostosowywanie widżetów 129
Upiększanie komentarzy 130
Podział komentarzy na wątki 131
Wyróżnianie autora wpisu 133
Dodawanie własnych pól 133
Najczęstsze zastosowanie własnych pól 133
Kwestia użyteczności 134
Tworzenie motywu bazowego 134
Publikowanie motywu 136
Lista punktów do sprawdzenia przed publikacją motywu 136
Motywy komercyjne a licencja GPL 139
Zgłaszanie motywów do WordPress.org 139
W następnym rozdziale 142

**Rozdział 5. Motywy potomne 143**
Genialność motywów potomnych 144
Jak działają motywy potomne 145
Piękno techniki przesłaniania plików szablonowych 147
Wspaniały szablon loop.php 147
Motywy potomne do zastosowań specjalnych 149
Inne spojrzenie na kwestię dziedziczenia 149
Często spotykane problemy 150
Motywy potomne a zarządzanie wieloma witrynami 150
Zarządzanie projektami wielu witryn 151
Nie zapominaj o pliku functions.php 152
Nie należy przesadzać 152
Szkielety motywów 153
Dla odmiany kilka słów o motywach nadrzędnych 153
Wyższy poziom wtajemniczenia 154

**Rozdział 6. Motywy dla zaawansowanych 155**
Planowanie motywu 156
Zasada 1. Stylizuj według kategorii, sortuj według tagów, a dostosowuj według formatów wpisów 157
Zasada 2. Starannie przemyśl własne pola 157
Zasada 3. Używaj stron i własnych typów wpisów 158
Czy to wszystko? 158
Indywidualne techniki stylizacji 158
Stylizowanie wpisów 159
Klasy dla elementu body 161
Przyklejone wpisy 163
Używanie własnych pól 164
Podstawy własnych pól 165
Tworzenie modułów meta 166



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Ciekawe funkcje własne 167
Poprawne dodawanie funkcji w pliku functions.php 168
Ikony wpisów 169
Własne menu 170
Własne nagłówki 170
Własne obrazy tła 171
Haki akcji 171
Używanie haków 172
Tworzenie własnych haków 173
Odłączanie akcji od haków 174
Taksonomie 174
Zastosowania taksonomii 175
Myśl 175
Taksonomie a przenośność 176
Własne typy wpisów 176
Używanie własnych typów wpisów 177
Używanie własnych typów wpisów w motywach 177
Strony opcji motywu 177
Problemy dotyczące opcji motywów 178
Obsługa różnych języków 179
Praca z plikami językowymi 180
Problem z nazwami 181
Kanały RSS 181
Kanały WordPressa 182
Tworzenie własnego kanału RSS 183
Podstawowe kwestie SEO 183
Pozbywanie się niepotrzebnych rzeczy z motywu 185
Motywy a wtyczki 186

**CZĘŚĆ III WTYCZKI DO WORDPRESSA 189**

**Rozdział 7. Anatomia wtyczki do WordPressa 191**
Rodzaje wtyczek 192
Zwykłe wtyczki 192
Wtyczki do rdzenia 192
Wtyczki obowiązkowe 193
Tworzenie wtyczek do użytku w sieciach witryn 193
Wtyczki dla całej sieci 194
Podstawy budowy wtyczek 195
Metody inkorporowania wtyczek 197
Używanie haków 197
Tworzenie własnych tagów szablonowych 198
Funkcje nadpisujące 199
Własne taksonomie i typy wpisów 199
Powody, aby użyć wtyczki 200
Tworzenie własnej taksonomii 200
Tworzenie własnego typu wpisów 202



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Co powinna mieć każda wtyczka 204
Ustawienia wtyczek 204
Baza danych a odinstalowywanie wtyczki 209
Po deinstalacji 210
Wtyczki tworzące widżety 211
Tworzenie widżetu 212
Widżety kokpitu 214
Kwestia korzystania z bazy danych we wtyczkach 216
Zgodność wsteczna wtyczek 217
Publikowanie wtyczek w portalu WordPress.org 217
Słowo ostrzeżenia na temat tworzenia wtyczek 218

**Rozdział 8. Wtyczka czy plik functions.php? 221**
Kiedy tworzyć wtyczki 222
Rozszerzanie funkcjonalności za pomocą wtyczek 222
Ostrzeżenie: wtyczki mogą spowolnić Twoją witrynę 222
Kiedy używać pliku functions.php 223
Dylemat ze skrótami kodowymi 224
Rozwiązanie problemu poprzez użycie motywu potomnego 225
Rozwiązanie uniwersalne: wtyczka na funkcje 225
Tworzenie wtyczki na funkcje 225
Jak ważna jest przenośność 227
Planowanie rozszerzania funkcjonalności WordPressa 228

**CZĘŚĆ IV DODATKOWE FUNKCJE I ROZSZERZANIE FUNKCJONALNOŚCI WORDPRESSA 229**

**Rozdział 9. Używanie WordPressa jako systemu CMS 231**
Czy WordPress jako CMS to dobry wybór 232
Lista punktów do sprawdzenia zanim wybierze się WordPressa 233
Ograniczanie WordPressa do minimum 234
Dostosowywanie panelu administracyjnego 235
Usuwanie funkcji typowych dla bloga 236
Idealna konfiguracja prostej statycznej witryny 237
Bardziej zaawansowane rozwiązania 239
Zastosowanie własnych typów wpisów i taksonomii w WordPressie używanym jako CMS 239
Wykorzystanie widżetów w CMS-ie 240
Obsługa menu 242
Integracja treści spoza WordPressa 242
Nie zapomnij dodać instrukcji obsługi 244
Ostatnie słowo na temat używania WordPressa jako systemu CMS 244

**Rozdział 10. Integracja WordPressa z mediami społecznościowymi 245**
Integracja WordPressa z Facebookiem 246
Przycisk Lubię to 246
Widżety profilowe 248



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Integracja WordPressa z Twitterem 248
Dodawanie przycisków i widżetów Twittera do strony 249
Rozszerzenia Twittera 250
Integracja witryny z Google+ 252
Korzystanie z zewnętrznej obsługi komentarzy 253
Jeden login do wszystkich serwisów 254
Jak ważne są media społecznościowe 256

**Rozdział 11. Sztuczki projektowe 257**
Zwiększanie kontroli nad wpisami 258
Tworzenie projektów opartych na tagach 258
Używanie własnych pól 259
Podpinanie się do funkcji body_class(), post_class() oraz comment_class() 261
Dodawanie własnych taksonomii 261
Ulepszanie menu 262
Przesuwane drzwi 263
Menu rozwijane 265
Wstawianie reklam w pętli 266
Tworzenie pomocnych stron błędu 404 268
Używanie bibliotek JavaScript w WordPressie 268
Rejestrowanie skryptów 269
Dostosowywanie stylu WordPressa do własnej marki 270
Własny formularz logowania 271
Motywy panelu administracyjnego 272
Dopieszczanie witryny 274

**Rozdział 12. Zabawa z mediami 275**
Tworzenie galerii obrazów 276
Stylizowanie galerii 277
Lepsze przeglądanie w lekkich okienkach 280
Galerie spoza WordPressa 281
Formaty wpisów 282
Osadzanie treści multimedialnej na stronach 283
Konfigurowanie ustawień 283
Magiczna technika oEmbed 284
Wyświetlanie losowych obrazów 284
Wyświetlanie losowych obrazów z galerii 285
Dodatkowe opcje losowania obrazów 286
Optymalne wykorzystanie serwisów do publikowania zdjęć 287
Przechowywanie obrazów w serwisie Flickr 288
Używanie pokazów slajdów z serwisu Flickr 290
Strzeż się bałaganu 292

**Rozdział 13. Dodatkowe funkcje 293**
Wyświetlanie treści na kartach 294
Inteligentne zastosowanie 294
Używać kart czy nie 297



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Wyświetlanie zawartości kanałów RSS 297
Wbudowany parser 298
Buforowanie przy użyciu API Transients 299
Mieszanie kanałów za pomocą SimplePie 300
Własne skróty kodowe 301
Dodawanie skrótów kodowych 301
Ciekawostki dotyczące skrótów kodowych 302
Wysyłanie e-maili z WordPressa 303
Dodawanie formularza logowania 304
Drukowanie treści 306
Więcej… 308

**Rozdział 14. Nietypowe zastosowania WordPressa 309**
Publikowanie treści dostarczanej przez użytkowników 310
Przyjmowanie wpisów od użytkowników 311
Obsługa wiadomości i recenzji publikowanych przez użytkowników 312
Tworzenie tablicy ogłoszeń o pracę 313
Funkcja wp_editor() 315
Ostatnie słowo na temat treści dodawanej przez użytkowników 316
WordPress jako baza wiedzy 316
Dodawanie funkcji 317
Dodatkowe ulepszenia 318
WordPress i handel elektroniczny 319
Prowadzenie sklepu opartego na WordPressie 320
Sprzedawanie produktów cyfrowych 320
Budowa sklepu 321
Tworzenie katalogu produktów 321
Tworzenie typu wpisów dla książek 322
Tworzenie strony dla książek 323
Promowanie produktów 326
Blog ze smakiem, czyli witryna z przepisami 328
Przystawka, czyli wybór motywu 329
Danie główne — przepisy jako typ wpisów 329
Deser — własne taksonomie 330
Ziółko na trawienie — podsumowanie 334
Tworzenie witryny z odnośnikami 334
Alternatywne rozwiązanie: odnośnikowy format wpisów 336
Kilka uwag na temat zastosowań 337
Mieszanie wpisów odnośnikowych ze zwykłą treścią 337
Inne zastosowania WordPressa 338
Strona z wydarzeniami i kalendarz 338
Intranet i współpraca 339
Społeczności i fora 339
Bazy danych 340
Statyczne witryny 340
Dzienniki i notatki 340
Możesz mieć wszystko, czego chcesz 341



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

**CZĘŚĆ V DODATKI 343**

**Dodatek A Niezbędne wtyczki do WordPressa 345**
Wtyczki związane z treścią 346
Wtyczki multimedialne 347
Wtyczki administracyjne 348
Wtyczki do zarządzania komentarzami i eliminowania spamu 353
Wtyczki mediów społecznościowych 354
Wtyczki subskrypcji i do obsługi urządzeń przenośnych 355
Wtyczki dotyczące SEO i wyszukiwania 356
Kod źródłowy i dane wyjściowe 358
Przestroga na zakończenie: czy na pewno potrzebujesz tej wtyczki 360

**Dodatek B Motywy bazowe 361**
Jak wybrać motyw 362
Znaczenie słowa szkielet 362
Propozycje motywów 363
Motywy Twenty Ten i Twenty Eleven 363
Starkers 364
Roots 365
Toolbox 366
Constellation 367
Spectacular 368
Bones 369
Twój motyw, Twoje zasady 369

**Skorowidz 371**



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# O autorze

**Thord Daniel Hedengren** jest uzależniony od pisania, przez co w 1996 roku założył swój pierwszy newsletter. Od tamtej pory cały czas miał z górki, założył kilkanaście stron internetowych oraz pracował jako redaktor i wolny strzelec w różnych miejscach w Szwecji i poza jej granicami.

Międzynarodową karierę rozpoczął od napisania wpisu na blogu, co w konsekwencji doprowadziło do zawarcia umowy z wydawnictwem Wiley na napisanie książki *Smashing WordPress: Beyond the Blog* i jej kontynuacji *Smashing WordPress Themes: Making WordPress Beautiful*. Przy okazji umocnił swoją pozycję w społeczności WordPressa. W ręku trzymasz trzecie wydanie dobrze przyjętej przez krytykę książki *Podręcznik WordPressa. Smashing Magazine*. Na tym jednak na pewno się nie skończy, bo Hedengren cały czas sypie pomysłami jak z rękawa.



Kiedy na chwilę zapomni o swojej obsesji pisarskiej, wraz z przyjaciółmi z firmy Odd Alice tworzy fajne strony internetowe oparte na WordPressie. Ponadto redaguje czasopisma i pisze artykuły zarówno do druku, jak i publikacji internetowych po szwedzku i angielsku. Możesz śledzić jego poczynania na stronie *http://tdh.me*.

Hedengren mieszka w kraju królów, Szwecji.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# Wprowadzenie

Napisanie książki o WordPressie nie jest łatwe. Podczas pierwszej rozmowy z redaktorem doszliśmy do wniosku, że książka ta powinna być nie tylko wprowadzeniem dla programistów do WordPressa, lecz tekstem, który pobudzi ich wyobraźnię i zachęci do szukania niekonwencjonalnych rozwiązań.

Innymi słowy, chcieliśmy, aby czytelnik wyszedł *poza ramy zwykłego bloga*. Idea ta jest zawarta w podtytule od pierwszego wydania i nadal jest aktualna.

Ogólnie chodzi przede wszystkim o udowodnienie, że WordPress jest nie tylko platformą blogową. Przy użyciu tego systemu można stworzyć w zasadzie dowolny portal internetowy i to w dodatku szybko i solidnie. WordPress nie w każdej sytuacji jest najlepszym rozwiązaniem, ale zawsze należy rozważyć możliwość jego użycia. Jedną z jego największych zalet jest łatwość użytkowania i tworzenia serwisów internetowych. Równie ważne są wsparcie bardzo aktywnej społeczności, która pomoże Ci, gdy będziesz w tarapatach, oraz nieograniczony dostęp do kodu źródłowego.



Mimo iż praktycznie każdy posiadający podstawową wiedzę o językach HTML, CSS i PHP może nauczyć się korzystać z WordPressa z tej książki, pragnę podkreślić, że nie jest to publikacja dla początkujących. Już od początku narzucam dość wysokie tempo, aby także zawodowcy solidnie zapoznali się z podstawami. Bardzo łatwo jest przywiązać się do pewnych rozwiązań tylko dlatego, że zawsze się tak robiło, przez co nie próbuje się dowiedzieć, jak to coś powinno się robić prawidłowo. Opis podstawowych zagadnień pomoże wprawdzie także początkującemu czytelnikowi, ale żeby skorzystać z tej książki, naprawdę musisz przynajmniej trochę znać języki HTML i CSS oraz mieć jakieś pojęcie o języku PHP. Jeśli te zagadnienia są Ci obce, koniecznie najpierw poczytaj o nich.

Podsumowując, książka *Podręcznik WordPressa. Smashing Magazine* została napisana w taki sposób, aby każdy, kto ma odpowiednią wiedzę podstawową, mógł nauczyć się z niej obsługi WordPressa i wykorzystać go także do nietypowych celów. Dzięki lekturze tej książki będziesz umiał zrobić wszystko z tym systemem.

Książka jest podzielona na cztery części.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# CZĘŚĆ I: PODSTAWY WORDPRESSA

W części pierwszej opisane są podstawowe zagadnienia dotyczące systemu WordPress — od sposobu instalacji po podstawy działania. Dowiesz się w niej wszystkiego, co trzeba wiedzieć, aby rozpocząć pracę z systemem, chociaż tempo jest tu szybsze niż w typowych książkach dla początkujących. Dodatkowo poznasz kwestie bezpieczeństwa WordPressa, dowiesz się, jak przenieść zainstalowany system na inny serwer i wielu innych rzeczy. Moim celem było nie tylko napisanie poradnika dla początkujących użytkowników, lecz również zwrócenie uwagi bardziej doświadczonych na pewne problemy i dostępne możliwości.

# CZĘŚĆ II: PROJEKTOWANIE I PROGRAMOWANIE MOTYWÓW WORDPRESSA

Motyw to zewnętrzna powłoka systemu. Jest to skóra Twojej strony internetowej, która dodatkowo pełni funkcję kontrolera pozwalającego sterować prezentacją treści. Każdy twórca serwisu internetowego opartego na WordPressie spędza mnóstwo czasu na edycji plików motywu w celu dostosowania go do swoich potrzeb. W tej części poznasz budowę motywów WordPressa i nauczysz się tworzyć własne motywy.



# CZĘŚĆ III: WTYCZKI DO WORDPRESSA

Trzecią część w całości poświęciłem tworzeniu wtyczek do WordPressa. Dzięki możliwości rozszerzania tego systemu za pomocą wtyczek można osiągnąć praktycznie wszystko. Jeśli coś da się zrobić przy użyciu PHP, można to również zrobić w WordPressie. Treść tej części jest w dużym stopniu teoretyczna, ponieważ koncentruje się ogólnie na podstawowych mechanizmach komunikacji wtyczki (która jest w istocie programem w PHP) z systemem.

# CZĘŚĆ IV: DODATKOWE FUNKCJE I ROZSZERZANIE FUNKCJONALNOŚCI WORDPRESSA

W czwartej części książki skupiłem się na używaniu WordPressa do wszystkich innych celów niż blogowanie. Dowiesz się tu, jak wykorzystać ten system jako CMS do prowadzenia typowych portali internetowych oraz przyjrzysz się budowie kilku serwisów od podstaw, aby się przekonać, że rzeczywiście WordPress może być używany nie tylko do tworzenia blogów.

W części tej spróbuję zmusić Cię do spojrzenia na WordPressa z innej perspektywy. Chcę, abyś pozbył się przeświadczenia, że jest to tylko platforma blogowa. W istocie WordPress to platforma do publikowania i nic więcej.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Ponadto w części tej znajdziesz opis różnych ciekawych sztuczek i technik, za pomocą których możesz poszerzyć możliwości swojej witryny. Wiele rzeczy, które będą Ci potrzebne, ktoś już zrobił, i tutaj tylko się o nich dowiesz.

## CZĘŚĆ V: DODATKI

W dodatkach znajduje się wykaz wtyczek i motywów, które mogą znacznie ułatwić Ci pracę. Są to narzędzia, dzięki którym zbudowanie dobrego serwisu internetowego w oparciu o WordPress jest łatwiejsze. Oczywiście wybór motywów i wtyczek do WordPressa jest oszałamiający — i bardzo dobrze — i dlatego zawartość tych dodatków należy traktować jako podręczną ściągawkę, a nie wyczerpującą listę. Mam nadzieję, że jeśli nie znajdziesz tu dokładnie tego, czego potrzebujesz, to przynajmniej będziesz wiedział, od czego zacząć poszukiwanie.

## ZACZNIJ MYŚLEĆ I PUBLIKUJ

Książka *Podręcznik WordPressa. Smashing Magazine* została napisana dla twórców stron internetowych, ale jej treść może być przydatna dla każdego, kto choć trochę zna języki HTML, CSS i PHP oraz WordPressa. Rozpoczęcie pracy z WordPressem jest banalnie łatwe i system ten nadaje się do tworzenia wszelkiego rodzaju serwisów internetowych. Gdy już zbudujesz podstawową witrynę, musisz jednak zakasać rękawy, aby zmodyfikować istniejący motyw lub utworzyć własny od podstaw w celu budowy wymarzonej witryny.

Innymi słowy, zacznij myśleć i publikować z WordPressem niezależnie od tego, czy rozpoczynasz prace nad nowym serwisem typu Engadget lub Wired, czy czymś kompletnie innym.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

CZĘŚĆ I

# PODSTAWY WORDPRESSA

Rozdział 1. Anatomia instalatora WordPressa

Rozdział 2. Składnia WordPressa

Rozdział 3. Pętla

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 1 ANATOMIA INSTALATORA WORDPRESSA

**INSTALACJA WORDPRESSA** jest łatwa i szybka, a na stronach WordPress.org można znaleźć doskonały przewodnik, dzięki któremu przejście przez ten proces nie sprawi problemu nikomu. Jednak dzięki dodatkowej wiedzy, którą zdobędziesz w tym rozdziale, będziesz mógł doładować swoją instalację motywami i wtyczkami. WordPress stanowi fundament Twojej witryny, ale tak naprawdę to wtyczki i motywy decydują o jej działaniu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*Słowo „WordPress" w tej książce odnosi się do samodzielnej wersji systemu dostępnej pod adresem http://wordpress.org. Nie należy jej mylić z wersją Automattic znajdującą się pod adresem http://wordpress.com. Tematem tej książki jest główna wersja WordPressa dostępna pod adresem wordpress.org, a dokładniej mówiąc WordPress 3.4.*

# PODSTAWOWA INSTALACJA

Instalacja WordPressa jest bardzo łatwa. Przechwałki twórców systemu mówiące, że cały proces trwa pięć minut, wcale nie są przesadą. W rzeczywistości większość tego czasu i tak zajmuje wysyłanie plików na serwer, gdy ktoś ma powolne łącze internetowe albo korzysta z usług hostingowych niskiej jakości. Zakładam, że już nieraz przeprowadzałeś podstawową instalację WordPressa, więc nie będę się na ten temat rozwodził.

Najpierw należy sprawdzić, czy nasz serwer spełnia wszystkie wymagania systemowe. Najnowszy opis tych wymagań zawsze można znaleźć pod adresem *http://wordpress.org/about/requirements*. Jeśli na Twoim serwerze zainstalowane są interpreter PHP 5.2.4 lub nowszy oraz baza danych MySQL 5.0 lub nowsza, to możesz zainstalować WordPressa. Dodatkowo serwer powinien mieć zainstalowany moduł `mod_rewrite`, dzięki któremu można włączyć obsługę ładnych adresów.



Są dwie metody instalacji WordPressa: z kreatorem i ręczna.

*Niektóre firmy hostingowe oferują jeszcze trzecią metodę — polegającą na instalacji WordPressa jednym kliknięciem. Dokładniejszy opis tej metody znajduje się w dalszej części tego rozdziału. Ponadto WordPressa można też zainstalować, korzystając z systemu Subversion lub Git, ale w książce nie będzie to omawiane.*

## INSTALACJA Z KREATOREM

WordPress jest jednym z najłatwiejszych w obsłudze systemów zarządzania treścią. Aby rozpocząć korzystanie z niego, wystarczy pobrać plik ZIP ze strony *http://pl.wordpress.org/*, wypakować pliki i wysłać je w wybrane miejsce na serwerze poprzez FTP. Jeśli np. chcesz zainstalować WordPressa pod adresem *mojastrona.pl*, wyślij pliki do katalogu głównego dla *mojastrona.pl*.

Następnie w przeglądarce internetowej wpisz adres do tego katalogu (w naszym przykładzie byłby to adres *http://mojastrona.pl*) i wprowadź wymagane informacje, jak pokazano na rysunku 1.1. Należy podać nazwę bazy danych, nazwę użytkownika, hasło tego użytkownika oraz adres do serwera baz danych, jeśli jest to adres zewnętrzny. Wszystkie te informacje można uzyskać od właściciela hostingu. W niektórych przypadkach bazę danych i użytkownika trzeba utworzyć samodzielnie za pomocą narzędzia cPanel, phpMyAdmin albo SSH. Wszystko zależy od konkretnego hostingu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Rysunek 1.1. Interfejs instalatora



Kliknij przycisk *Wyślij*, aby przejść na stronę instalacji witryny widoczną na rysunku 1.2. W formularzu tym należy wpisać nazwę swojej witryny, dane administratora itd. Wpisz poprawny adres e-mail i zapamiętaj hasło. Po wpisaniu wszystkich danych kliknij przycisk *Zainstaluj WordPressa*. To wszystko. Możesz się zalogować i rozpocząć korzystanie z gotowej witryny internetowej!

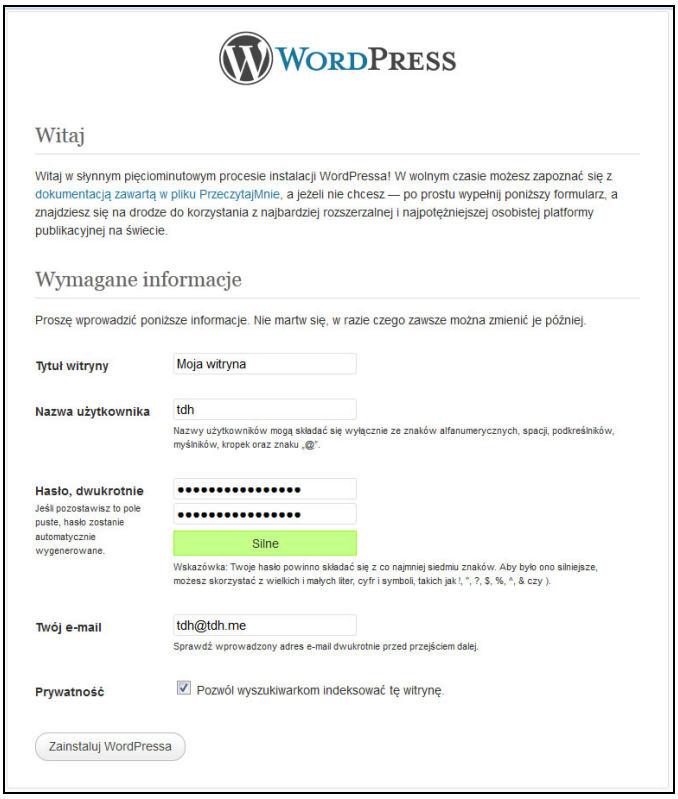
Ta metoda instalacji nie pozwala na wybór innej wersji systemu niż domyślna. Jeśli chcesz mieć WordPressa w innym języku niż polski, wejdź na stronę *http://codex.wordpress.org/WordPress_in_Your_Language* i znajdź tam odpowiednią dla siebie wersję (szczegółowe informacje na ten temat znajdują się w następnym podrozdziale dotyczącym ręcznej metody instalacji).

## INSTALACJA RĘCZNA

Do ręcznej instalacji potrzebne są następujące rzeczy:

- Najnowsza wersja WordPressa (dostępna pod adresem *http://pl.wordpress.org/*).
- Baza danych MySQL z użytkownikiem mającym uprawnienia do zapisu (jeśli nie wiesz, jak ją utworzyć, poproś o pomoc administratora hostingu).
- Program FTP.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 1.2. Formularz z danymi witryny

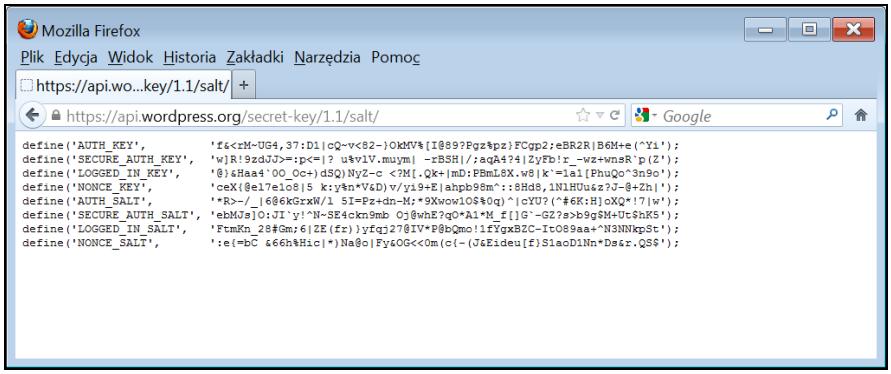
Wypakuj pobrane pliki systemu na dysk swojego komputera i wyślij je na serwer. Następnie otwórz plik *wp-config-sample.php* i przejdź do sekcji dotyczącej konfiguracji bazy danych, w której należy wpisać nazwę bazy oraz jej użytkownika i hasło. Poniżej przedstawiona jest zawartość pliku *wp-config-sample.php*:

```
// ** Ustawienia MySQL-a - możesz uzyskać je od administratora Twojego serwera ** //
/** Nazwa bazy danych, której używać ma WordPress */
define('DB_NAME', 'nazwa_bazy_danych');

/** Nazwa użytkownika bazy danych MySQL */
define('DB_USER', 'nazwa_uzytkownika');

/** Hasło użytkownika bazy danych MySQL */
define('DB_PASSWORD', 'haslo');

/** Nazwa hosta serwera MySQL */
define('DB_HOST', 'localhost');

/** Kodowanie bazy danych używane do tworzenia tabel w bazie danych. */
define('DB_CHARSET', 'utf8');

/** Typ porównań w bazie danych. Nie zmieniaj tego ustawienia, jeśli masz jakieś wątpliwości. */
define('DB_COLLATE', '');
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Następnie w tym samym pliku znajdź sekcję dotyczącą kluczy tajnych. Zaczyna się ona od komentarza `Unikatowe klucze uwierzytelniania i sole`. Pod nią znajdują się pola, w których można wpisać klucze uwierzytelniania. Jest to zabezpieczenie, dzięki któremu instalacja powinna być bardziej bezpieczna i odporna na ataki hakerów. Klucze te wpisuje się tylko raz. Wpisać można dowolny ciąg znaków, ale dostępny jest też automatyczny generator oferowany w serwisie WordPress.org. Wystarczy skopiować z komentarza adres *https://api.wordpress.org/secret-key/1.1/salt/* i wkleić go w pasku adresu przeglądarki. Pojawi się strona przypominająca pokazaną na rysunku 1.3.




Rysunek 1.3. Losowy klucz, który można skopiować i wkleić w swoim pliku



Skopiuj zawartość tej strony i wklej ją do pliku *wp-config-sample.php* w miejsce poniższego kodu:

```
define('AUTH_KEY',         'wprowadz tutaj unikatowa fraze');
define('SECURE_AUTH_KEY',  'wprowadz tutaj unikatowa fraze');
define('LOGGED_IN_KEY',    'wprowadz tutaj unikatowa fraze');
define('NONCE_KEY',        'wprowadz tutaj unikatowa fraze');
define('AUTH_SALT',        'wprowadz tutaj unikatowa fraze');
define('SECURE_AUTH_SALT', 'wprowadz tutaj unikatowa fraze');
define('LOGGED_IN_SALT',   'wprowadz tutaj unikatowa fraze');
define('NONCE_SALT',       'wprowadz tutaj unikatowa fraze');
```

Wstawiając ten kod do pliku, sprawiłeś, że Twoja instalacja będzie nieco bardziej odporna na ataki tych wstrętnych hakerów.

Ostatnim, co możesz chcieć zmienić w pliku *wp-config-sample.php*, jest język. Aby móc zmienić domyślnie zainstalowany język polski na inny, należy w folderze *wp-content/languages/* umieścić odpowiedni plik językowy w formacie *.mo*. Takie pliki dla większości języków można znaleźć pod adresem *http://codex.wordpress.org/WordPress_in_Your_Language*. Dodatkowo w pliku *wp-config-sample.php* musisz wpisać kod swojego języka w poniższym fragmencie kodu:

```
define('WPLANG', '');
```

Kod języka jest taki sam jak nazwa pliku językowego bez rozszerzenia. Jeśli zatem chcesz mieć WordPressa po angielsku, musisz pobrać plik o nazwie *en_EN.mo*, umieścić go na serwerze w katalogu *wp-content/languages/* i zmienić wywołanie funkcji `WPLANG` następująco:

```
define('WPLANG', 'en_EN');
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Nie oznacza to, że wszystkie motywy i wtyczki będą również w wybranym przez Ciebie języku, ale rdzeń WordPressa tak (temat lokalizacji motywów i wtyczek jest opisany w rozdziale 6.).

To wszystko! Zmień nazwę pliku *wp-config-sample.php* na *wp-config.php* i wpisz w przeglądarce adres, pod którym znajdują się pliki systemu. Zostanie wyświetlone łącze pozwalające rozpocząć instalację. Będziesz musiał podać nazwę witryny i adres e-mail administratora oraz zdecydować, czy witryna ma być dostępna do indeksowania dla wyszukiwarek (zazwyczaj należy udostępniać serwis wyszukiwarkom, ale jeśli chcesz najpierw nad nim popracować, to możesz usunąć zaznaczenie tej opcji i włączyć ją później w ustawieniach). Następnie otrzymasz nazwę użytkownika administratora i losowe hasło (zapisz je!) i, miejmy nadzieję, zostaną wyświetlone: wiadomość, że instalacja została zakończona pomyślnie, oraz odnośnik do bloga.

Proste, prawda?

## KORZYSTANIE Z ZEWNĘTRZNEGO SERWERA BAZ DANYCH

Jedną z najczęstszych przyczyn niepowodzenia podczas instalacji WordPressa jest to, że baza danych MySQL znajduje się na osobnym serwerze. Jeśli wyświetlają Ci się błędy połączenia z bazą danych, a masz pewność, że nazwa użytkownika z pełnymi uprawnieniami zapisu i hasło, których używasz, są prawidłowe, to najprawdopodobniej problem tkwi właśnie w tym, że serwer baz danych nie jest lokalny.



Aby rozwiązać ten problem, znajdź w pliku *wp-config.php* (albo *wp-config-sample.php*, jeśli jeszcze nie zmieniłeś nazwy) słowo `localhost` i zastąp je adresem swojego serwera baz danych.

```
define('DB_HOST', 'localhost');
```

Adres serwera MySQL należy uzyskać od administratora hostingu. Może mieć postać w rodzaju *mysql67.superhost.com* albo wyglądać całkiem inaczej. Niezależnie od wyglądu wpisz adres w miejsce `localhost` i ponownie uruchom instalatora. Jeśli trzeba dodatkowo określić numer portu, należy go wpisać po dwukropku.

Oczywiście jeśli nie znasz adresu serwera baz danych, spytaj administratora.

## INNE USTAWIENIA BAZY DANYCH

Jeśli chcesz, możesz przed zainstalowaniem WordPressa zdefiniować jeszcze inne ustawienia bazy danych. Rzadko się z nich korzysta, ale warto wiedzieć o takiej możliwości.

Po pierwsze: można ustawić kodowanie znaków i opcję sortowania (ang. *collation*) bazy danych. Ustawienia te poinformują WordPressa, jaki zestaw znaków jest używany w bazie danych, chociaż w większości przypadków powinien to być UTF-8. Takie też jest domyślne ustawienie w pliku *wp-config-sample.php*, a więc zazwyczaj nie trzeba nic zmieniać, chyba że ma się jakieś specjalne wymagania. Gdybyś zdecydował się to zrobić, poszukaj poniższego wiersza kodu:

```
define('DB_CHARSET', 'utf8');
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jest to domyślne ustawienie zestawu znaków UTF-8 (w kodzie oczywiście zapisane jako `utf8`). Najprawdopodobniej nie musisz i nie powinieneś zmieniać tego ustawienia, chociaż zdarzają się nietypowe sytuacje, w których może to być potrzebne. Zatem zachowaj te informacje na wszelki wypadek.

Sortowanie znaków w zestawie znaków, zwane po angielsku *collation*, jest ustawiane w bazie danych podczas instalacji systemu. Można je zmienić w poniższym wierszu kodu:

```
define('DB_COLLATE', '');
```

Jak widać, domyślnie opcja ta jest pusta, co oznacza, że zostanie zastosowane sortowanie `DB_CHARSET`, czyli przy typowych ustawieniach UTF-8. Jeśli chcesz, możesz to zmienić na coś innego:

```
define('DB_COLLATE', 'wybrany_zestaw_znaków');
```

## PRZYDATNE FUNKCJE PLIKU WP-CONFIG.PHP

Powyżej zostały opisane najczęściej zmieniane ustawienia w pliku *wp-config.php*. Ale to nie znaczy, że są one jedyne. Uwaga dla początkujących: bądźcie ostrożni, bo z plikiem *wp-config.php* nie ma żartów. Jeśli przez nieuwagę wprowadzicie jakieś nieprawidłowe ustawienia, możecie sprawić, że cała wasza strona przestanie działać. Dlatego dobrze jest najpierw wszystkie ustawienia sprawdzić na testowej instalacji WordPressa i wprowadzić we właściwym systemie dopiero wtedy, gdy ma się pewność, że są poprawne.



W pliku *wp-config.php* można zrobić wiele ciekawych rzeczy. Jedną z nich jest zmiana adresu URL witryny i ścieżki do WordPressa, które są zazwyczaj zapisywane w bazie danych (w tabeli `wp_options`) na stronie ustawień systemu. Aby ustawić adres URL witryny w pliku *wp-config.php*, można użyć poniższego kodu:

```
define('WP_SITEURL', 'http://mojadomena.pl/blog/');
```

Analogicznie można też ustawić ścieżkę do instalacji WordPressa. Służy do tego opcja `WP_HOME`:

```
define('WP_HOME', 'http://mojadomena.pl/wp/');
```

Oba te ustawienia nadpiszą ustawienia zdefiniowane w panelu administracyjnym witryny. To samo dotyczy opcji `WP_CONTENT_URL`, która określa położenie folderu *wp-content* i umożliwia przeniesienie go w inne miejsce niż domyślne:

```
define('WP_CONTENT_URL', 'http://mojadomena.pl/files/wp-content');
```

Zwróć uwagę, że na końcu ścieżki do folderu *wp-content* nie ma ukośnika. To samo można zrobić z folderem wtyczek. Ogólnie większość ustawień ścieżek wygląda podobnie. Oczywiście można też dynamicznie określać położenie folderów za pomocą PHP lub ustawień serwera, ale nie będę się nad tym rozwodził.

**Wersje wpisów** to zapisane wersje wpisów i stron. Można wyłączyć ich zapisywanie:

```
define('WP_POST_REVISIONS', false );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jeśli nie chcesz całkiem wyłączać opcji zapisywania wersji wpisów, ale ograniczyć ich liczbę do np. pięciu, wystarczy, że zamiast słowa `false` wpiszesz liczbę:

```
define('WP_POST_REVISIONS', 5 );
```

WordPress automatycznie zapisuje wpisy i strony w panelu administracyjnym. Domyślnie robi to co 60 sekund, ale można zmienić ustawienie na dowolny inny okres, np. co 180 sekund:

```
define('AUTOSAVE_INTERVAL', 180 );
```

Jeśli wystąpią jakieś problemy i chcesz dowiedzieć się, co jest ich przyczyną, możesz użyć opcji `WP_DEBUG`. Narzędzie to powoduje wyświetlenie powiadomień o błędach i ostrzeżeń informujących o użyciu wycofywanych i już nieobsługiwanych funkcji lub plików:

```
define('WP_DEBUG', true);
```

Ustawienie `WP_DEBUG` na `false` jest równoznaczne z nieustawieniem tej opcji w ogóle, tzn. wszelkie błędy nie mające bezpośredniego wpływu na działanie witryny będą ukrywane.

Także opcję tworzenia sieci blogów, tzw. **multisite**, włącza się w pliku *wp-config.php*:

```
define('WP_ALLOW_MULTISITE', true);
```



To ustawienie powoduje pojawienie się w panelu administracyjnym dodatkowych opcji i ustawień. Wrócimy do tego w dalszej części książki, więc na razie nie zaprzątaj sobie tym głowy. Jeśli chcesz poznać więcej sztuczek na temat ustawień dostępnych w pliku *wp-config.php*, zajrzyj na stronę *http://codex.wordpress.org/Editing_wp-config.php*. Znajdziesz na niej także informacje, jak rejestrować błędy w działającej witrynie przy użyciu opcji `WP_DEBUG_DISPLAY` i `WP_DEBUG_LOG`, tak aby nie były wyświetlane zwykłym użytkownikom, lecz zapisywane w dzienniku.

## KILKA SŁÓW NA TEMAT INSTALATORÓW

Niektóre firmy hostingowe oferują instalatory WordPressa umożliwiające instalację systemu za pomocą jednego kliknięcia w panelu administracyjnym konta hostingowego. Jednym z najpopularniejszych tego typu rozwiązań jest Fantastico. W pierwszej chwili może się wydawać, że możliwość tak szybkiej instalacji jest bardzo dobrym pomysłem, ponieważ nie trzeba bawić się w ustawienia w plikach konfiguracyjnych ani żadne inne. Po prostu klikamy i mamy bloga.

Zanim jednak zdecydujesz się na skorzystanie z tej opcji, dobrze się zastanów. Przede wszystkim sprawdź, która wersja WordPressa zostanie zainstalowana. Instalowanie starych wersji powinno być zabronione nie tylko dlatego, że są przestarzałe, ale również dlatego, że stanowią zagrożenie bezpieczeństwa. W każdej nowej wersji systemu likwidowanych jest kilka luk w zabezpieczeniach, a więc aktualizacja do najnowszej wersji to nie tylko zdobywanie najnowszych fajnych funkcji.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Instalatory typu Fantastico są doskonałym rozwiązaniem, pod warunkiem że instalują najnowszą wersję WordPressa. Jeśli dowiesz się, że instalator oferowany w Twoim hostingu tak robi, spróbuj też zrobić rozeznanie u innych użytkowników, czy nie mieli z nim jakichś poważnych problemów. Jeśli nie znajdziesz przeciwwskazań i naprawdę nie chce Ci się korzystać ze słynnego 5-minutowego instalatora ręcznego, użyj automatu.

Po zainstalowaniu WordPressa za pomocą instalatora natychmiast dokonaj aktualizacji systemu do najnowszej wersji, korzystając z jego wbudowanej funkcji. Jeśli nie możesz skorzystać z tej funkcji, dokonaj aktualizacji, wysyłając nowe pliki przez FTP. Sprawdź, czy instalator z jakichś dziwnych przyczyn nie uniemożliwia dokonywania aktualizacji — chyba nie chcesz być związany na stałe z plikami instalatora.

## PRZENOSZENIE INSTALACJI WORDPRESSA DO NOWEGO KATALOGU

Czasami lepiej jest umieścić cały system w jakimś folderze. Dzięki temu pozbywamy się bałaganu w głównym katalogu domeny na koncie hostingowym, a poza tym łatwiej możemy zarządzać kilkoma witrynami na tym samym koncie. Przypuśćmy, że chcemy zainstalować jakieś dodatkowe aplikacje sieciowe. Jeśli pomieszamy wszystkie pliki różnych programów w jednym katalogu, to później trudno będzie się w tym połapać (chociaż wszystkie pliki WordPressa w katalogu głównym jego instalacji mają nazwy zaczynające się od przedrostka *wp-*). Problemy zaczynają się, gdy obok WordPressa chcemy używać jeszcze innych aplikacji.

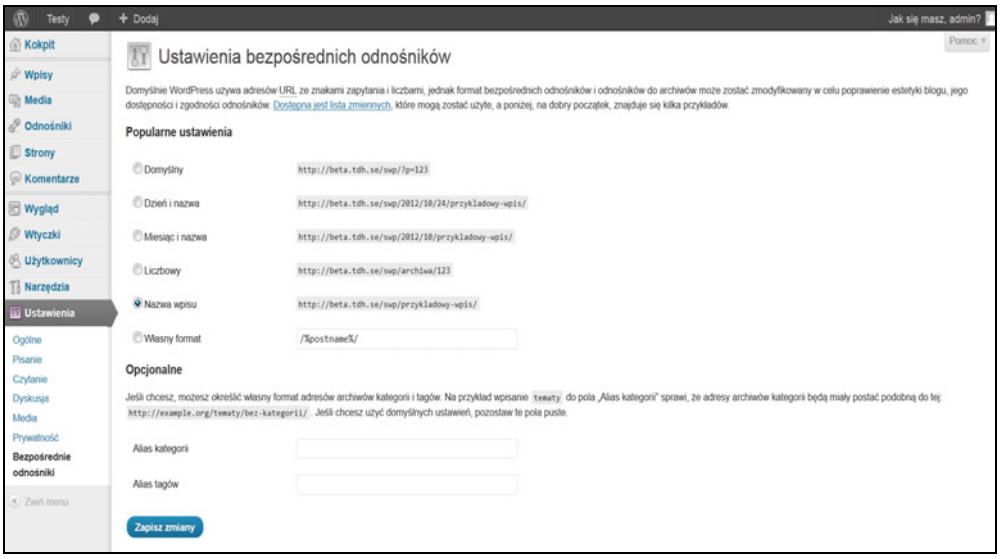


Instalacja WordPressa w podfolderze przebiega tak samo jak instalacja w katalogu głównym domeny, a więc nie będę się na ten temat rozpisywał. Ogólnie rzecz biorąc, chodzi o to, aby pliki WordPressa znajdowały się w podfolderze, natomiast sam blog działał tak, jakby znajdował się w katalogu głównym domeny. Do wyboru mamy dwa rozwiązania: możemy od razu zainstalować WordPressa w podfolderze albo zainstalować go w katalogu głównym i potem przenieść pliki do podfolderu. Niezależnie od tego, na które z tych rozwiązań się zdecydujesz, oba są bardzo łatwe.

Przyjmuję założenie, że masz już WordPressa zainstalowanego w katalogu głównym i chcesz go przenieść do podfolderu. Załóżmy, że obecnie pliki WordPressa znajdują się w katalogu głównym domeny domena.pl i chcemy je przenieść do podkatalogu *wpsystem*, pozostawiając przy tym samą witrynę w katalogu głównym. Innymi słowy, gdy zwykły użytkownik wejdzie pod adres *http://domena.pl*, zobaczy naszą witrynę. My natomiast do panelu administracyjnego będziemy się logować w folderze *wpsystem*, korzystając z adresu *domena.pl/wpsystem/wp-admin/*.

Przed rozpoczęciem przenosin należy ustawić bezpośrednie odnośniki, ponieważ chcemy, aby łącza działały niezależnie od tego, czy używamy podfolderu. Opcje bezpośrednich odnośników znajdują się w menu *Ustawienia/Bezpośrednie odnośniki* (rysunek 1.4).

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 1.4. Strona ustawień bezpośrednich odnośników jest jedną z kilku stron z ustawieniami WordPressa



Przenoszenie instalacji WordPressa do nowego katalogu zaczniemy od utworzenia folderu o nazwie *wpsystem*. Następnie przechodzimy na stronę *Ustawienia/Ogólne* i w polu *Adres URL WordPressa* wpisujemy *http://domena.pl/wpsystem*, aby wskazać adres do nowego folderu, a w polu *Adres URL witryny* wpisujemy *http://domena.pl*, czyli adres do strony głównej naszej witryny. Kliknij przycisk *Zapisz zmiany* i przenieś do katalogu *http://domena.pl/wpsystem* wszystkie pliki WordPressa oprócz plików *index.php* i *.htaccess*, które powinny pozostać w katalogu głównym domeny.

Po przeniesieniu plików otwórz plik *index.php* i znajdź w nim poniższy fragment:

```
require('./wp-blog-header.php');
```

Zmień go następująco:

```
require('./wpsystem/wp-blog-header.php');
```

Teraz plik *wp-blog-header.php* jest dołączany z folderu *wpsystem*.

Zaloguj się w panelu administracyjnym WordPressa (teraz pod adresem *domena.pl/wpsystem/wp-admin/*) i zaktualizuj bezpośrednie odnośniki. To wszystko.

# MODYFIKOWANIE BAZY DANYCH

Zazwyczaj bazą danych nie trzeba się w ogóle przejmować, ponieważ WordPress automatycznie robi w niej wszystko, co trzeba. Czasami zmienia się w niej coś, gdy instalowana jest aktualizacja systemu, ale wszystko to odbywa się automatycznie bez naszej ingerencji. Dlatego najczęściej jedyną czynnością wykonywaną na bazie danych jest robienie jej kopii zapasowej.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Niemniej jednak czasami, gdy coś pójdzie źle, może być konieczne dokonanie pewnych zmian bezpośrednio w bazie, aby to naprawić. Najczęściej w bazie grzebie się z powodu utraty haseł, wystąpienia dziwnych adresów spowodowanego przenosinami witryny, zmiany domeny lub problemów z wtyczkami.

---

*Ostrzeżenie: pamiętaj, że bezpośrednie modyfikowanie zawartości bazy danych to poważna ingerencja w system. Nie da się tu niczego cofnąć, a więc jeśli coś usuniesz, to już tego nie odzyskasz. Nawet jeśli dobrze wiesz, co robisz, zawsze przed rozpoczęciem takich prac zrób kopię zapasową bazy danych. Jeśli nie znasz się na bazach MySQL i nie potrafisz posługiwać się narzędziem phpMyAdmin, lepiej tego nie ruszaj, bo na pewno coś popsujesz.*

---

## STRUKTURA BAZY DANYCH WORDPRESSA

Baza danych systemu WordPress ma nieskomplikowaną strukturę. Zawiera 11 tabel, w których przechowywana jest treść. Odpowiedzi na większość pytań można znaleźć, po prostu przeglądając tę bazę, a gdy wiadomo, czego szukać, można od razu dokonać poprawek. Oczywiście w dokumentacji (*http://codex.wordpress.org/Database_Description*) można też znaleźć wyczerpujący opis struktury bazy danych i warto tam zaglądać, gdy chce się coś znaleźć.

Oto lista 11 głównych tabel bazy danych WordPressa:

- `wp_commentmeta`: metadane komentarzy;
- `wp_comments`: treść komentarzy;
- `wp_links`: dodane łącza i ich dane;
- `wp_options`: opcje witryny;
- `wp_postmeta`: metadane wpisów;
- `wp_posts`: treść wpisów;
- `wp_terms`: kategorie i tagi;
- `wp_term_relationships`: tworzy relacje kategorii i tagów z wpisami;
- `wp_term_taxonomy`: opisy kategorii i tagów;
- `wp_usermeta`: metadane użytkowników;
- `wp_users`: użytkownicy.

Wszystkie te tabele są bardzo ważne, ale modyfikacje najczęściej wprowadza się w tabelach `wp_options` (gdy trzeba naprawić problemy związane z adresami URL), `wp_posts` (gdy trzeba dokonać masowych zmian we wpisach) i `wp_users` (gdy trzeba zresetować hasła itp.).

## USUWANIE PROBLEMÓW BEZPOŚREDNIO W BAZIE DANYCH

Przez większość czasu WordPress działa prawidłowo i nie trzeba ingerować w jego bazę danych. Gdy jednak zdarzy Ci się, że po wejściu na swoją stronę ujrzysz straszny biały ekran, możesz podejrzewać, że przyczyną problemu jest któryś z widżetów i możliwym rozwiązaniem jest skasowanie widżetów w bazie danych. Dane widżetów są przechowywane w tabeli



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

`wp_options`. Sposób postępowania i nazwy dostępnych widżetów zależą od tego, jakie mamy zainstalowane wtyczki, a więc należy być ostrożnym. Dane zazwyczaj występują pod nazwą odzwierciedlającą w jakiś logiczny sposób używane wtyczki, a więc nie powinno być problemu ze znalezieniem tego, czego się szuka. Może się to wydawać dość ryzykowne, ale warto spróbować, gdy po dokonaniu aktualizacji WordPressa zamiast swojej strony zobaczymy biały ekran. Jeśli będziesz potrzebował pomocy, warto poszukać jej na forum na stronach WordPress.org.

Kolejnym problemem, który może wymagać ingerencji w bazę danych, jest zmienienie lub zresetowanie hasła użytkownika. Najlepiej jest to robić z poziomu panelu administracyjnego, ale jeśli z jakiegoś powodu nie mamy do niego dostępu, można poradzić sobie, właśnie bezpośrednio modyfikując bazę danych. W istocie nie ma możliwości odzyskania hasła z bazy danych, ponieważ hasła są przechowywane w zaszyfrowanej formie i zamiast nich widać tylko nic nam nie mówiący ciąg znaków. Można natomiast zmienić hasło na coś innego. Trzeba tylko pamiętać, że musi ono być zaszyfrowane algorytmem MD5, czego można dokonać np. w narzędziu phpMyAdmin i dowolnym narzędziu do zarządzania bazami danych MySQL. Należy wpisać hasło jako zwykły tekst i dla pola, w którym się je wpisało, wybrać opcję MD5. W efekcie otrzymasz bezładny ciąg znaków. Jeśli wydaje Ci się to straszne, to lepiej tego nie rób, tylko najpierw więcej poczytaj na ten temat.



Czasami też chciałoby się dokonać masowej edycji wpisów. Gdy np. witrynę przeniesie się pod nowy adres, trzeba zmienić adresy wszystkich obrazów graficznych, np. z *staradomena.pl/wp-content/obrazek.png* na *nowadomena.pl/wp-content/obrazek.png*. Do zmiany ścieżek można użyć specjalnych wtyczek, co jest lepszym rozwiązaniem. Jeśli jednak biegle posługujesz się bazą danych, możesz napisać specjalne zapytanie SQL, które spowoduje znalezienie wszystkich wystąpień określonego ciągu znaków i zamienienie ich na inny, na przykład:

```
UPDATE wp_posts SET post_content = REPLACE (
  post_content,
  'staradomena.pl/wp-content/',
  'nowadomena.pl/wp-content/');
```

To zapytanie spowoduje wyszukanie w tabeli `wp_posts` wszystkich wystąpień ciągu `staradomena.pl/wp-content/` i zastąpienie ich ciągiem `nowadomena.pl/wp-content/`. W ten sposób zostaną naprawione wszystkie odwołania do grafik. Sprytne zapytania SQL do masowego dokonywania zmian są bardzo przydatne, ale pamiętaj: w bazie danych niczego nie można cofnąć i dlatego przed wykonaniem jakichkolwiek czynności zawsze najpierw rób kopię zapasową całej bazy.

# ROBIENIE KOPII ZAPASOWEJ

Każdy, kto choć raz stracił wszystkie dane z powodu awarii dysku twardego, wie, jak ważne jest wykonywanie kopii zapasowej ważnych plików. To samo oczywiście dotyczy treści przechowywanych na serwerach. Robienie kopii zapasowej WordPressa to proces dwuetapowy, ponieważ każda witryna oparta na tym systemie składa się z plików statycznych (obrazy

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

graficzne i inne załączniki) i bazy danych (cała treść). Wśród plików znajdują się pliki motywu i wtyczek, które mogłeś zmodyfikować, ale nawet jeśli tego nie zrobiłeś i tak lepiej jest ich nie stracić, aby w razie awarii nie musieć ich później wszystkich zbierać od nowa. Odkąd do panelu administracyjnego została dodana znakomita funkcja automatycznej aktualizacji WordPressa, robienie kopii zapasowej stało się jeszcze bardziej ważne.

Żadnych problemów nie sprawi Ci praktycznie jedynie utrata plików rdzenia WordPressa. Zawsze można je pobrać ponownie, chociaż czasami warto mieć przynajmniej kopię zapasową pliku *wp-config.php*.

Kopię zapasową bazy danych można wykonać na wiele sposobów. Najbardziej oczywistym rozwiązaniem jest użycie narzędzia typu phpMyAdmin do pobrania skompresowanego archiwum z całą zawartością bazy. Opis tej metody znajduje się w dalszej części rozdziału, a więc na razie pominę szczegóły. Wadą tej metody jest to, że trzeba pamiętać o regularnym wykonywaniu kopii zapasowej, co może być trudne. Ponadto phpMyAdmin i podobne narzędzia nie należą do najłatwiejszych w obsłudze, przez co mało kto chce z nich korzystać, jeśli nie musi.

Na szczęście istnieje masa wtyczek do WordPressa, za pomocą których można automatycznie wykonywać kopię zapasową bazy danych i wysyłać ją na wybrany adres e-mail lub do jakiejś usługi synchronizacji. Wtyczek tego typu jest kilka, a niektóre najlepsze z nich zostały wymienione w dodatku A.



To była treść witryny. Teraz zajmiemy się plikami. Robienie kopii zapasowej plików statycznych WordPressa jest bardzo łatwe — po prostu trzeba regularnie kopiować folder *wp-content*. W folderze tym przechowywane są wszystkie pliki wysłane do witryny (obrazy, filmy itp.), a także motywy i wtyczki. Jest to jedyny folder w WordPressie, w którym cokolwiek powinno się zmieniać. Oprócz tego zmiany czasami wprowadza się jeszcze w trzech plikach: *wp-config.php*, *.htaccess* i *index.php* znajdującym się w katalogu głównym. Kopiując plik *wp-content*, zabezpieczysz wszystkie swoje pliki statyczne, motywy i wtyczki, pod warunkiem że nie zmieniłeś standardowych ustawień, aby te rzeczy trzymać gdzie indziej.

W jaki sposób można wykonać kopię zapasową folderu *wp-content*? Najprostsza metoda, czyli pobieranie plików poprzez FTP, jest niestety zarazem kłopotliwa, ponieważ wymaga od nas, abyśmy o niej pamiętali. Niektóre konta hostingowe mają sprytne skrypty wysyłające kopie zapasowe do zewnętrznych usług, takich jak Amazon S3 czy serwer FTP. Rozwiązania te są tanie i warto z nich korzystać, zamiast polegać na swojej pamięci, aby co określony czas pobrać pliki z serwera. Co więcej, często te narzędzia obsługują także bazy danych, a więc można przy ich użyciu zrobić pełną kopię zapasową witryny. Lepiej dmuchać na zimne, niż się sparzyć.

Ostatnią deską ratunku w sytuacji, gdy nastąpi utrata danych, jest kopia zapasowa zrobiona przez firmę hostingową. Oczywiście nikt mnie nie przekona, żebym bezgranicznie ufał firmie hostingowej, choćby nie wiem jak dobra była. Niektóre firmy hostingowe naprawdę spełniają obietnice i robią co godzinę kopie zapasowe, łączą dyski w RAID i stosują inne sprytne zabezpieczenia danych, jednak nawet najlepiej zaplanowane rozwiązanie może okazać się

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

nieskuteczne albo ulec awarii. Większość firm hostingowych automatycznie wykonuje kopie zapasowe, ale co się stanie, gdy całe centrum danych ulegnie awarii albo wyłączą prąd? Może Ci się wydawać, że w tych czasach takie rzeczy się nie zdarzają, ale skoro nawet Google może chwilowo zniknąć z internetu, to Twój host również.

Innymi słowy, zawsze miej własną kopię zapasową swoich danych. Miejmy nadzieję, że nigdy Ci nie będzie potrzebna, ale będziesz spokojniejszy, wiedząc, że pomyślałeś też o tym już na samym początku.

# ZMIENIANIE HOSTA

Czasami zmuszeni jesteśmy do zmiany hostingu. Może być to konieczne, gdy nasza witryna rozrośnie się i obecny hosting stanie się niewystarczający (gratulacje!) albo jakość usług, z których korzystamy, drastycznie spadnie. Niezależnie od tego, jaki może być tego powód, zmiany hostingu zdarzają się stosunkowo często. Proces ten wymaga przekierowania domen na inne serwery, a także przeniesienia plików i bazy danych. W tym podrozdziale znajdziesz tylko wskazówki na ten temat dotyczące samego WordPressa. Jeśli potrzebujesz pomocy w przenoszeniu domeny, skrzynek pocztowych itp., skontaktuj się z nowym administratorem, który powinien być w stanie Ci pomóc.



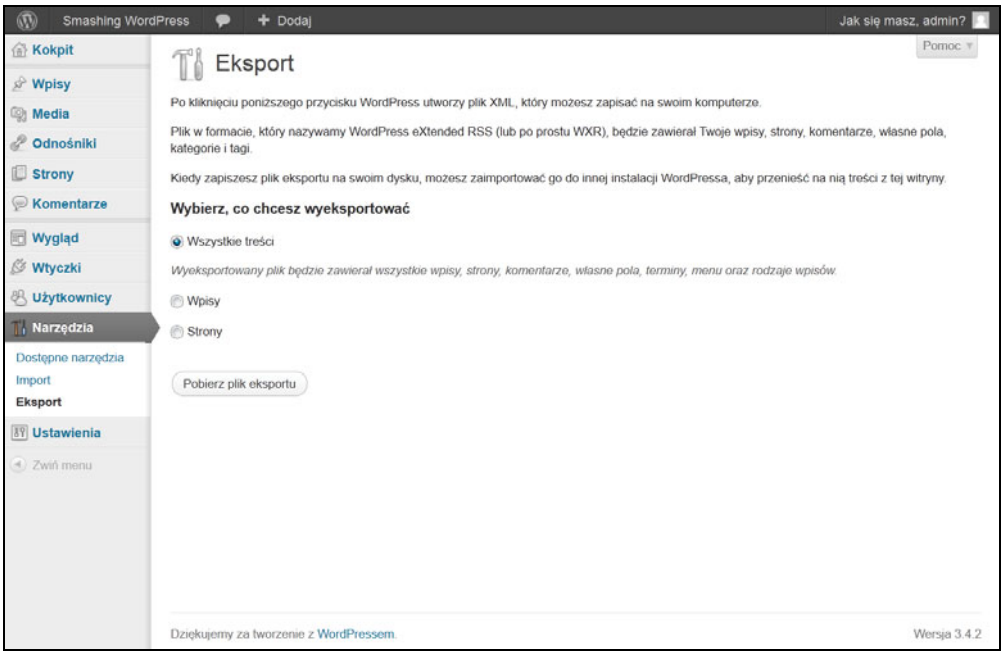
Jest kilka sposobów na przeniesienie witryny na nowy serwer. Ja preferuję metodę z użyciem funkcji eksportu i importu dostępnych w narzędziach w panelu administracyjnym.

## NARZĘDZIA EKSPORTU I IMPORTU

Kiedyś w panelu administracyjnym WordPressa znajdowało się kilka narzędzi do eksportu i importu danych, ale zamieniono je na wtyczki. Dlatego gdy będziesz chciał coś wyeksportować lub zaimportować, możesz być zmuszony zainstalować jakąś wtyczkę. Nie jest to problem. Ponadto przed rozpoczęciem przenosin upewnij się, że Twój WordPress jest w najnowszej wersji. Następnie przejdź do menu *Narzędzia* i wybierz opcję eksportu treści, jak pokazano na rysunku 1.5. W efekcie otrzymasz plik z danymi.

Teraz zainstaluj WordPressa na nowym serwerze. Każda porządna firma hostingowa udostępnia zastępczy adres URL do konta, aby można było z niego korzystać także bez podpinania domeny. Po zainstalowaniu WordPressa usuń standardowe strony i wpisy, ponieważ podczas importu nie zostaną one zlikwidowane. Przecież nie chcesz pozostawić w witrynie przykładowych elementów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974



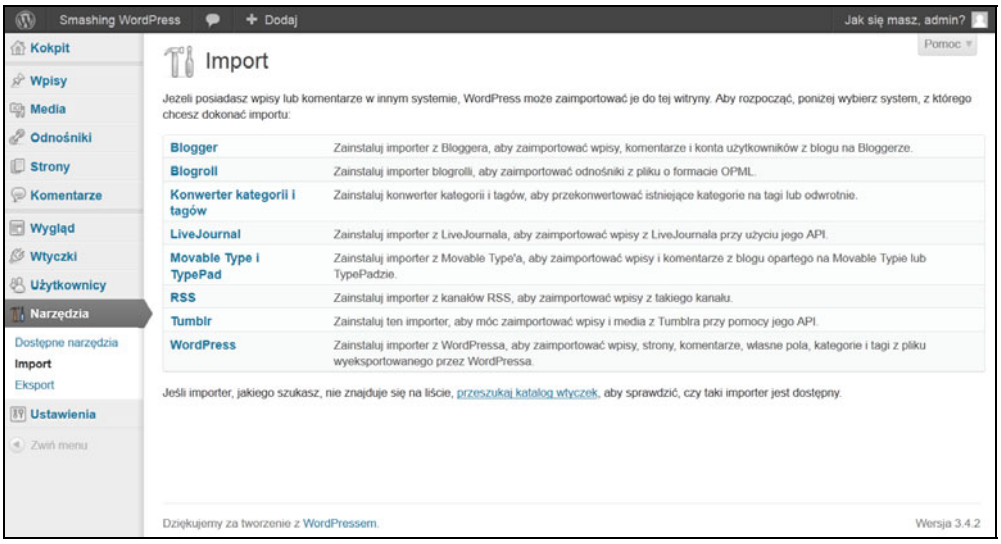

Rysunek 1.5. Eksportowanie danych



Następnie pobierz folder *wp-content* ze starego serwera i wyślij go na nowy serwer. Dzięki temu wszystkie Twoje obrazy, filmy, motywy i wtyczki będą już na swoim miejscu. W narzędziu do importowania danych dostępna jest opcja pozwalająca pobrać obrazy z wpisów na nowy serwer, ale niestety rzadko działa tak, jak powinna. Dlatego lepiej jest przenieść pliki własnoręcznie, korzystając z FTP.

Teraz możesz zaimportować plik wyeksportowany ze starego serwera. Kliknij pozycję *Narzędzia* (rysunek 1.6), a następnie uruchom kreatora importu i postępuj zgodnie z wyświetlanymi wskazówkami. Gdy skrypt zakończy pracę, import będzie zakończony! Sprawdź, czy wszystko działa prawidłowo, poklep się z uznaniem po plecach i skieruj domenę na nowy serwer. Możliwe, że będzie konieczna zmiana ustawień w nowym blogu, ponieważ system mógł wykorzystywać ścieżki wewnętrzne ze starego hostingu. W razie potrzeby zmień je na nowe. Dopóki domena nie zostanie przekierowana na nowy serwer, Twoja nowa witryna nie będzie poprawnie działać. Ale to nie problem, ponieważ będzie cały czas działała na starym hostingu. Dobrze jest czasowo wyłączyć możliwość dodawania komentarzy, ponieważ znikną one, gdy serwis zacznie być serwowany z nowego konta.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 1.6. W WordPressie istnieje możliwość importu danych z różnych systemów

## PROBLEMY Z IMPORTOWANIEM I EKSPORTOWANIEM DANYCH

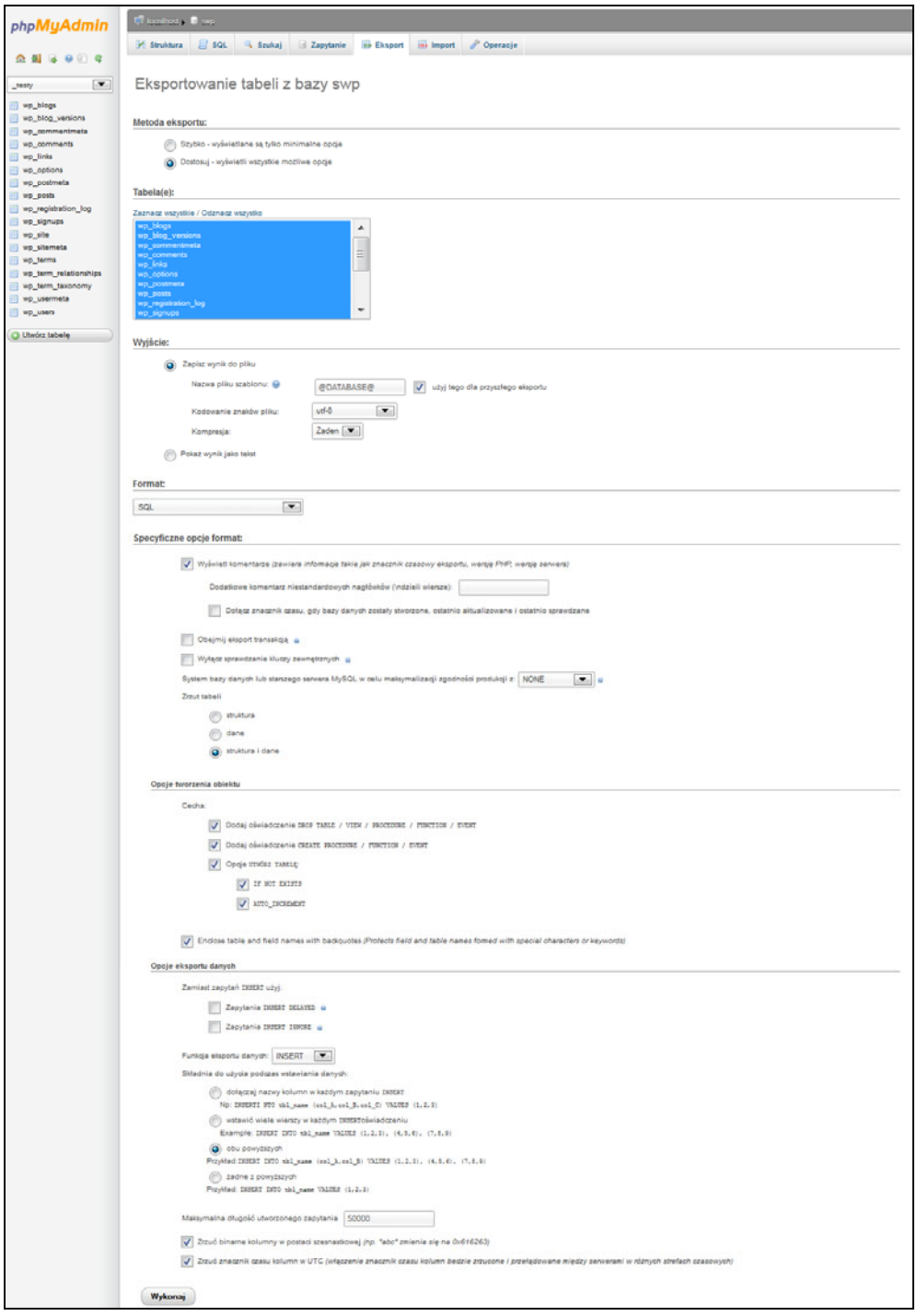


Zdarza się, że funkcje importu i eksportu nie chcą działać — najczęściej spowodowane jest to zbyt dużą ilością danych do przetworzenia dla PHP. Każdy hosting ma ustawione określone limity i nie da się ich ominąć, ale problemy te dotyczą głównie tylko bardzo dużych witryn.

Jeśli masz ten problem, musisz poradzić sobie w inny sposób. Najlepiej gdybyś miał możliwość odtworzenia całego swojego środowiska na nowym serwerze, z bazą danych o takiej samej nazwie oraz taką samą nazwą użytkownika i identycznym hasłem. Jeśli możesz to zrobić, to przenosiny będą łatwe. Wystarczy, że wykonasz zrzut bazy danych za pomocą specjalnego narzędzia dostępnego w panelu administratora konta serwerowego, a następnie zaimportujesz go na nowym serwerze. Do tego celu zwykle używa się narzędzia phpMyAdmin oraz postępuje się zgodnie ze wskazówkami zawartymi w dokumentacji WordPressa (*http://codex.wordpress.org/Backing_Up_Your_Database*):

1. Zaloguj się do phpMyAdmina i wybierz bazę danych, którą chcesz skopiować.
2. Przejdź na kartę *Eksport*, klikając odnośnik znajdujący się w górnej części strony (rysunek 1.7).
3. Zaznacz wszystkie tabele, klikając opcję *Zaznacz wszystkie*. Jeśli nie przechowujesz w bazie niczego innego, to w ten sposób zaznaczysz wszystko, czego potrzebujesz.
4. Niżej znajdź sekcję *Specyficzne opcje format* i zaznacz w niej pole *Struktura i dane*, następnie w sekcji *Opcje tworzenia obiektu* zaznacz opcje *Dodaj oświadczenie DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT*, *AUTO_INCREMENT* oraz *Enclose table and field names with backquotes* (umieść nazwy tabel i pól w pojedynczych cudzysłowach).

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 1.7. Narzędzie phpMyAdmin jest bardziej skomplikowane od WordPressa, ale bardzo dobrze działa



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

5. Przewiń stronę w dół, aby znaleźć i kliknąć przycisk *Wykonaj*. Pojawi się okno zapisu pliku, w którym należy kliknąć przycisk *OK* (jeśli w znajdującej się na górze sekcji *Kompresja* wybrałeś format kompresji, to zostanie wygenerowany plik w wybranym przez Ciebie formacie, np. ZIP).
6. Po kliknięciu przycisku *OK* nastąpi pobranie zawartości bazy danych, którą będzie można następnie zaimportować na nowym serwerze.
7. Importowanie zrzutu bazy w narzędziu phpMyAdmin jest jeszcze łatwiejsze od eksportowania. Należy tylko utworzyć bazę danych o takiej samej nazwie jak poprzednia oraz takiego samego użytkownika z identycznym hasłem. Dzięki temu nie będzie trzeba niczego zmieniać w pliku *config.php*.
8. Zaimportuj zrzut bazy danych, korzystając z menedżera baz danych MySQL. Jeśli używasz phpMyAdmina, to wystarczy wybrać przygotowaną bazę danych, przejść na kartę *Import*, która znajduje się obok karty *Eksport*, i znaleźć plik eksportu na dysku, a następnie go zaimportować.
9. Na koniec pobierz ze starego serwera wszystkie pliki WordPressa i wyślij je w ten sam sposób na nowy serwer. Posprawdzaj, czy wszystko działa, korzystając z dostępnego tymczasowego adresu i jeśli wszystko będzie wyglądać w porządku, przekieruj domenę na nowy serwer. Gdy przekierowanie zostanie wykonane, witryna powinna działać jak wcześniej.



Czasami jednak nie da się odtworzyć całego środowiska ze starego serwera. Wówczas trzeba wprowadzić odpowiednie zmiany w pliku *wp-config.php*, czyli najczęściej zmienić tylko nazwę użytkownika i bazy danych oraz hasło. Czasami też trzeba wpisać adres zewnętrznego serwera baz danych.

Przenoszenie WordPressa na inny serwer może Ci się w pierwszej chwili wydawać bardzo trudne, ale w rzeczywistości wcale tak nie jest. Oczywiście jeśli masz dużą witrynę i nie masz doświadczenia w pracy z narzędziami do zarządzania bazami danych na serwerze, takimi jak phpMyAdmin, może być Ci ciężko. W takim przypadku najlepiej poszukać pomocy albo spróbować uporać się z tym samodzielnie. Pamiętaj tylko o wykonaniu kopii zapasowej wszystkiego, co może być Ci potrzebne, i nie zepsuj niczego na starym (bieżącym) serwerze. Dzięki temu jeśli coś Ci się nie uda na nowym serwerze, zawsze będziesz mógł zacząć wszystko od początku.

# ZABEZPIECZANIE WORDPRESSA

Jest kilka prostych i parę trudnych sposobów na zwiększenie bezpieczeństwa systemu WordPress. Podstawą jest instalowanie aktualizacji, gdy tylko się pojawią. Każda nowa wersja ma jakieś nowe funkcje, ale przede wszystkim zawiera łaty znanych luk i poprawki błędów, które mogą narażać Twoją witrynę na niebezpieczeństwo, jeśli nie dokonasz aktualizacji.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*Należy również w pliku* wp-config.php *zapisać klucze tajne. Dzięki nim instalacja jest bardziej bezpieczna. Więcej informacji na ten temat znajduje się we wcześniejszym podrozdziale „Podstawowa instalacja". Zazwyczaj klucze te są wstawione standardowo, ale niektóre automatyczne instalatory ich nie wstawiają. Dlatego lepiej jest sprawdzić, czy w pliku* wp-config.php *wstawione są klucze tajne, i w razie potrzeby je wstawić.*

## UŻYTKOWNICY I HASŁA

Pierwsza rzecz, jaką robię po zainstalowaniu WordPressa, to utworzenie nowego użytkownika z uprawnieniami administratora i loguję się przy użyciu tego konta zamiast domyślnego konta o nazwie *admin*. Dlaczego to robię? Ponieważ każdy wie, że konto o nazwie *admin* ma zwykle uprawnienia administratora. Dlatego każdy, kto chce się włamać do WordPressa, zaczyna od sprawdzenia, czy istnieje konto o takiej nazwie i próby złamania jego hasła metodą siłową. Gdy atak się powiedzie, haker ma pełną kontrolę nad naszą witryną. W związku z tym najlepiej po utworzeniu nowego konta administracyjnego usunąć konto o nazwie *admin*, ponieważ spełniło ono już swoje zadanie i nie jest potrzebne.

Oczywiście usunięcie konta o nazwie *admin* nie stanowi gwarancji, że hakerzy nie namierzą innego użytkownika, którego warto atakować. Jeśli wyświetlasz w swojej witrynie archiwa użytkowników, to znajdą Cię w nich. Rozwiązaniem może być ich niewyświetlanie albo usunięcie łączy do stron autorów (innych niż te, które zostały utworzone poza wewnętrznymi funkcjami WordPressa). Ale co zrobić, gdy są nam one potrzebne? W WordPressie jest mnóstwo miejsc, w których można pobrać nazwy użytkowników, także wiele motywów je wykorzystuje, co sprawia, że jest to jeszcze łatwiejsze. Zatem chyba się zgodzisz, że nie ma żadnego powodu, aby pozostawiać użytkownika o nazwie *admin*, o którym wszyscy wiedzą, że ma uprawnienia administratora.



Przede wszystkim należy oszczędnie gospodarować uprawnieniami kont. Do publikowania i edytowania wpisów oraz stron nie potrzebujesz konta administratora. Do tego wystarczą Ci uprawnienia redaktora. Wprawdzie jeśli ktoś włamie się na konto redaktora, to również będzie mógł narobić dużych szkód, ale mimo wszystko nie będzie miał dostępu do narzędzi administracyjnych. Poza tym przecież i tak regularnie robisz kopię zapasową danych, prawda?

Kolejnym wrażliwym punktem są hasła. Zapewne nieraz słyszałeś radę, żeby używać długich haseł składających się z liter, cyfr, znaków specjalnych itd. Zastosuj się do niej. Im bardziej skomplikowane hasło, tym trudniej jest je złamać.

## ZABEZPIECZENIA PO STRONIE SERWERA

Użytkownik bazy danych MySQL, który nie powinien być współużytkowany z innym systemem, nie potrzebuje wszystkich uprawnień zapisu. Nie musi mieć możliwości blokowania tabel i indeksów czy tworzenia tymczasowych tabel, referencji i procedur. Innymi słowy, możesz ograniczyć mu trochę uprawnienia, aby podnieść poziom bezpieczeństwa systemu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Niektórzy zalecają też wymuszenie dodatkowego logowania za pomocą pliku *.htaccess* serwera Apache. Sam nie stosuję tej metody, ponieważ denerwują mnie te formularze. Poza tym istnieją wtyczki, które lepiej rozwiązują ten problem (poszukaj ich w dodatku A).

Kolejną rzeczą, jaką możesz zrobić, jest umieszczenie pustego pliku *index.php* lub *index.html* w każdym folderze, w którym nie ma pliku indeksowego. W WordPressie pliki te są utworzone standardowo, ale nie szkodzi sprawdzić. W ten sposób uniemożliwia się bezpośrednie przeglądanie zawartości katalogów na serwerze, co niektóre firmy hostingowe umożliwiają.

Ten sam efekt można uzyskać, stosując poniższy wpis w pliku *.htaccess*, który wyłączy wyświetlanie zawartości katalogów dla całego konta hostingowego:

```
# Wyłączenie możliwości przeglądania katalogów
Options All -Indexes
```

Opcja `-Indexes` wyłącza możliwość przeglądania zawartości katalogów. Jej przeciwieństwem jest opcja `+Indexes`.

Kolejną rzeczą, jaką możemy zrobić, jest wymuszenie szyfrowania SSL dla połączenia administratora z WordPressem. W ten sposób znacznie utrudnisz przechwytywanie danych przesyłanych przez Ciebie do serwera podczas zarządzania witryną. Włączenie SSL jest bardzo łatwe. Wystarczy w pliku *wp-config.php*, nad komentarzem `/* To wszystko, zakończ edycję w tym miejscu! Miłego blogowania! */`, umieścić poniższy wiersz kodu:



```
define('FORCE_SSL_ADMIN', true);
```

Oczywiście aby SSL działało, musi być obsługiwane przez serwer. Niektóre firmy udostępniają wszystko, czego trzeba, aby włączyć tę usługę w panelu administracyjnym. Są jednak i takie, które robią to za użytkownika i mogą nawet za to pobierać dodatkowe opłaty. Przy okazji: z FTP łącz się, korzystając z bezpiecznego protokołu SFTP. Jeśli Twój host nie obsługuje zabezpieczonych połączeń FTP, poproś o ich włączenie.

Na koniec warto jeszcze sprawdzić, czy wszystkie pliki na serwerze mają odpowiednio ustawione uprawnienia dostępu. Pliki powinny być własnością konta użytkownika z możliwością zapisu — uprawnienia 644. Foldery powinny mieć uprawnienia 755. Aby skonfigurować te ustawienia, musisz mieć możliwość wykonywania polecenia `chmod` albo skorzystać z klienta FTP. Jeśli nie jesteś pewien, co należy zrobić, poproś o pomoc administratora serwera. Więcej informacji na temat plików i folderów oraz uprawnień dostępu można znaleźć w dokumentacji WordPressa pod adresem *http://codex.wordpress.org/Changing_File_Permissions*.

# W NASTĘPNYM ROZDZIALE

Nieważne, czy to jest Twoje pierwsze spotkanie z cudownym światem WordPressa, czy jesteś doświadczonym użytkownikiem i programistą tego systemu. Ważne jest to, że wiesz, jak go prawidłowo zainstalować i zabezpieczyć, oraz że rozumiesz, jak on działa. Teraz nauczysz się tworzyć witryny i wtyczki, aby osiągnąć zamierzone cele.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W następnych rozdziałach szczegółowo poznasz budowę i działanie WordPressa, to znaczy nauczysz się programować pętlę, wtyczki i motywy oraz uruchomisz kreatywne obszary swojego mózgu. Geniusz WordPressa tkwi w jego możliwościach adaptacji, dzięki którym przy jego użyciu można zrobić prawie wszystko. Samo myślenie o tym, co można zrobić, może być inspiracją do działania.

Jeśli masz do dyspozycji instalację WordPressa, na której możesz robić testy (najlepiej niedostępną publicznie, ponieważ w czasie tych testów wiele może się zdarzyć), to zbierz swoje zabawki i przygotuj się do zabawy ze składnią WordPressa.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 2 SKŁADNIA WORDPRESSA

**MASZ JUŻ WORDPRESSA** zainstalowanego na serwerze, a więc możesz zacząć go używać. W tym rozdziale poznasz ogólne zasady działania systemu. Nie opisuję tu szczegółowo każdego pliku, tylko staram się wprowadzić Cię w tematykę tego, w jaki sposób WordPress zwraca treść. Poznasz ważne pojęcie tagów szablonowych i warunkowych oraz dowiesz się, w jaki sposób kontrolować ich zachowanie za pomocą parametrów. Ponadto przyjrzysz się motywom i dowiesz się, jak są zbudowane, dzięki czemu jeszcze lepiej zrozumiesz działanie witryn opartych na WordPressie.

Zatem zaczynamy!

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# WORDPRESS I PHP

Od tego miejsca bardzo pomocna będzie znajomość języka PHP, a także języków HTML i CSS. Jeśli technologie te nie są Ci znane, najpierw poczytaj trochę na ich temat. Na początek dobrym, choć dość trudnym w odbiorze źródłem informacji mogą być dokumentacje techniczne W3 (*www.w3.org*) i kurs Zend PHP 101 (*http://devzone.zend.com/article/627*). Nie musisz być ekspertem w tych dziedzinach, ale podstawowa wiedza jest niezbędna.

System WordPress jest napisany w PHP, popularnym języku skryptowym, przy użyciu którego programista może napisać praktycznie wszystko, co chce. Jeśli choć odrobinę znasz język PHP, szybko zorientujesz się w WordPressie i jego różnych funkcjach służących do tworzenia wtyczek i motywów. Nie musisz jednak być programistą PHP, aby móc robić wiele ciekawych rzeczy z WordPressem. Oczywiście znajomość tego języka jest potrzebna np. do pisania wtyczek, ale korzystając z wbudowanych tagów szablonowych, których używa się w motywach, również wiele da się zdziałać. Tagi szablonowe i warunkowe pozwalają na robienie różnych rzeczy w WordPressie bez konieczności pisania nowych funkcji.

Czy to dla Ciebie chińszczyzna? Nie martw się, nawet jeśli nigdy nie napisałeś skryptu *Witaj, świecie* w PHP, po lekturze tej książki będziesz umiał stworzyć każdą witrynę w WordPressie.



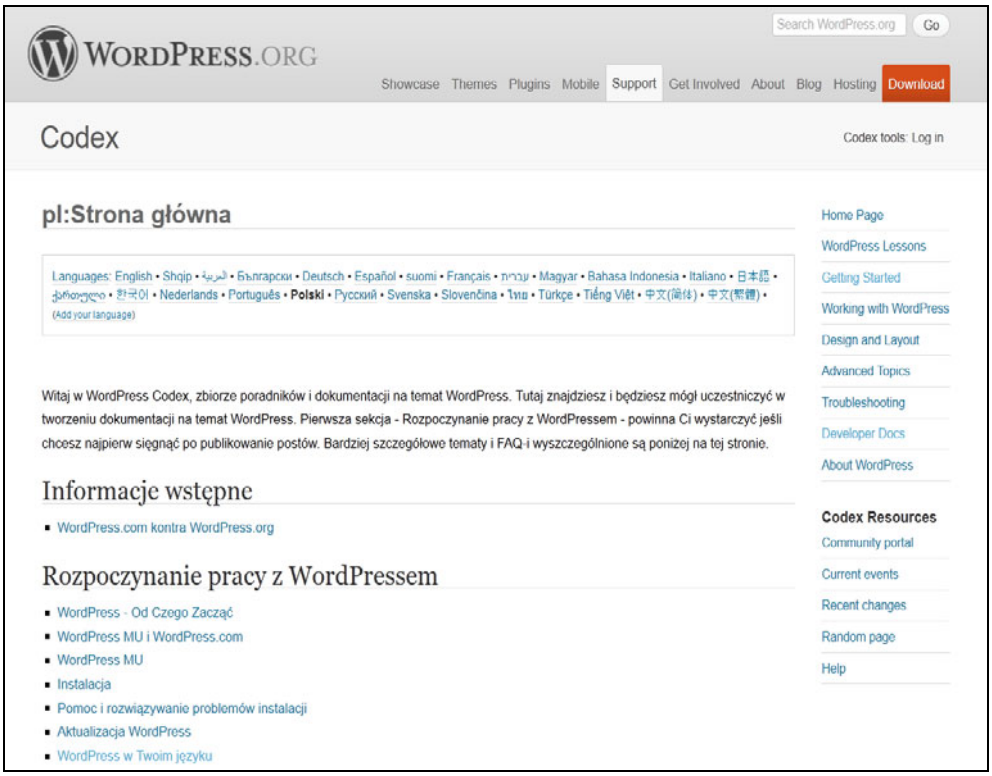
# DOKUMENTACJA WORDPRESSA

Dokumentacja WordPressa nazywa się Codex (rysunek 2.1), ma postać podręcznika wiki, a jej polska wersja znajduje się pod adresem *http://codex.wordpress.org/pl:Strona_główna*. Jest bardzo przydatna, zwłaszcza dla początkujących. Powinieneś się z nią zapoznać i zapisać ją sobie w ulubionych. Gdy podczas lektury dalszych rozdziałów tej książki poczujesz chęć stworzenia czegoś na własną rękę albo dowiedzenia się więcej na jakiś temat, to Codex jest pierwszym miejscem, w którym należy szukać dodatkowych informacji. Podczas gdy dokumentacja przede wszystkim zawiera informacje i poradniki w zwięzłej formie, są dwie strony, do których na pewno będziesz często wracać: *http://codex.wordpress.org/Template_Tags* (zwięzły opis tagów) i *http://codex.wordpress.org/Function_Reference* (opis funkcji dla bardziej zaawansowanych).

# RDZEŃ WORDPRESSA

W każdym dobrym systemie zarządzania treścią (CMS) pliki rdzenia są oddzielone od pozostałych plików, dzięki czemu wprowadzane zmiany nie powodują awarii w samym systemie. WordPress nie jest w tej kwestii wyjątkiem. W przypadku tego systemu **rdzeniem** są wewnętrzne pliki, na których opiera się jego działanie. Nie należy ich nigdy zmieniać, ponieważ każda aktualizacja systemu spowoduje utratę tych zmian. Ponadto modyfikowanie plików rdzenia może spowodować wadliwe działanie motywów i wtyczek oraz narazić witrynę na ataki hakerów. Krótko mówiąc, nie należy ruszać plików rdzenia systemu WordPress. Jedyny plik z tej kategorii, który można edytować, to *wp-config.php* (zobacz rozdział 1.) zawierający dane konfiguracyjne instalacji.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 2.1. Jak najszybciej zapoznaj się z dokumentacją WordPressa



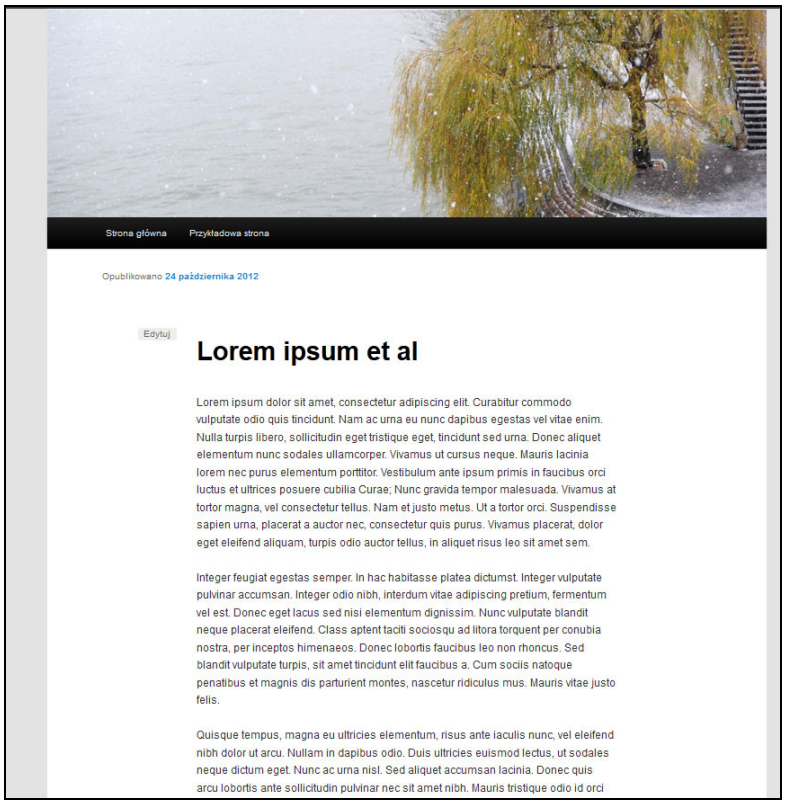
Do rdzenia nie należy folder *wp-content*, w którym zapisuje się motywy, wtyczki i pliki wysyłane do witryny. Wszystkie te dodatki działają jak nadbudówka na WordPressie i nie ma ryzyka uszkodzenia przez nie rdzennych plików (chyba że zainstaluje się jakiś złośliwy kod, ale to już inna historia) podczas prac nad witryną. Ogólnie rzecz biorąc, z folderu *wp-content* wychodzimy tylko raz — podczas instalacji systemu i ewentualnie przy przenoszeniu witryny na nowy serwer. Oczywiście można też robić różne ciekawe rzeczy w pliku *.htaccess* i istnieją wtyczki wymagające wykonania pewnych czynności poza folderem *wp-content*. Nie ma w tym nic złego, ale trzeba uważać. Innymi słowy, gdy ktoś poprosi Cię o modyfikację pliku znajdującego się poza folderem *wp-content*, bądź czujny. Tworzenie nowych plików to jedno, ale modyfikowanie istniejących, oprócz *wp-config.php* i *.htaccess*, jest wykluczone.

Cały czas podkreślam, że rdzeń WordPressa jest nietykalny. Nie ruszaj go, jeśli nie jest to absolutnie konieczne. A jeśli już uznasz, że musisz coś w nim zmienić, zastanów się jeszcze dwa razy, bo prawie na pewno da się to zrobić inaczej. Modyfikowanie rdzenia WordPressa jest złym pomysłem i dlatego właśnie motywy i wtyczki do tego systemu działają tak dobrze. Mimo wszystko należy jednak pamiętać, że WordPress to platforma o otwartym kodzie źródłowym, a więc można z nim robić, co się chce. Możesz nawet, jeśli chcesz, utworzyć jego nową gałąź i zacząć niezależnie rozwijać własny system. W tej książce przyjmuję założenie, że nie interesuje Cię tworzenie własnej platformy i że wolisz używać standardowego WordPressa, a to oznacza, że nie powinieneś modyfikować plików rdzenia. Pamiętaj jednak, że gdybyś się uparł, możesz to robić.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# MOTYWY I SZABLONY

Mówiąc najprościej, **motyw** to skóra Twojej witryny. Do wyboru masz wiele różnych motywów i każdy z nich będzie przedstawiał Twoją treść w inny sposób, jak pokazano na rysunkach 2.2 i 2.3, na których widać ten sam wpis, ale wyświetlony w dwóch różnych motywach.




Rysunek 2.2. Na tym rysunku widać przykładowy post w motywie Twenty Eleven

Możesz użyć podstawowego motywu, który tylko wyświetla treść w prosty sposób, albo całkowicie zmienić wygląd witryny, korzystając z motywu, który nada jej całkiem nowy styl.

Każdy motyw musi zawierać plik arkuszy stylów o nazwie *style.css*. W pliku tym przechowywane są podstawowe style witryny oraz nazwa motywu i najważniejsze informacje o tym motywie. Oprócz tego motyw składa się z pewnej liczby plików PHP, z których część jest niezbędna, a część pełni tylko rolę pomocniczą, aby ułatwić pracę użytkownikowi. Pliki te nazywają się **szablonami**. Główny plik motywu ma nazwę *index.php*. Służy on do wyświetlania treści i wyników wyszukiwania oraz jest używany, gdy nie ma żadnych innych plików szablonów. Inne typowe pliki szablonów to *sidebar.php*, zawierający treść paska bocznego, *header.php* zawierający treść nagłówka, *comments.php* zawierający logikę obsługi komentarzy oraz *footer.php* reprezentujący stopkę. Dodatkowo motyw może zawierać pliki: *single.php* reprezentujący stronę z wyświetlonym pojedynczym wpisem, *page.php* reprezentujący statyczne strony WordPressa, specjalny plik do obsługi wyszukiwarki o nazwie *search.php*,



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Rysunek 2.3. Ten sam wpis co na rysunku 2.2, ale w motywie Notes Blog

plik obsługujący wyświetlanie list wpisów w poszczególnych kategoriach *category.php* itd. Takich szablonowych plików można dodać do WordPressa dowolną ilość, co jest jedną z przyczyn decydujących o tym, że system ten jest tak wszechstronny. Zasadniczo motyw może składać się tylko z pliku *style.css* i pliku *index.php*, ale większość motywów zawiera ich więcej.

Przy użyciu szablonów, funkcji WordPressa, wtyczek i języka PHP można utworzyć praktycznie każdą witrynę internetową! Nie chcesz żadnych komentarzy? Po prostu usuń odpowiedni kod! Chcesz, żeby strona jakiegoś produktu wyglądała całkiem inaczej niż pozostałe strony? Utwórz szablon o takim wyglądzie, jaki Cię interesuje. Takie przykłady mnożna mnożyć, a w dalszej części książki zobaczysz witryny, które w ogóle nie będą przypominały typowego bloga.

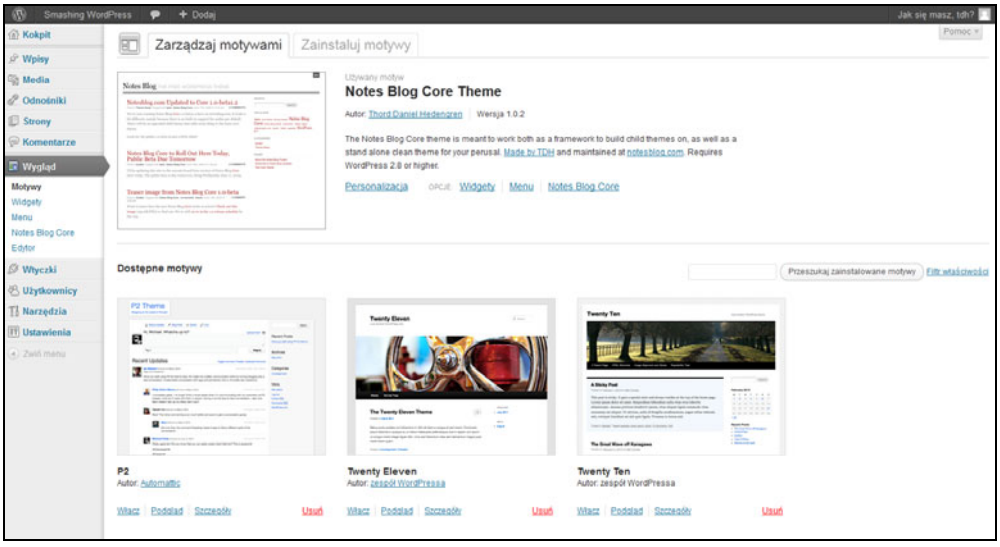
Aby było ciekawiej, dodam, że jest jeszcze jeden plik, za pomocą którego można rozszerzać funkcjonalność witryny. Jest to plik *functions.php*, w którym można dodawać funkcje na zasadzie podobnej do wtyczek. Ponadto każdy plik szablonowy może zawierać kod PHP niezwiązany bezpośrednio z samym WordPressem. A przecież nie wspomniałem jeszcze ani słowem o **widżetach**, czyli dodatkach, które można umieszczać w specjalnych obszarach witryny



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

w panelu administratora. Można też oddzielić pętlę, czyli kod zwracający główną treść w motywach, od plików szablonowych i umieścić ją w pliku *loop.php*. Szczegółowy opis pętli znajduje się w rozdziale 3.

Na rysunku 2.4 przedstawiona jest strona zarządzania motywami w panelu administracyjnym. Można na niej jednym kliknięciem zmienić motyw witryny, a także pobrać nowe motywy z repozytorium motywów portalu WordPress.org.




Rysunek 2.4. Zarządzanie motywami w panelu administracyjnym WordPressa jest bardzo łatwe

Najłatwiej zrozumieć sposób działania motywów, używając ich samemu. Dlatego zainstaluj w swojej testowej witrynie jakiś motyw oraz przyjrzyj mu się i sprawdź, z jakich plików się składa. Nie zawracaj sobie głowy obrazami i arkuszem stylów, ponieważ to są elementy projektu wyglądu zewnętrznego. Przyjrzyj się natomiast plikom *index.php*, *header.php* i *footer.php*, aby dowiedzieć się, jak są zbudowane. Nie ma w nich nic skomplikowanego: najpierw ładowany jest plik *index.php* (albo inny plik szablonowy, np. *single.php*, *category.php* itd.), który następnie wczytuje kolejne pliki, takie jak *header.php*, *footer.php* i zazwyczaj też *sidebar.php*.

Motywami pobawisz się później. Na razie pamiętaj tylko, że najważniejsze wydarzenia mają miejsce w plikach szablonowych motywów. W nich znajduje się kod wysyłający na wyjście treść, którą opublikowałeś w WordPressie. Mimo że motywy mogą się od siebie różnić wyglądem i zachowaniem, tak naprawdę każdy z nich wyświetla tę samą treść, tylko w inny sposób, dzięki plikom szablonowym.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# TAGI SZABLONOWE

WordPress, mimo że został napisany w języku PHP, tak naprawdę sam w sobie jest szkieletem programistycznym. Można używać PHP do modyfikowania wtyczek i motywów, ale tak naprawdę większość czynności wykonuje się przy użyciu tagów szablonowych. Gdy otworzysz dowolny plik motywu z rozszerzeniem *.php*, np. *index.php* albo *single.php*, znajdziesz w nim funkcje wyglądające, jakby należały do języka PHP, na przykład:

```
<?php bloginfo( 'name' ); ?>
```

Jest to tag szablonowy zwracający nazwę bloga. Kod PHP, czyli w tym przypadku `<?php` i `;` `?>` , stanowi dla WordPressa wskazówkę, że wszystko, co znajduje się między tymi ciągami znaków, tu: tag `bloginfo()`, ma zostać przetworzone. W nawiasie w cudzysłowie znajduje się parametr o nazwie `'name'`, powodujący wysłanie na wyjście nazwy bloga.

Poniżej znajduje się przykład użycia tagu `get_template_directory_uri()` pobierającego URI (jak widać bez końcowego ukośnika) folderu zawierającego szablon, przy użyciu którego można odwołać się do obrazu. Poniższy kod powoduje wyświetlenie obrazu *smashing.gif* w pliku szablonowym:

```
<img src="<?php get_template_directory_uri(); ?>/smashing.gif" />
```

Zapewne rozpoznałeś element `img` języka HTML. Tag `get_template_directory_uri()` zwraca ścieżkę do folderu motywu. Wystarczy do niego dodać nazwę pliku graficznego *smashing.gif*, aby otrzymać pełną ścieżkę do obrazu. Oczywiście żeby to działało, obraz musi również znajdować się w folderze motywu.



Można zatem powiedzieć, że tagi szablonowe to w istocie funkcje PHP, które wykonują różne działania w zależności od przekazanych im parametrów. Niektóre z nich mają domyślne wartości i kilka opcji konfiguracyjnych, natomiast inne nie mają wartości domyślnych i są bardziej ubogie pod względem możliwości konfiguracji. Większości tagów można używać w dowolnym miejscu w plikach motywu, ale istnieje grupa takich, które mogą być używane wyłącznie w pętli (pętla to kod wyświetlający treść, np. wpisów i stron — więcej na jej temat piszę w następnym rozdziale).

Pełną listę tagów można znaleźć w dokumentacji pod adresem *http://codex.wordpress.org/Template_Tags*. Zaglądaj do niej zawsze, gdy chcesz zrobić coś nietypowego w swoich motywach, a także gdy chcesz zmodyfikować jakiś już istniejący motyw. Każdy tag jest zwięźle opisany i opatrzony przykładem użycia. To jest właśnie piękne w WordPressie: często można uzyskać żądany efekt bez znajomości PHP, kopiując tylko i wklejając w odpowiednim miejscu fragmenty kodu.

## TAGI DOŁĄCZANIA PLIKÓW

Istnieje grupa tagów szablonowych, które można znaleźć w prawie każdym pliku szablonowym motywu. **Tagi dołączania plików** to zwykłe funkcje PHP służące do wczytywania plików do innych plików. Innymi słowy, są one narzędziem ułatwiającym dołączanie do szablonu takich elementów jak nagłówek, stopka i pasek boczny:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php get_header(); ?>
<?php get_footer(); ?>
<?php get_sidebar(); ?>
```

Tagi dołączające pliki różnią się od pozostałych tagów tym, że nie dodają żadnych nowych funkcji ani elementów, tylko treść innych plików. Innymi słowy, tagi dołączające pliki dodają pliki szablonowe zawierające tagi szablonowe.

Znajdziesz je na przykład w pliku *index.php* swojego motywu. Za ich pomocą automatycznie dołączane są pliki *header.php*, *footer.php* i *sidebar.php* i są one wstawiane w miejscu wystąpienia tych tagów. Co więcej, te trzy wymienione tagi umożliwiają określanie alternatywnych nagłówków, stopek i pasków bocznych za pomocą parametrów:

```
<?php get_sidebar( 'left' ); ?>
```

Ten kod spowodowałby dołączenie pliku *sidebar-left.php*, a nie domyślnego *sidebar.php*, choć oczywiście trzeba by było ten dodatkowy plik utworzyć. Gdyby go nie było, zostałby automatycznie dołączony domyślny plik *sidebar.php*. Ta sama zasada dotyczy także tagów `get_header()` i `get_footer()`.

Czasami można też spotkać tag `get_template_part()` pobierający pętlę z jej pliku, na przykład:

```
<?php get_template_part( 'loop', 'index' ); ?>
```



Pierwszy parametr (`loop`) oznacza, że ma zostać pobrany plik szablonowy pętli. Natomiast drugi parametr powoduje, że WordPress będzie jej szukał w pliku o nazwie *loop-index.php*. Gdyby drugi parametr miał wartość `page`, WordPress szukałby pliku *loop-page.php*. Gdyby plik *loop-index.php* nie został znaleziony, WordPress użyłby domyślnego pliku *loop.php*. Bardziej szczegółowo opisuję to w rozdziale 3. To samo dotyczy drugiego parametru, tzn. niekoniecznie musi on mieć wartość `loop`.

Gdybyś chciał dołączyć do szablonu inne pliki, np. alternatywny szablon dla treści, do tego celu również mógłbyś użyć tagu `get_template_part()`:

```
<?php get_template_part( 'content' ); ?>
```

Pamiętaj tylko, że tag `get_template_part()` służy do dołączania fragmentów szablonów, a nie całych plików. Gdybyś z jakiegoś powodu chciał użyć zwykłej funkcji PHP, mógłbyś ją połączyć z tagiem `get_template_part()`, ale nie należy tego robić.

Istnieje też tag dołączający komentarze i w każdym dobrze zbudowanym motywie znajduje się on w pliku o nazwie *comments.php*. Jeśli nie ma tego pliku w motywie, WordPress dołączy domyślny plik *wp-includes/tcheme-compat/comments.php*. Wystarczy w miejscu, w którym ma być dostępna funkcja komentarzy, wstawić tag `comments_template()`:

```
<?php comments_template(); ?>
```

W razie potrzeby można też załadować alternatywny plik komentarzy przy użyciu parametru do tagu `comments_template()`. Można także rozdzielić komentarze według typu. Wartość

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

`true` oznacza, że zwykłe komentarze mają być oddzielone od pingbacków. Więcej informacji na temat tagu `comments_template()` znajduje się w dokumentacji pod adresem *http://codex.wordpress.org/Function_Reference/comments_template*.

## PRZEKAZYWANIE KILKU PARAMETRÓW DO TAGÓW SZABLONOWYCH

Pobieranie treści za pomocą tagów szablonowych jest bardzo łatwe. Niektóre z nich nie przyjmują żadnych parametrów, inne, jak np. `bloginfo()`, przyjmują jeden parametr, ale są też i takie, które przyjmują kilka parametrów.

Dwa bardzo przydatne tagi to `edit_post_link()` i `edit_comment_link()`. Zasadniczo działają tak samo, tzn. dodają łącza do edycji wpisów i komentarzy, dzięki czemu zalogowany użytkownik o odpowiednich uprawnieniach może szybko dokonać zmian. Kliknięcie takiego łącza powoduje przejście do panelu administracyjnego, gdzie można wprowadzić poprawki we wpisie lub w komentarzu. Tagi te są szczególnie przydatne w sytuacjach, gdy wyłączony jest górny pasek administracyjny.

Tagi te należy umieszczać w plikach szablonowych obok wpisów i komentarzy. Oba te tagi działają tak samo i muszą znajdować się w pętli, o której będzie mowa dopiero w następnym rozdziale, a na razie wystarczy zapamiętać, że `edit_post_link()` powinien być w kodzie wyświetlającym treść wpisów, natomiast `edit_comment_link()` — w kodzie wyświetlającym komentarze.



Poniżej znajduje się przykład użycia tagu `edit_post_link()` z domyślnymi parametrami:

```
<?php edit_post_link(); ?>
```

Gdy umieści się ten kod w swoim motywie, to na stronach w jego miejscu zostanie wyświetlony odnośnik *Edytuj*. Przypuśćmy, że chcesz, aby łącze to wyświetlało się w osobnym akapicie, przed nim znajdowało się słowo *Admin*, a jego treść zamiast *Edytuj* brzmiała *Edytuj ten wpis*. W takim przypadku należy użyć następującego kodu:

```
<?php edit_post_link( 'Edytuj ten wpis', '<p>Admin: ', '</p>' ); ?>
```

Tag `edit_post_link()` przyjmuje cztery parametry. Pierwszy określa tekst łącza, np. `'Edytuj ten wpis'`, natomiast drugi — to, co ma się znaleźć przed tym łączem. W przytoczonym przykładzie chcieliśmy, aby łącze znajdowało się w osobnym akapicie i było poprzedzone słowem *Admin* — dlatego właśnie użyliśmy kodu `'<p>Admin: '` (zwróć uwagę na spację za dwukropkiem, która stanowi odstęp między nim a łączem). Trzeci parametr określa, co ma się znajdować za łączem — w tym przypadku jest to zamknięcie akapitu, czyli znacznik `</p>`. Czwarty parametr, który tu został opuszczony, to `$id`, czyli identyfikator wpisu.

Innymi słowy, tag `edit_post_link()` przyjmuje maksymalnie cztery parametry, co posługując się składnią PHP, można wyrazić następująco:

```
<?php edit_post_link( $link, $before, $after, $id ); ?>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Pamiętaj, że parametry zazwyczaj przekazuje się w cudzysłowach i oddziela się przecinkami oraz spacjami dla polepszenia czytelności.

Niezbyt skomplikowane, prawda? Trzeba tylko wiedzieć, jakich parametrów można używać i w jakiej kolejności powinny one być wpisane. Kolejność jest ważna: jeśli ją pomylisz, możesz otrzymać łącze z niewłaściwym tekstem i zeszpecić projekt, a przynajmniej spowodować, że kod strony będzie niepoprawny.

## ARGUMENTY W STYLU FUNKCJI I ŁAŃCUCHÓW ZAPYTAŃ

Teraz obejrzymy coś bardziej skomplikowanego:

```
<?php wp_tag_cloud(); ?>
```

Ten tag szablonowy wyświetla chmurę tagów zawierającą maksymalnie 45 elementów, z których najmniejszy ma rozmiar czcionki 8 punktów, a największy 22 punkty. Tagi zostaną wyświetlone w postaci płaskiej listy i posortowane według nazw w rosnącej kolejności. Wiadomo to, ponieważ takie są ustawienia domyślne, których jest całkiem sporo — patrz tabela 2.1. W istocie tag `wp_tag_cloud()` może przekazywać 14 argumentów do parametru.



Tabela 2.1. Domyślne parametry chmury tagów

| Parametr | Opis | Domyślna wartość |
|---|---|---|
| `smallest` | Rozmiar najmniejszego tagu | `8` |
| `largest` | Rozmiar największego tagu | `22` |
| `unit` | Jednostka do określania rozmiaru czcionki | `pt` |
| `number` | Maksymalna liczba wyświetlanych tagów | `45` |
| `format` | Sposób wyświetlania tagów | `flat` |
| `separator` | Znak oddzielający tagi | spacja |
| `orderby` | Sposób uporządkowania tagów | `name` |
| `order` | Sposób sortowania tagów | `ASC` |
| `exclude` | Tagi, które należy wykluczyć | brak |
| `include` | Tagi, które należy dodać | wszystkie |
| `link` | Czy mają być tworzone odnośniki do edycji, czy do widoku | `view` |
| `taxonomy` | Baza chmury tagów | `post_tag` |
| `topic_count_text_callback` | Służy do tworzenia własnych dymków dla łączy | `default_topic_count_text` |
| `echo` | Czy chmura ma zostać wyświetlona, czy tylko zwrócona | `true` |

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jeśli porównasz dane z tej tabeli z opisem domyślnego wyglądu chmury tagów tworzonej przez tag `wp_tag_cloud()`, spostrzeżesz, że parametry te są przekazywane automatycznie i nie trzeba ich wpisywać.

Teraz zmienimy działanie tego tagu, modyfikując wartości jego parametrów. Argumenty można przekazywać na dwa sposoby: funkcyjnie lub przy użyciu łańcuchów zapytania. Jeśli używasz WordPressa od dłuższego czasu, to z pewnością spotkałeś się już z jednym i drugim sposobem, ale aktualnie preferowany jest sposób funkcyjny.

Zaczniemy od przekazania jednego argumentu przy użyciu metody łańcuchowej:

```
<?php wp_tag_cloud( 'unit=px' ); ?>
```

Jest to prosta technika: w tym przypadku ustawiliśmy jednostkę (`unit`) na piksele (`px`). Opis parametru `unit` znajduje się w tabeli 2.1 oraz w dokumentacji WordPressa (*http://codex.wordpress.org/Function_Reference/wp_tag_cloud*). Gdy przekazuje się tylko jeden lub kilka argumentów, to metoda łańcuchowa wydaje się odpowiednia. Każdy kolejny parametr należy oddzielić od poprzedniego znakiem `&` (bez żadnych odstępów). Poniżej zmieniamy uporządkowanie na według liczby (`count`) zamiast według nazwy (`name`):

```
<?php wp_tag_cloud( 'unit=px&orderby=count' ); ?>
```



Proste, prawda? Teraz przełączymy się na preferowaną metodę funkcyjną i muszę Cię poprosić o odrobinę cierpliwości, ponieważ korzyści płynące z jej zastosowania staną się widoczne dopiero za jakiś czas. Najpierw przepiszemy poprzedni przykład przy użyciu nowej składni. Aby to zrobić, argumenty musimy zapisać w tablicy i dopiero tę tablicę przekazać do tagu `wp_tag_cloud()`:

```
<?php
  // Argumenty
  $args = array(
      'unit' => 'px',
       'orderby' => 'count'
  );
  // Przekazanie argumentów
  wp_tag_cloud( $args );
?>
```

Użyliśmy tych samych argumentów co poprzednio, tylko zapisaliśmy je w tablicy `$args`, którą przekazaliśmy jako argument do tagu `wp_tag_cloud()`. Proste.

W razie potrzeby można dodać kolejne parametry, oddzielając je przecinkami. Możemy też zmienić ich kolejność oraz ustawić najmniejszy rozmiar czcionki na `10px` (ponieważ domyślne 8 jest dość małe, jeśli ktoś używa pikseli, a nie punktów), a największy — na `24px`:

```
<?php
  // Argumenty
  $args = array(
       'smallest' => 10,
       'largest' => 24,
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
        'unit' => 'px',
        'orderby' => 'count',
        'order' => RAND
    );
    // Przekazanie argumentów
    wp_tag_cloud( $args );
?>
```

Kod ten jest prosty i przejrzysty. Porównaj go z odpowiednikiem w stylu łańcuchowym:

```
<?php
    wp_tag_cloud( 'smallest=10&largest=24unit=px&orderby=count&order=RAND' );
?>
```

Tu już trudno mówić o przejrzystości kodu. Morał z tego taki, że warto wiedzieć o istnieniu czegoś takiego jak łańcuch zapytania, ale gdy trzeba przekazać wiele argumentów do tagu, lepiej użyć stylu funkcyjnego. Składnia ta jest prostsza i łatwiejsza w użyciu, a ponadto jest preferowaną metodę przekazywania parametrów.

Jeszcze krótki komentarz na temat ostatniego przykładu. Wartość określająca sposób sortowania `RAND` jest napisana wielkimi literami. Tak zwyczajowo zapisuje się wartość argumentu sortowania (dwie pozostałe wartości to `ASC` i `DESC`). Zapewne zauważyłeś też, że argumenty `smallest` i `largest` zostały umieszczone przed argumentem jednostki. Dobrym zwyczajem jest wpisywanie argumentów w takiej samej kolejności, w jakiej są opisane, dzięki czemu łatwiej jest je znaleźć, gdy trzeba coś zmienić w kodzie.



## TYPY DANYCH

Do tagów szablonowych można przekazywać trzy typy danych: łańcuchy, liczby całkowite oraz wartości logiczne. Wprawdzie w opisie każdego tagu w dokumentacji napisane jest, jak należy przekazywać wartości, ale mimo to warto dokładnie wiedzieć, o co chodzi, aby móc zawsze wybrać najwłaściwszą wartość.

Pierwszy typ danych to `string`, czyli po prostu ciąg znaków. Łańcuchem jest np. `bloginfo( 'name' )`, ponieważ parametrem jest tu `'name'`. Łańcuchy można wpisywać w podwójnym lub pojedynczym cudzysłowie (nie ma różnicy), chociaż znacznie częściej spotyka się cudzysłów pojedynczy i jest on używany także w tej książce.

Drugi typ danych to **liczby całkowite**, czyli takie, które nie mają części ułamkowej, np. `55900` albo `-3`. Można je przekazywać w cudzysłowach, ale nie jest to konieczne. Liczb całkowitych najczęściej używa się do określania identyfikatorów, których może być bardzo dużo. W dalszej części książki napotkasz zarówno tagi szablonowe, jak i warunkowe, w których używane są tego typu liczby.

W końcu trzeci typ danych to wartości logiczne, czyli prawda (`true`) lub fałsz (`false`). Wartości prawdy i fałszu można zapisywać jako słowa kluczowe wielkimi (`TRUE` i `FALSE`) lub małymi (`true` i `false`) literami oraz w postaci liczbowej (`1` oznacza prawdę, a `0` — fałsz). Wartości logicznych nie można zapisywać w cudzysłowach. Na przykład tag `get_calendar()` przyjmuje tylko jeden parametr określający, czy ma wyświetlać jednoliterowy skrót nazwy

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

dnia wielką literą, czy kilkuliterowy skrót małymi literami. Domyślną wartością jest `true`, co oznacza, że wyświetlany jest skrót jednoliterowy (np. P dla poniedziałku). Jeśli zatem chcesz wyświetlić słowo *pon* zamiast *P*, musisz tagowi `get_calendar()` przekazać wartość `false`:

```
<?php get_calendar( FALSE ); ?>
```

Innym przykładem instrukcji logicznej jest tag `the_date()` służący do wyświetlania daty publikacji wpisu. Można na przykład chcieć tę informację wykorzystać tylko do jakichś celów w swoim skrypcie i nie wyświetlać jej na stronie. Można zmienić format daty (pierwszy łańcuch na liście parametrów), to, co ma się znajdować przed wyświetloną datą (drugi łańcuch), oraz to, co ma się znajdować za nią (trzeci łańcuch). Czwarta wartość, jaką można przekazać, to wartość logiczna określająca, czy dane mają zostać wysłane na wyjście (domyślnie `true`). Gdybyśmy zatem chcieli wyświetlić datę w formacie dzień.miesiąc.rok (`d.m.Y` w składni PHP, którą WordPress w pełni obsługuje), napisalibyśmy następujący kod:

```
<?php the_date( 'd.m.Y', '<p>', '</p>' ); ?>
```

Gdybyś jednak chciał użyć tego tagu gdzieś w skrypcie PHP poza funkcjami WordPressa i anulować wysyłanie danych na wyjście, mógłbyś ustawić opcję `echo` tego tagu na `false`. Jest to wartość logiczna, którą należy umieścić na końcu listy parametrów i której nie można umieścić w cudzysłowie:

```
<?php the_date( 'd.m.Y', '<p>', '</p>', FALSE ); ?>
```



Wynik wykonania tego kodu będzie taki sam jak poprzednio, tzn. będzie nim data w elemencie `p`, ale nie zostanie on nigdzie wyświetlony, a więc nadaje się on do użycia w skrypcie PHP. Innymi słowy, powyższy kod służy do przypisywania wyniku tagu `the_date()` do zmiennej, a nie jego wyświetlania bezpośrednio na stronie.

Przypomnę, że łańcuchy to po prostu ciągi tekstu w cudzysłowach, liczby całkowite to po prostu liczby bez części ułamkowej, a wartości logiczne to wartości oznaczające prawdę i fałsz, których nie można umieszczać w cudzysłowach. Dzięki tej wiedzy będzie Ci znacznie łatwiej zrozumieć działanie tagów szablonowych, których będziesz używać do tworzenia swoich niesamowitych witryn internetowych.

---

*Nie zapisuj formatu daty w tagach `the_date()` i `the_time()`, chyba że dobrze wiesz, co robisz. Te ustawienia są dostępne w panelu administracyjnym WordPressa, a więc w większości przypadków nie ma potrzeby ich modyfikowania.*

---

# TAGI WARUNKOWE

Tagi warunkowe to niezwykle przydatna rzecz. Służą do tworzenia instrukcji warunkowych w plikach motywów. Dzięki nim można wyświetlać różne informacje w zależności od sytuacji. Dobrym przykładem takiego tagu jest `is_front_page()`, który sprawdza, czy aktualnie otwarta strona to strona główna. Przy jego użyciu można wyświetlić powitanie, bo tak nakazują kultura i dobre wychowanie:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php
  if ( is_front_page() ) {
    echo '<p class="welcome">Hej Ty, witaj na mojej stronie. Uwielbiam nowych
    ↪gości! </p>';
  }
?>
```

Powyższy kod spowoduje wyświetlenie akapitu przypisanego do klasy `welcome` i zawierającego tekst, ale tylko wtedy, gdy użytkownik znajduje się na stronie głównej. To był bardzo prosty przykład, więc przeanalizujemy jeszcze jeden.

Przypuśćmy, że jedna z kategorii w naszej witrynie powinna mieć inny pasek boczny niż wszystkie pozostałe. Pomysł ten możemy zrealizować, korzystając z tagu `is_category()`. Gdy wyświetlana jest strona tej wybranej kategorii, dołączamy specjalny pasek boczny, na wszystkich pozostałych stronach wyświetlamy normalny pasek za pomocą tagu `get_sidebar()`.

Poniższy kod zastąpi tag `get_sidebar()` odpowiednim kodem na wszystkich stronach, na których ma to znaczenie, a więc np. w plikach *index.php*, *category.php* czy *single.php*:

```
<?php
  if (is_category( 'very-special' )) {
    get_sidebar( 'special' );
  } else {
    get_sidebar();
  }
?>
```



Najpierw sprawdzamy, czy wyświetlana kategoria ma uproszczoną nazwę `very-special` (uproszczona nazwa — inaczej slug — to nazwa używana w łączach itp.). Równie dobrze można by było użyć identyfikatora albo nazwy kategorii i chociaż identyfikator jest chyba najbardziej niezawodny, to jednak nazwy w postaci **rozpoznawalnych** słów (tzn. bez żadnych specjalnych znaków itp.) łatwiej się czyta. Jeśli kategoria ma nazwę `very-special`, zostanie użyty tag `get_sidebar('special')`, w którym parametr `special` oznacza, że wczytany ma zostać plik *sidebar-special.php*.

Jeśli kategoria ma inną nazwę, nastąpi przejście do klauzuli `else` i wykonanie tagu `get_sidebar()`, który dołącza standardowy pasek boczny *sidebar.php*.

Jak widać, zasada działania tagów szablonowych jest bardzo prosta. Niedługo będziesz przy ich użyciu robić różne ciekawe rzeczy.

## CO W NICH TAKIEGO WYJĄTKOWEGO

Tagi warunkowe są podstawowym mechanizmem umożliwiającym kontrolowanie motywu. Mając możliwość spytania WordPressa, w którym miejscu witryny i w jakich warunkach znajduje się użytkownik, można odpowiednio reagować na różne sytuacje. Dołączanie specjalnego paska bocznego to tylko prosty przykład, ale dzięki tagom warunkowym można zrobić o wiele więcej. Dzięki możliwości wczytywania różnych części plików szablonowych

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

w zależności od tego, w którym miejscu witryny znajduje się użytkownik, można zredukować liczbę szablonów albo sprawić, że motyw będzie nieco bardziej dynamiczny. W połączeniu z prostą instrukcją `if` tagi warunkowe stwarzają jeszcze większe możliwości.

Początkujący webmasterzy często zapominają, że wiedząc, gdzie użytkownik się znajduje, wiedzą też, gdzie go nie ma. Jeśli chcesz wykonać jakąś czynność dla wszystkich określonych sytuacji oprócz jednej, możesz użyć konstrukcji „jeśli nie", którą wyraża się za pomocą instrukcji `if` i wykrzyknika:

```
<?php
  if ( ! is_front_page() ) {
    // Robimy coś na wszystkich stronach oprócz głównej
  }
?>
```

Ten kod pozwala wykonać dowolne działania, gdy wynik tagu `is_front_page()` jest negatywny, czyli na wszystkich stronach oprócz głównej.

# W NASTĘPNYM ROZDZIALE: PĘTLA

Wiesz już, że witryny w WordPressie są oparte na motywach, które są zbudowane z szablonów zawierających różne tagi i wykonujących różne ciekawe działania. Czas zatem dowiedzieć się, jak wyświetlać i prezentować treść. Do tego służy specjalna konstrukcja zwana **pętlą**. Jest to fragment kodu PHP stanowiący serce WordPressa. Choć wiele rzeczy można zrobić, nawet nie wiedząc o pętli, to jednak jeśli chcesz zmusić system do całkowitego posłuszeństwa, ta wiedza będzie Ci niezbędna. Za pomocą pętli można zmienić kolejność wyświetlania wpisów i ogólnie sterować sposobem prezentacji całej treści.



Wiele rzeczy, jakie można zrobić w WordPressie, da się wykonać tylko w pętli. A to oznacza, że czasami trzeba utworzyć kilka pętli, każdą do czego innego. Albo przynajmniej dobrze jest wiedzieć, gdzie dana pętla się zaczyna i kończy, aby dodawać do niej tagi i funkcje wtyczek, których bez wątpienia wiele znajdziesz i zechcesz użyć.

Nie ma mowy, żeby to jakoś obejść. Pętla jest ważna i im wcześniej się z nią zapoznasz, tym lepiej.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 3 PĘTLA

**MASZ JUŻ ZAINSTALOWANEGO** WordPressa i wiesz ogólnie, jak działają motywy. Możesz w takim razie zapoznać się z samym sercem tego systemu, czyli **pętlą**. Pętla to nieskomplikowane wyrażenie w języku PHP, które służy do kontaktowania się z WordPressem w celu pobrania od niego treści. Na początku tego rozdziału przedstawiona jest prosta pojedyncza pętla. Później przejdziemy do bardziej skomplikowanych struktur złożonych z kilku pętli oraz pokażę Ci kilka ciekawych sztuczek, za pomocą których można uzyskać różne efekty.

Jeśli chce się tworzyć naprawdę dobre witryny internetowe w WordPressie, znajomość zasad działania pętli jest niezbędna. Nie musisz być ekspertem, ale podstawowa wiedza w tym zakresie bardzo pomaga. Dzięki temu będziesz wiedział, jak przeszukiwać dostępne funkcje, gdy zechcesz zastosować rozwiązanie wymagające niestandardowego podejścia.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# ZASADA DZIAŁANIA PĘTLI WORDPRESSA

Pętla to serce systemu, które bije w plikach szablonowych każdego motywu. Wprawdzie można utworzyć motyw, nie używając pętli, ale wówczas obsługa treści, jak np. wyświetlanie najnowszych wpisów czy implementacja nawigacji po archiwach, byłaby bardzo utrudniona. Niektóre pliki szablonowe, jak np. plik błędu 404, nie mają pętli, ale większość ją zawiera.

Ponadto niektóre tagi szablonowe działają tylko wewnątrz pętli, a więc musisz wiedzieć, jak je znaleźć. Jak się wkrótce przekonasz, nie jest to nic trudnego.

## NAJPROSTSZA PĘTLA

Jeśli chcesz tworzyć witryny internetowe oparte na WordPressie, musisz zrozumieć, jak działa pętla. Na szczęście struktura ta w najprostszej postaci jest bardzo łatwa do opanowania. Jej początek wygląda tak:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
```

Natomiast koniec pętli wyznacza poniższy kod:

```
<?php endwhile; else: ?>
    <p>Jakiś komunikat o błędzie lub coś w tym rodzaju.</p>
<?php endif; ?>
```



Powiadomienie o błędzie potrzebne jest tylko w tych plikach szablonowych, które służą do wyświetlania komunikatów o błędach. Ponieważ jednak w wielu motywach można to spotkać, pokazałem to też i tutaj. Powiadomieniem o błędzie może być napis *Wpis nie został znaleziony* albo wynik wyszukiwania z informacją, że dla określonego zapytania nie ma żadnych wyników. Zasadniczo początek pętli wyznacza instrukcja `while`, natomiast koniec — `endwhile`, ale w wielu motywach dodatkowo używane są instrukcje `if`, za pomocą których dodaje się powiadomienia o błędach itp. Instrukcję taką widzimy też w tym przykładzie.

Wewnątrz pętli znajduje się właściwy kod zawierający tagi szablonowe służące do wysyłania na wyjście treści wpisów. Poniższy kod lub coś bardzo podobnego można znaleźć w pliku *index.php* lub *loop.php* w większości motywów:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
  <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
    <h2><a href="<?php the_permalink(); ?>" title="<?php
      the_title_attribute(); ?>">
        <?php the_title(); ?>
      </a></h2>
      <?php the_content(); ?>
  </div>
<?php endwhile; else: ?>
  <div class="post">
    <h2>Błąd!</h2>
    <p>Coś się nie udało! Spróbuj jeszcze raz.</p>
  </div>
<?php endif; ?>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W prawdziwym motywie należałoby wpisać jakieś bardziej treściwe powiadomienie o błędzie, ale nie to jest dla nas teraz ważne.

Podstawowa pętla sprawdza, czy są jakieś wpisy, a o jej działaniu decydują globalne ustawienia witryny (ile wpisów ma zostać wyświetlonych i w jaki sposób mają być zaprezentowane) i to, w którym miejscu witryny znajduje się użytkownik. Na stronie pojedynczego wpisu powinien oczywiście zostać wyświetlony tylko jeden wpis (najlepiej zażądany przez użytkownika), natomiast na stronie kategorii powinna zostać wyświetlona określona w ustawieniach liczba wpisów należących do wybranej kategorii.

Jeśli są jakieś wpisy, następuje uruchomienie pętli `while`, która pobiera i wysyła na wyjście kolejne wpisy zgodnie z ustawieniami i warunkami. Po zakończeniu jej działania (czyli gdy wszystkie odpowiednie wpisy zostaną już wyświetlone) następuje przejście do instrukcji `endwhile`, a później do `endif`.

Jeśli nie ma żadnych wpisów spełniających określone kryteria, następuje wykonanie bloku `else`, w którym zostaje wyświetlone powiadomienie o błędzie lub inny komunikat (albo nic, jeśli programista nic nie zdefiniował). Po tym pętla definitywnie kończy działanie.

Podsumowując: pętla pobiera dane z bazy danych zgodnie z ustawieniami WordPressa i innymi kryteriami wyboru określonymi przez stronę, która została otwarta. Logiczne, prawda?



## ZAPISYWANIE PĘTLI W PLIKU SZABLONOWYM LOOP.PHP

Kiedyś pętla była wpisywana w pliku *index.php* i ewentualnie kilku innych plikach szablonowych. Obecnie jednak często stosuje się inne podejście. Pętlę dołącza się do różnych plików szablonowych z osobnego pliku przy użyciu specjalnego tagu o nazwie `get_template_part()` (o którym była mowa w rozdziale 2.). Dzięki temu ten sam kod pętli można wykorzystać wielokrotnie w różnych miejscach. W istocie niektóre pliki szablonowe mogą zawierać własną pętlę i nie korzystać z tagu `get_template_part()`, a inne mogą ją dołączać z pliku *loop.php*, ale prezentować treść na różne sposoby w zależności od miejsca w witrynie. Na przykład archiwum kategorii zazwyczaj wygląda inaczej niż strona pojedynczego wpisu, ale jedno i drugie można wyświetlić przy użyciu tej samej pętli (aczkolwiek nie przy użyciu tego samego kodu wewnątrz pętli w pliku *loop.php*) lub dwóch całkiem różnych pętli. Wszystko zależy od miejsca w witrynie i liczby szablonów znajdujących się w motywie.

Przypomnę z rozdziału 2., że tag `get_template_part()` dołącza pętlę z pliku *loop.php*, w którym oczywiście w takim razie powinniśmy ją zapisać.

W związku z powyższym przykładowy kod pętli przedstawiony wcześniej nie zostałby umieszczony w pliku *index.php* i w zamian mielibyśmy coś takiego:

```
<?php get_template_part( 'loop', 'index' ); ?>
```

Ten kod spowoduje, że system najpierw poszuka pliku *loop-index.php* i jeśli go nie znajdzie, użyje domyślnie pliku *loop.php*. Kod pętli można umieścić w jednym z tych dwóch plików. Pliku *loop-index.php* można by było użyć do zapisania pętli dla konkretnej strony (w tym

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

przypadku *index.php*), pozostawiając pętlę ogólną (w pliku *loop.php*) bez zmian. Gdybyś zatem chciał np. utworzyć osobną pętlę dla archiwów kategorii, mógłbyś w pliku *category.php* użyć następującego kodu:

```
<?php get_template_part( 'loop', 'category' ); ?>
```

Ten kod będzie szukał pliku o nazwie *loop-category.php* i w tym właśnie pliku należy zapisać odpowiednią pętlę. Wiadomości te tak naprawdę wykorzystasz dopiero w rozdziale 5., w którym będzie mowa o tworzeniu motywów potomnych, a na razie wystarczy zapamiętać, że istnieje taki dobry sposób na oddzielanie od szablonów kodu pętli, który może być dość długi.

---

*Nie ma żadnej reguły nakazującej używanie pliku* loop.php *ani w ogóle oddzielanie pętli od szablonów w ten sposób. Za pomocą tagu* `get_template_part()` *można dołączać do plików, co się chce. Poza tym niektórzy wolą mieć podstawowe części pętli w plikach szablonowych i dołączać tylko te fragmenty, które decydują o różnicach w wyświetlaniu treści.*

---

## KILKA SŁÓW O WP_QUERY



WP_Query to najważniejsza część, mimo że nie widać jej w najprostszej postaci tej konstrukcji. Jest to klasa odpowiadająca za sposób działania pętli i jej definicję można znaleźć w pliku *wp-includes/query.php*. Objaśnienie wszystkich właściwości i metod tej klasy znajduje się w dokumentacji WordPressa pod adresem *http://codex.wordpress.org/Function_Reference/WP_Query*.

Podstawowa pętla korzysta z domyślnego obiektu klasy `WP_Query` o nazwie `$wp_query`. Oznacza to, że pisząc `have_posts()` i `the_post()`, w istocie używasz instrukcji `$wp_query->have_posts()` i `$wp_query->the_post()`. Innymi słowy, instrukcje typu `have_posts()` domyślnie odnoszą się do obiektu `$wp_query`. Aby móc korzystać z klasy `WP_Query` w inny sposób, o czym będzie mowa w części poświęconej tworzeniu wielu pętli, jak również w innych dalszych częściach tej książki, należy najpierw utworzyć sobie nowy obiekt tej klasy w następujący sposób:

```
<?php $brand_new_query = new WP_Query(); ?>
```

Teraz wszystkie zapytania będą wysyłane do obiektu `$brand_new_query` zamiast do `$wp_query`, dzięki czemu różne rzeczy będzie można robić poza główną pętlą.

Klasa `WP_Query` jest wykorzystywana w wielu miejscach witryny, np. w plikach szablonowych i tagach warunkowych. Można się do niej podłączyć i ją modyfikować, ale lepiej tego unikać, jeśli się da. A gdy całkowicie opanujesz obsługę WordPressa, możesz zechcieć robić kompletnie niestandardowe rzeczy, ale to już inna historia.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# UŻYWANIE PĘTLI

Wiesz już, do czego służy pętla (do wyświetlania treści) i że możesz używać kilku pętli naraz. Teraz w takim razie powinieneś dowiedzieć się, jak używać jej efektywnie. Czasami na przykład nie chcemy, aby na liście kategorii były wyświetlane całe wpisy, a tylko ich wycinki, w interfejsie WordPressa nazywane wypisami. Do tego potrzebne są tagi szablonowe i tagi służące do kontrolowania działania pętli.

W tym podrozdziale znajduje się opis najczęściej używanych tagów. Możliwe, że kiedyś zechcesz wyświetlić tytuły różnych wpisów i sprawić, aby były odnośnikami do pełnej treści tych wpisów. Poniższy krótki fragment kodu znajduje się praktycznie w każdym pliku szablonowym wyświetlającym jakąś listę wpisów, a czasami także w szablonach wyświetlających tylko pojedyncze wpisy:

```
<h2>
  <a href="<?php the_permalink(); ?>" title="<?php
    the_title_attribute( array(
      'before' => 'Łącze do: ',
      'after' => '')
    );
  ?>">
    <?php the_title(); ?>
  </a>
</h2>
```



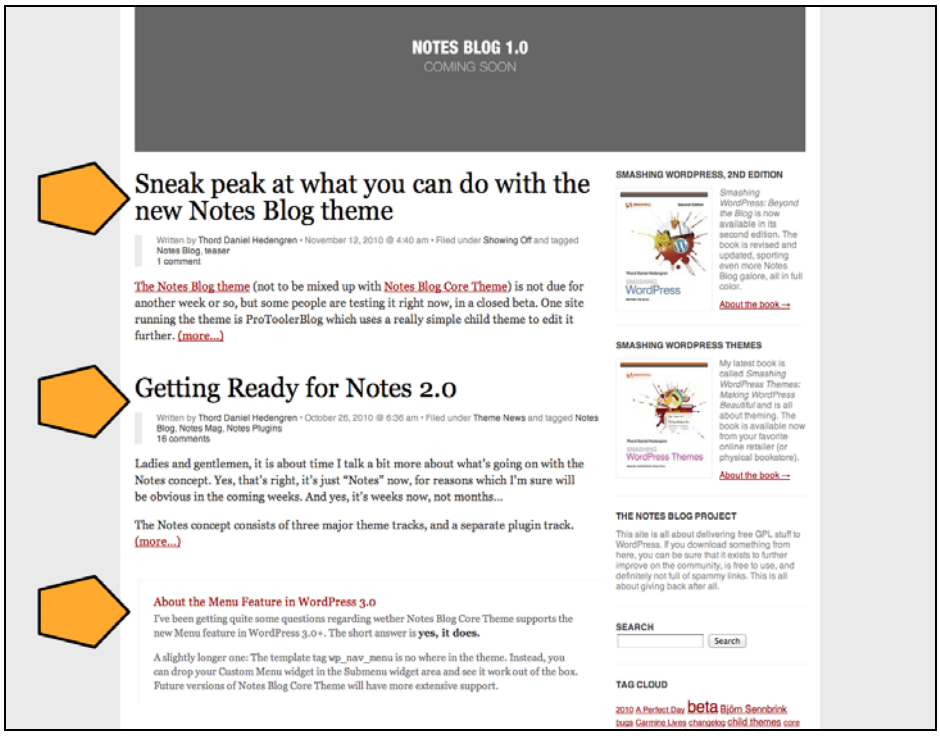
Jest to bardzo prosty kod: za pomocą specjalnego tagu wyświetlany jest tytuł wpisu. Tytuł ten został umieszczony w zwykłym odnośniku, którego wartość atrybutu `href` jest ustawiana za pomocą kodu `<?php the_permalink(); ?>` zwracającego stałe łącze do wpisu. Tytuł dla hiperłącza można pobrać za pomocą tagu `the_title_attribute()`, który zwraca tytuł wpisu. W tym przypadku tagowi temu została przekazana tablica ustawień. Przed tytułem wstawiany jest tekst *Łącze do*, a za nim — nic. Tagu `the_title_attribute()` można także użyć bez żadnych ustawień i wówczas zwróci on po prostu dostosowaną wersję tytułu wpisu. Jeśli potrzebujesz niezmodyfikowanego tytułu, możesz użyć tagu `the_title()`.

Do wyświetlania treści można użyć jednego z poniższych tagów:

```
<?php the_content() ;?>
<?php the_excerpt() ;?>
```

Pierwszy z tych tagów wyświetla całą treść wpisu, chyba że znajdujemy się na stronie zawierającej listę kilku wpisów. Wówczas wyświetlana jest tylko treść do miejsca, w którym znajduje się tag `<!-- more -->`. Jeśli nie ma tego tagu, wyświetlany jest wpis w całości (rysunek 3.1). Użytkownicy WordPressa dobrze znają ten tag. W interfejsie użytkownika edytora wpisów występuje on jako przycisk *Więcej*, a po wstawieniu do wpisu powoduje wyświetlenie łącza *Czytaj dalej*.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 3.1. Wszystkie te wpisy zostały wyświetlone za pomocą pętli



Oczywiście treść tego odnośnika można zmienić, jak pokazano poniżej:

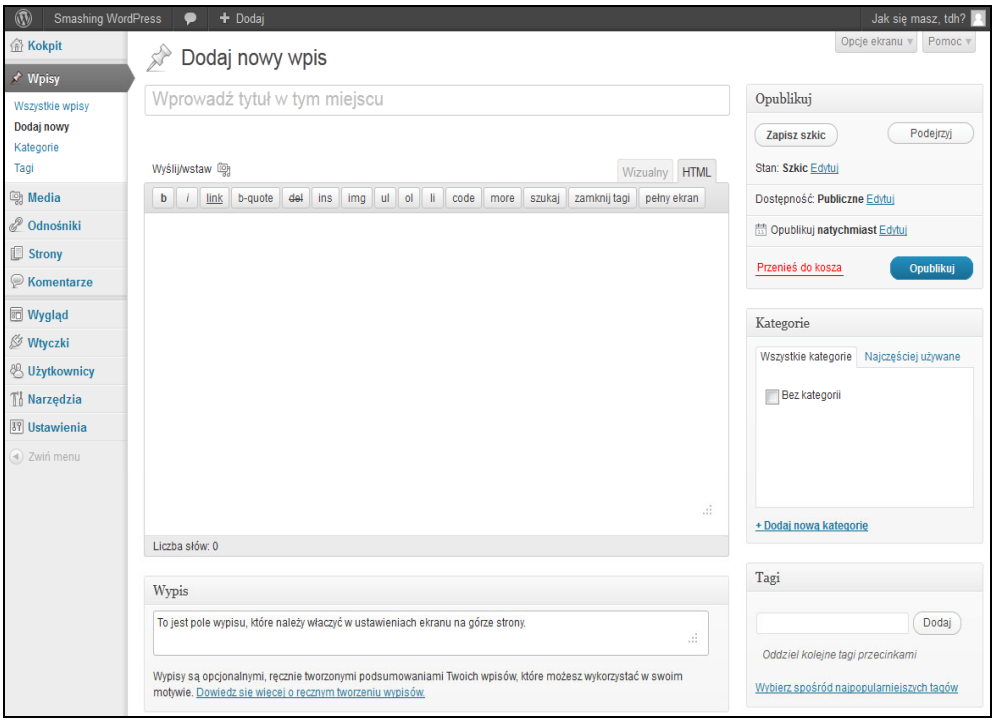
```
<?php the_content( 'Nie krępuj się, przeczytaj resztę!' ); ?>
```

W tym przypadku domyślna treść odnośnika zostałaby zmieniona na *Nie krępuj się, przeczytaj resztę!*. Jeśli chcesz, możesz też umieścić tu kod HTML, a nawet grafikę. Najważniejsze jest to, że tag szablonowy `the_content()` zwraca treść wpisu, a jeśli jest użyty w liście wpisów, wyświetla treść tylko do tagu `<!--more-->`. Oczywiście na stronach pojedynczych wpisów tag `the_content()` nie wyświetla łącza *Czytaj dalej*, tylko całe wpisy.

Drugi z wymienionych tagów to `the_excerpt()`. Tag ten wyświetla tylko początkowy fragment wpisu o maksymalnej długości 55 słów. Z fragmentu tego usuwane są wszystkie znaczniki HTML, a więc należy tagu tego używać ostrożnie, gdyż za jego pomocą nie wyświetlimy akapitów, obrazów czy filmów z YouTube. Co więcej, tag ten nie ma żadnych parametrów. Wydaje się, że powinna istnieć możliwość ustawiania długości wyświetlanego fragmentu, ale niestety. Aby to zrobić, trzeba dodać filtr do haka `excerpt_length`.

Mimo to tag `the_excerpt()` ma pewne zastosowania. Zauważyłeś, że na stronie edycji każdego wpisu znajduje się pole o nazwie *Wypis* (rysunek 3.2 — czasami trzeba włączyć wyświetlanie tego pola w opcjach ekranu u góry tej samej strony). Jeśli wpiszesz coś w tym polu, tag `the_excerpt()` wyświetli to zamiast wspominanych 55 słów. Jeśli zatem w polu *Wypis* wpiszesz *Cześć, to jest mój wpis*, to WordPress wyświetli dokładnie te słowa zamiast 55-wyrazowego skrótu, mimo że jest ich o wiele mniej niż 55.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 3.2. Pole Wypis pod edytorem wpisu w WordPressie



Jaki w takim razie pożytek można mieć z tagu `the_excerpt()`? Może on być przydatny do wyświetlania list wpisów, na których chcemy mieć precyzyjną kontrolę nad długością wyświetlanego skrótu, np. na stronach wyszukiwania albo w różnych archiwach. Jednak najbardziej oczywisty sposób jego wykorzystania to reklamowanie wpisów. Można utworzyć pętlę pobierającą wpisy mające przypisany określony tag i wyświetlającą ich tytuły w normalny sposób, ale zamiast treści zwracającą tylko wycinki („wypisy"). Pokażę, jak to zrobić, w następnym przykładzie.

Żeby wszystko było jasne, poniżej przedstawiam podstawowy sposób użycia tagu `the_excerpt()`:

```
<?php the_excerpt(); ?>
```

Pamiętaj, aby go nie używać w szablonach służących do wyświetlania pojedynczych wpisów, np. w pliku *single.php*. Aby wyświetlić całą treść, należy użyć tagu `the_content()`! I oczywiście jeśli chcesz, możesz używać obu tych tagów jednocześnie.

Poniżej opisuję, jak za pomocą tagu `the_excerpt()` utworzyć chwytliwe wstępy do wpisów w motywie Twenty Ten (ale oczywiście tylko w pojedynczych wpisach):

1. Otwórz folder motywu Twenty Ten (*wp-content/themes/twentyten/*) i znajdź w nim plik o nazwie *single.php*. Plik ten służy do wyświetlania pojedynczych wpisów, a więc należy mu się dokładnie przyjrzeć. Jak widać, w pliku tym znajduje się tag `get_template_part()` pobierający zawartość pliku *loop-single.php*:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php
/**
 * The Template for displaying all single posts.
 *
 * @package WordPress
 * @subpackage Twenty_Ten
 * @since Twenty Ten 1.0
 */

get_header(); ?>

        <div id="container">
            <div id="content" role="main">

            <?php
            /* Run the loop to output the post.
             * If you want to overload this in a child theme then include a file
             * called loop-single.php and that will be used instead.
             */
             get_template_part( 'loop', 'single' );
            ?>

            </div><!-- #content -->
        </div><!-- #container -->

<?php get_sidebar(); ?>
<?php get_footer(); ?>
```



1. Teraz możemy zapomnieć o pliku *single.php* i otwieramy plik *loop-single.php*. Nad tagiem `the_content()`, w okolicy wiersza 33., dodaj tag `the_excerpt()`. Tag `the_content()` aktualnie znajduje się w elemencie `div` przypisanym do klasy `entry-content`:

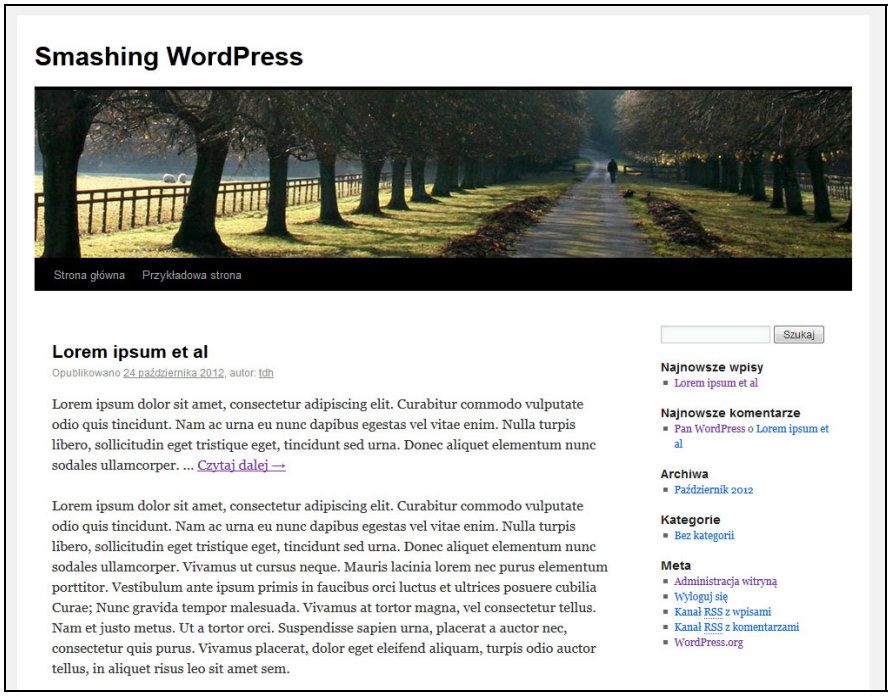
```
<div class="entry-content">
    <?php the_content(); ?>
    <?php wp_link_pages( array( 'before' => '<div class="page-link">'
    ↪. __( 'Pages:', 'twentyten' ), 'after' => '</div>' ) ); ?>
</div><!-- .entry-content -->
```

3. Bezpośrednio nad tagiem `the_content()` dodaj tag `the_excerpt()` w jego własnym elemencie `div`, aby można było się do niego odwoływać z poziomu arkusza stylów:

```
<div class="entry-content">
    <div class="entry-intro">
           <?php the_excerpt(); ?>
    </div>
    <?php the_content(); ?>
    <?php wp_link_pages( array( 'before' => '<div class="page-link">'
     ↪. __( 'Pages:', 'twentyten' ), 'after' => '</div>' ) ); ?>
</div><!-- .entry-content -->
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

4. Zapisz plik *loop-single.php* i wyślij go na serwer. Nad treścią każdego wpisu będzie teraz wyświetlony domyślny 55-wyrazowy skrót. Tylko że coś tu nie gra, na końcu skrótu widać odnośnik *Czytaj dalej*, którego nie powinno tu być (rysunek 3.3).




Rysunek 3.3. Na początku tego wpisu wyświetlony jest wycinek („wypis"). Zwróć uwagę na wielokropek na końcu fragmentu

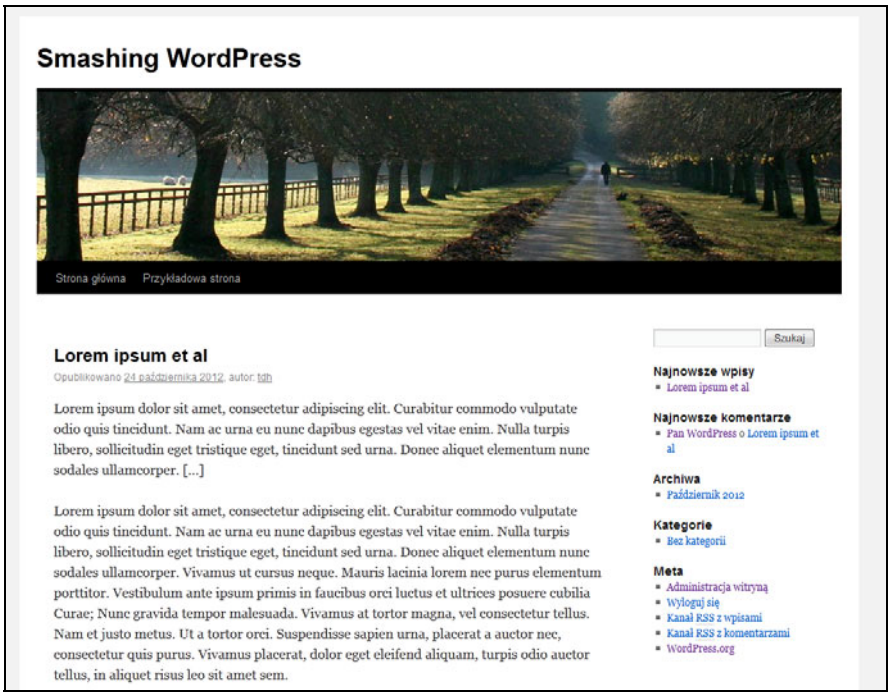
Za pojawienie się tego odnośnika odpowiada funkcja motywu Twenty Ten dodająca go do filtru `excerpt_more` (szczegółowy opis punktów zaczepienia i filtrów znajduje się w rozdziale 6.). Można go usunąć za pomocą dwóch prostych wierszy kodu, które należy umieścić przed tagiem `the_excerpt()`. Dwa wiersze kodu są potrzebne, ponieważ odnośnik, o którym mowa, może pojawiać się w dwóch sytuacjach — gdy zostanie wypełnione pole *Wypis* oraz gdy wycinek zostanie wygenerowany automatycznie. Dodaj poniższy kod do pliku *functions.php* motywu (oczywiście między znacznikami `<?php` i `?>`). Na rysunku 3.4 pokazano efekt tych modyfikacji:

```
// Usuwa filtry excerpt_more
remove_filter( 'excerpt_more', 'twentyten_auto_excerpt_more' );
remove_filter( 'get_the_excerpt', 'twentyten_custom_excerpt_more' );
```

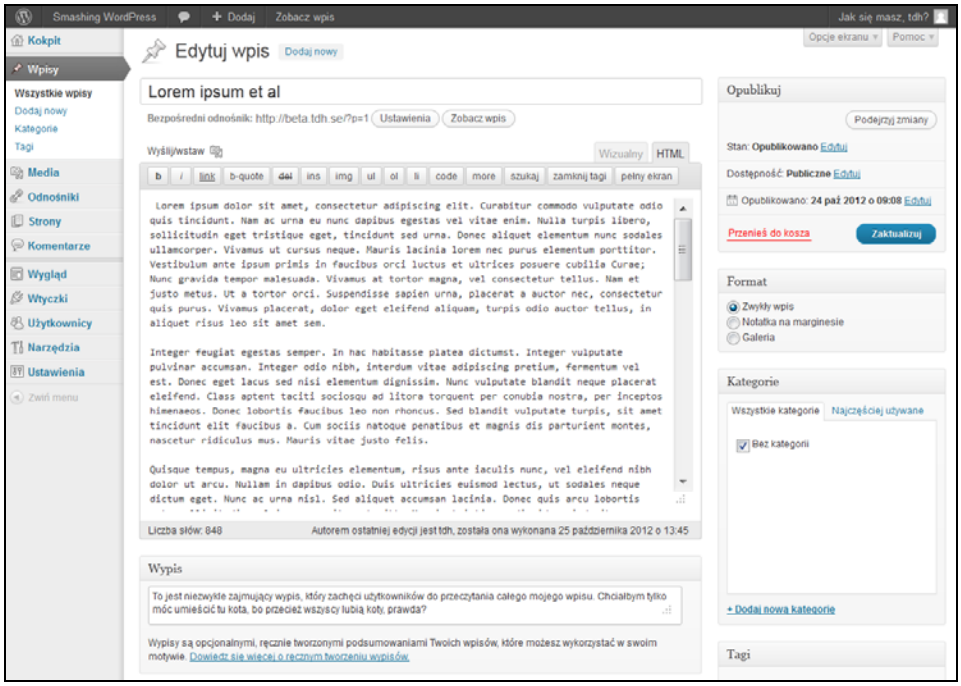
5. Jeśli na stronie edycji wpisu w panelu administracyjnym wpiszesz w przeznaczonym do tego polu (rysunek 3.5) jakiś tekst, zamiast domyślnego skrótu zostanie wyświetlony właśnie ten tekst (rysunek 3.6). Dzięki temu pozbędziesz się też tego paskudnego wielokropka *[…]*!



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974



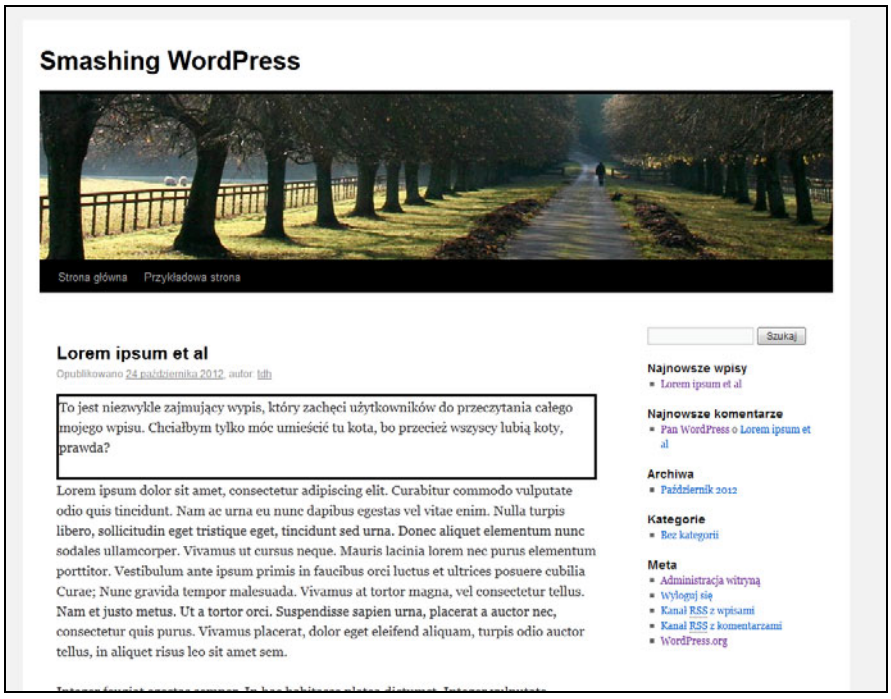

Rysunek 3.4. Odnośnik zniknął, a w jego miejsce pojawił się standardowy wielokropek w nawiasie




Rysunek 3.5. Edycja wpisu w panelu administracyjnym WordPressa i wypełnienie pola Wypis



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 3.6. Teraz na stronie wyświetlony jest tekst zdefiniowany przez użytkownika, a nie domyślny skrót



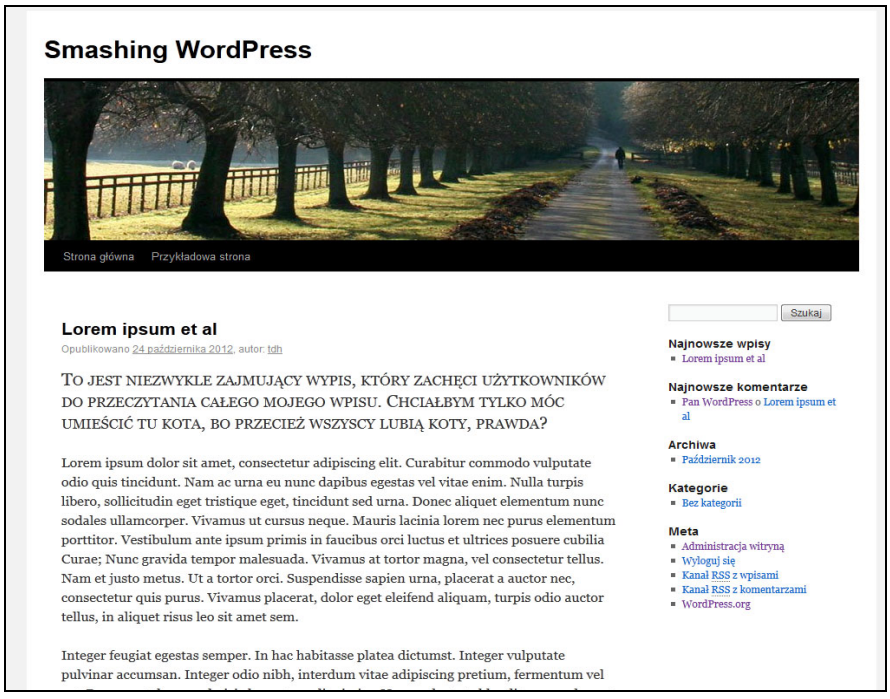
6. Prawie skończyliśmy. Teraz pozostało nam już tylko odpowiednio sformatować wygląd wycinka tekstu. W tym celu otwórz plik *style.css* i na jego końcu (albo w dowolnym innym miejscu, które uznasz za odpowiednie) dodaj kod pokazany poniżej. Arkusz ten spowoduje powiększenie tekstu w elemencie `.entry-intro`, ale u siebie możesz oczywiście zrobić coś całkiem innego. Efekt zastosowania tego kodu jest przedstawiony na rysunku 3.7.

```
div.entry-intro {
    font-size: 130%;
    line-height: 130%;
    font-variant: small-caps;
}
```

## PRZYKLEJANIE WPISÓW

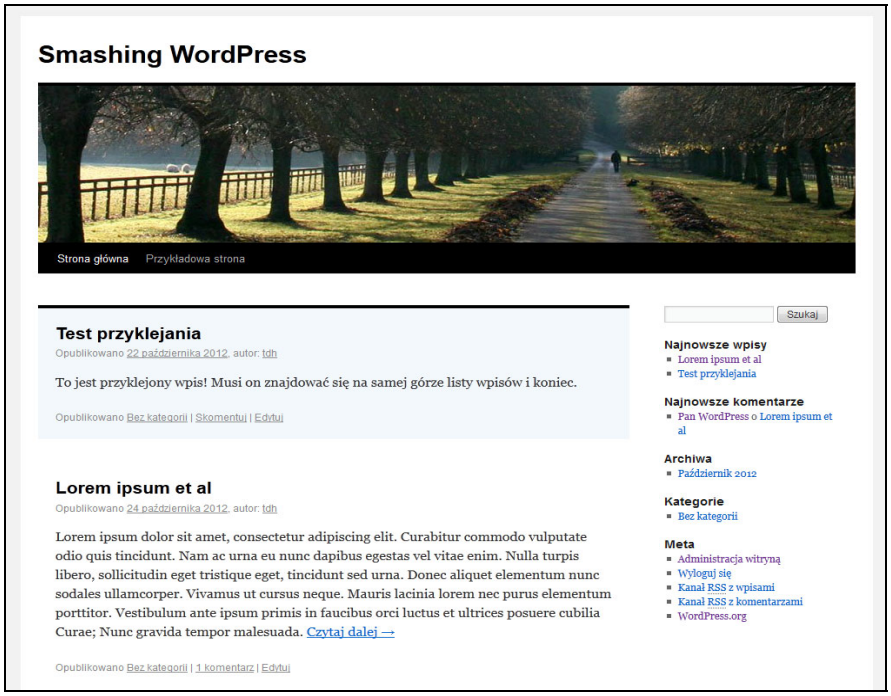
W WordPressie 2.7 dodano możliwość **przyklejania wpisów** na stronie głównej. Osoby, które śledziły wydarzenia na forach internetowych w tamtym czasie, od razu wiedzą, o co chodzi. Ogólnie rzecz biorąc, przyklejanie wpisów polega na umieszczaniu wpisów na początku listy wpisów i pozostawieniu ich tam niezależnie od dodawania kolejnych wpisów. Bez względu na to, ile nowych wpisów zostanie dodanych, przyklejony wpis będzie wyświetlony na pierwszym miejscu. Aby przykleić wpis, należy użyć odpowiedniej opcji w oknie edycji (rysunek 3.8). Jeśli jest kilka przyklejonych wpisów, są one wyświetlane w kolejności chronologicznej. Gdy usunie się opcję przyklejenia wpisu, zostanie on przeniesiony w miejsce zgodne z normalną kolejnością sortowania.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974






Rysunek 3.7. Wstęp do wpisu napisany kapitalikami




Rysunek 3.8. Przyklejony wpis

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Przyklejonym wpisom można przypisywać odpowiednią klasę za pomocą tagu `sticky_class()`, chociaż od kiedy wprowadzono tag `post_class()`, nie jest on już potrzebny:

```
<div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
```

Tag `post_class()` przypisuje wpisowi kilka klas w zależności od różnych ustawień. Ponieważ tag ten jest bardzo przydatny dla projektantów motywów, w rozdziale 6. zamieściłem jego szczegółowy opis.

Gdy wpis zostanie przyklejony, tag `post_class()` doda mu klasę `.sticky`, dzięki której można go sformatować w dowolny sposób, np. zdefiniować mu szare tło, większą czcionkę itp. Wystarczy w tym celu wpisać odpowiednie deklaracje CSS w arkuszu stylów w regule z selektorem `.sticky`:

```
.sticky {
    padding: 15px;
    background: #eee;
    border: 1px solid #bbb;
    color: #444;
    font-size: 18px; }
```

Powyższy kod spowoduje, że przyklejony wpis będzie miał jasnoszare tło z nieco przyciemnionym obramowaniem i rozmiar pisma 18 pikseli. Poniżej przedstawiony jest cały kod pętli za tym tagiem:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
    <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
        <h2>
            <a href="<?php the_permalink(); ?>"
              title="<?php the_title_attribute(); ?>">
                <?php the_title(); ?>
            </a>
        </h2>
        <?php the_content(); ?>
    </div>
<?php endwhile; else: ?>
    <p>Powiadomienie o błędzie lub coś w tym rodzaju.</p>
<?php endif; ?>
```

Na stronie internetowej powyższy kod spowoduje powstanie poniższego kodu HTML. Zwróć szczególną uwagę na klasy dodane przez tag `post_class()`:

```
<div id="post-1" class="post-1 post type-post status-publish format-standard
sticky hentry category-uncategorized">
    <!-- TREŚĆ WPISU -->
</div>
```

Jeśli podstawowa funkcjonalność przyklejania wpisów to dla Ciebie za mało, możesz użyć tagu warunkowego `is_sticky()`, za pomocą którego można robić różne ciekawe rzeczy.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Poniższy kod wyświetla informację, że dany wpis jest niezwykle ważny i koniecznie trzeba go przeczytać:

```
<?php if ( is_sticky() ) echo 'Bardzo ważna informacja. Koniecznie przeczytaj!'; ?>
```

Może to trochę naciągany przykład, ale czasami naprawdę dobrze jest coś wyświetlić lub zmienić, gdy na stronie pojawia się przyklejony wpis. Gdybyś na przykład sprzedawał e-booki, mógłbyś za pomocą przyklejonych wpisów promować najnowszą publikację:

```
<?php if (is_sticky()) echo 'Najnowszy e-book specjalnie dla Ciebie'; ?>
```

Nie wszystkie motywy mają widoczną obsługę przyklejania wpisów. Pamiętaj, że w niektórych motywach funkcja ta jest ukryta w interfejsie administratora.

## FORMATY WPISÓW

Formaty wpisów wprowadzono w WordPressie 3.1. Są one obok przyklejania dodatkową metodą kontrolowania treści. Istnieje 9 formatów wpisów:

- Notatka na marginesie: krótkie informacje, zwykle bez tytułu.
- Plik dźwiękowy: do użytku z plikami dźwiękowymi.
- Czat: do publikowania zapisów czatów.
- Galeria: do publikowania wpisów będących galeriami.
- Obrazek: do wyświetlania jednego obrazu.
- Odnośnik: do tworzenia odnośników do innych stron.
- Cytat: do wyświetlania cytatów.
- Informacja o stanie: do wyświetlania zwięzłych informacji o stanie.
- Film: do wyświetlania pojedynczych filmów wideo.



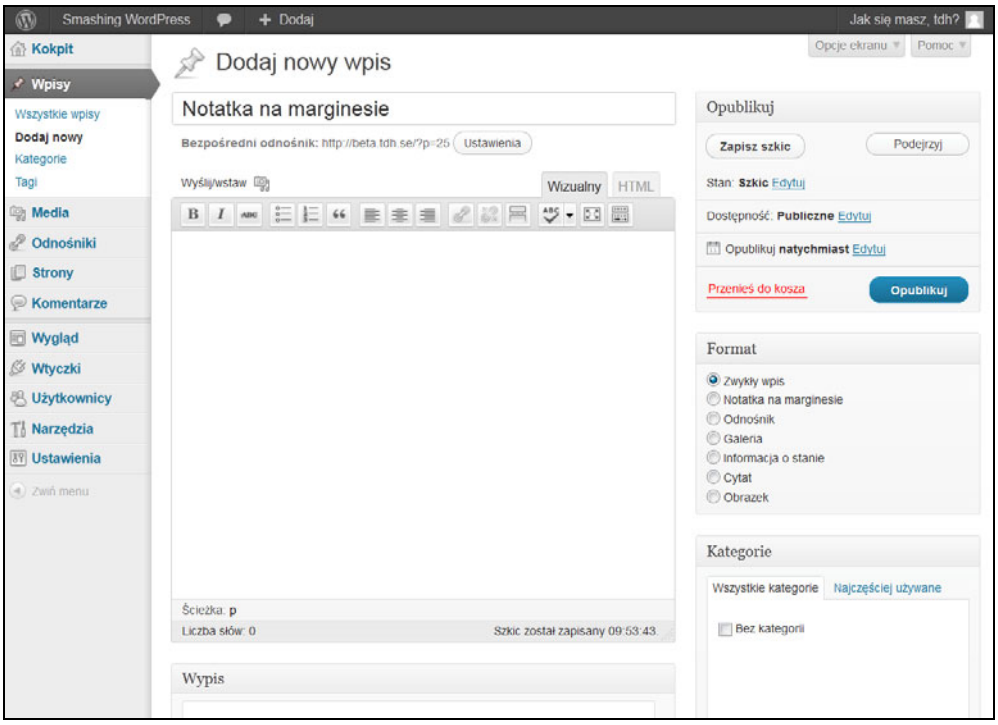
Oczywiście typów tych można używać do dowolnych celów, w zależności od tego, co z nimi robi poprzez CSS i tagi warunkowe używany przez Ciebie motyw.

Format wpisu wybiera się w okienku po prawej stronie. Na rysunku 3.9 przedstawione są możliwości do wyboru dostępne w motywie Twenty Eleven, który obsługuje sześć formatów, nie licząc standardowego (domyślnego).

Wpis opublikowany jako notatka na marginesie w motywie Twenty Eleven jest wyświetlany bez tytułu.

W takim razie do czego służą formaty wpisów? Po pierwsze: udostępniają dodatkowe klasy CSS, których można używać do formatowania wpisów. Na przykład format Notatka na marginesie dodaje klasę `format-aside` (rysunek 3.10). Dzięki temu nie trzeba samodzielnie dodawać specjalnej kategorii dla wpisów tego typu. Ponadto tag `body_class()` dodaje klasy do elementu `body`, co daje nam jeszcze większą kontrolę nad stroną.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 3.9. Okienko wyboru formatu wpisu



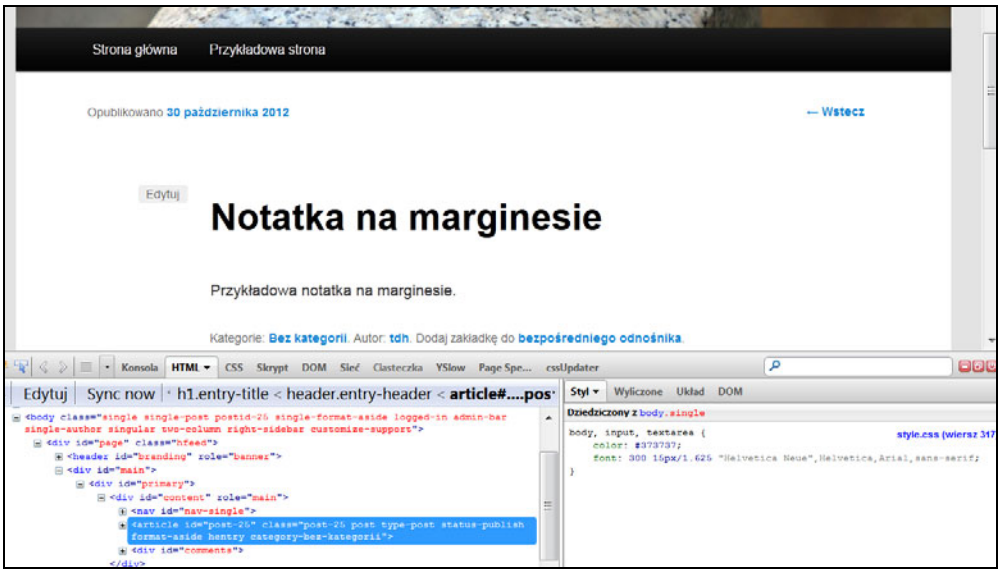
Najlepsza zabawa zaczyna się jednak dopiero wtedy, gdy użyjemy tagów warunkowych. Możemy na przykład wewnątrz pętli sprawdzić, czy dany wpis ma określony format, i w zależności od wyniku tego sprawdzenia wykonać odpowiednie działania.

```
<?php
  if ( has_post_format( 'aside' ) ) {
      echo 'To jest notatka na marginesie, a więc powinna być krótka!';
  }
?>
```

Aby dodać obsługę wybranych formatów wpisów do motywu, należy w pliku *functions.php* wywołać funkcję `add_theme_support()`, przekazując jej jako parametr tablicę tych formatów. Na przykład poniższy kod dodaje obsługę formatów *Notatka na marginesie*, *Galeria* i *Film*:

```
add_action( 'after_setup_theme', 'smashingtheme_setup' );
function smashingtheme_setup() {
  // Dodanie formatów wpisów
  add_theme_support( 'post-formats', array(
    'aside',
    'gallery',
    'video'
  ) );
}
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 3.10. Klasy CSS dodane do Notatki na marginesie, sformatowanej przy użyciu klasy format-aside



Kod ten powoduje dodanie funkcji `smashingtheme_setup()` do zaczepu `after_setup_theme`. Funkcja ta w tym przypadku wywołuje funkcję `add_theme_support()` w trybie dodawania obsługi formatów wpisów. Formaty, które chcemy włączyć, są przekazane w tablicy. W następnym rozdziale dokładniej pokażę Ci, jak wywoływać funkcje w motywach.

Więcej informacji o formatach wpisów znajduje się na stronie *http://codex.wordpress.org/Post_Formats*.

## TAG GET_TEMPLATE_PART() I FORMATY WPISÓW

Niektórzy projektanci motywów nie wpisują całej pętli w pliku *loop.php*, tylko używają tagu `get_template_part()` do pobierania kodu mniejszymi porcjami. Metoda ta jest wykorzystywana np. w motywie Twenty Eleven, w którym domyślny format wpisu jest obsługiwany przez plik *content.php*. Jeśli przyjrzysz się plikom obecnym w tym motywie, to zauważysz, że znajduje się wśród nich po jednym pliku dla każdego formatu wpisów, np. *content-aside.php*, *content-video.php* itd. W motywie Twenty Eleven początek opisanej wcześniej pętli `while` znajduje się w pliku szablonowym (*index.php*, *category.php* itd.), natomiast kod wysyłający dane na wyjście znajduje się w plikach *content.php*. W ten sposób trochę łatwiej jest zapanować nad plikami szablonowymi.

Jednak najlepsza zabawa zaczyna się dopiero wtedy, gdy połączymy tag `get_template_part()` z `get_post_format()`. Funkcja `get_post_format()` zwraca format określonego wpisu. W motywie Twenty Eleven można znaleźć następujący fragment kodu:

```
get_template_part( 'content', get_post_format() );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Funkcja `get_post_format()` zwraca format wpisu do `get_template_part()` i dodaje go do `content`, podobnie jak było w przypadku pętli w poprzednim rozdziale. Jeśli zatem formatem wpisu jest *Notatka na marginesie* (`aside`), to `get_template_part()` dołączy plik *content-aside.php*. W ten sposób można łatwo rozdzielić różne rodzaje treści i sposoby ich prezentowania.

Oczywiście konfiguracja taka najlepiej sprawdza się, gdy używane są formaty wpisów, ale i tak warto o niej pamiętać.

## FUNKCJA QUERY_POSTS()

Każdy projektant motywów i pętli WordPressa powinien umieć posługiwać się doskonałym narzędziem, jakim jest funkcja `query_posts()`. Najprościej mówiąc, funkcja ta pozwala przejąć to, co WordPress ma do wyświetlenia (najczęściej najnowsze wpisy), aby zrobić z tym coś innego, np. ograniczyć liczbę wpisów, wykluczyć jakieś kategorie albo zmienić uporządkowanie wpisów.

---

*Drobne ostrzeżenie: funkcja `query_posts()` służy do modyfikowania pętli głównej, a nie do tworzenia dodatkowych pętli. Mimo iż tworzenie pętli za jej pomocą jest możliwe, skutki tego mogą być nieprzewidywalne, a więc lepiej tego nie robić. Jeśli chcesz się dowiedzieć, w jaki sposób utworzyć kilka pętli, przeczytaj podrozdział „Tworzenie wielu pętli”.*

---



Funkcję `query_posts()` należy wywoływać przed pętlą główną:

```
<?php query_posts();
if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
  <!-- Kod formatujący wpisy itp. -->
<?php endwhile; else: ?>
  <p>Komunikat o błędzie lub coś w tym rodzaju.</p>
<?php endif; ?>
```

Jeśli chcesz, możesz trochę zredukować ilość kodu, bo przecież funkcje `query_posts()` i `have_posts()` nie muszą znajdować się w osobnych blokach PHP.

Jak działa funkcja `query_posts()`? W powyższym kodzie zasadniczo nic ona nie robi, ponieważ nie zostały przekazane jej żadne parametry, a nie ma też żadnych wartości domyślnych. Pętla w tym przypadku będzie działać w normalny sposób, zgodnie z ustawieniami WordPressa. Można ją jednak wykorzystać do zmodyfikowania zapytania SQL wysyłanego przez pętlę, aby dostosować je do własnych potrzeb.

Aby dobrze zrozumieć sposób działania funkcji `query_posts()`, obejrzymy kilka przykładów jej użycia. Za pomocą tego tagu szablonowego można kontrolować sposób prezentowania treści na wiele różnych sposobów. Ma on wiele ustawień, a w połączeniu z tagami warunkowymi pozwala uzyskać setki różnych efektów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Zaczniemy od usunięcia z list wszystkich postów należących do określonej kategorii. Powiedzmy na przykład, że w kategorii *Notatki* publikujemy tylko krótkie bezsensowne wpisy. Najpierw musimy sprawdzić, jaki identyfikator ma ta kategoria. W tym celu wystarczy zalogować się w panelu administracyjnym WordPressa i wejść na stronę edycji kategorii. Identyfikator każdej kategorii znajduje się na końcu adresu strony jej edycji:

```
http://domena.pl/wp-admin/categories.php?action=edit&cat_ID=130
```

Z fragmentu *cat_ID=130* wynika, że identyfikator interesującej nas kategorii to *130*. Pamiętaj, że identyfikatory kategorii (jak i w ogóle wszystkiego w WordPressie) nie są ze sobą powiązane i są wszystkie razem wymieszane, tzn. jedna kategoria może mieć identyfikator 3, a kolejna już 749. Jest to spowodowane tym, że liczby te przypisuje baza danych podczas tworzenia wpisów i kategorii.

Wracając do głównego tematu, nie chcemy wyświetlać wpisów należących do kategorii Notatki, która ma identyfikator 130. W dokumentacji WordPressa na temat funkcji `query_posts()` (*http://codex.wordpress.org/Template_Tags/query_posts*) wymieniono kilka sposobów na przekazanie informacji o kategorii do tej funkcji. W tym przykładzie posłużymy się parametrem `cat` przyjmującym identyfikatory jako wartości:



```
<?php
  // Argumenty
  $args = array(
    'cat' => -130
  );
  // Przekazywanie argumentów
  query_posts( $args );
?>
```

Jeśli umieścisz ten kod przed pokazaną wcześniej pętlą, nie wyświetli ona wpisów należących do kategorii o identyfikatorze 130. Zwróć uwagę na znak minus znajdujący się przed identyfikatorem kategorii. Jest on bardzo ważny, ponieważ bez niego efekt działania tego kodu byłby odwrotny, tzn. wyświetlone zostałyby posty **tylko** z tej kategorii.

Teraz wyświetlimy tylko wpisy oznaczone tagami `blue`, `green` i `yellow`:

```
<?php
  // Argumenty
  $args = array(
    'tag' => 'blue+green+yellow'
  );
  // Przekazanie argumentów
  query_posts( $args );
?>
```

W tym kodzie wykorzystany został parametr `tag`. Istnieje więcej takich parametrów. Inne rzeczy, jakie można zrobić za pomocą funkcji `query_posts()`, to wyświetlenie wpisów opublikowanych przez wybranego autora, wyświetlenie wpisów według dat oraz posortowanie wpisów według dowolnych kryteriów. Do sortowania można nawet wykorzystać dane

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

zawarte w polach własnych. Wiele z opisywanych tu możliwości będziesz wielokrotnie wykorzystywać podczas budowy nowych witryn opartych na WordPressie.

Przy korzystaniu z funkcji `query_posts()` trudność może Ci sprawiać określanie liczby wyświetlanych wpisów. Wartość ta jest określona w głównych ustawieniach WordPressa, ale na szczęście ją również można kontrolować. Przypuśćmy, że na stronie głównej chcesz wyświetlać tylko pięć wpisów, a na wszystkich innych stronach z listami wpisów ma być ich po 10. Problem ten można rozwiązać na dwa sposoby. Można utworzyć szablon o nazwie *home.php* i w nim umieścić odpowiednie wywołanie funkcji `query_posts()` albo użyć tagu warunkowego w szablonie *index.php*. Zaczniemy od pierwszego sposobu, ponieważ powoduje mniejszy bałagan i jest łatwiejszy do zastosowania. Do określenia liczby wpisów użyjemy funkcji `query_posts()` i parametru `posts_per_page`:

```
<?php
  // Argumenty
  $args = array(
    'posts_per_page' => 5
  );
  // Przekazanie argumentów
  query_posts( $args );
?>
```

Umieszczając ten kod przed pętlą w pliku *home.php*, spowodujesz, że będzie wyświetlanych tylko pięć wpisów na stronie. Proste, prawda? Teraz użyjemy tagów warunkowych i powyższy kod przeniesiemy do pliku *index.php*, ponieważ tak naprawdę wcale nie musimy tworzyć tego nowego pliku *home.php*. Musimy tylko użyć tagu warunkowego `is_home()` wewnątrz instrukcji `if`, aby sprawdzić, czy znajdujemy się na stronie głównej. Jeśli nie, nic szczególnego nie robimy. Jeśli tak, to wykonujemy instrukcję wywołania funkcji `query_posts()`. Wstawiając poniższy kod przed pętlą w pliku *index.php*, osiągniemy taki sam efekt jak poprzednio przy użyciu pliku *home.php*:

```
<?php
  // Czy to jest strona główna?
  if ( is_home() ) {
    // Argumenty
    $args = array(
        'posts_per_page' => 5
    );
    // Przekazanie argumentów
    query_posts( $args );
   }
?>
```

Skoro możesz teraz ustawiać limit wyświetlanych wpisów, to możesz również go usuwać. Jeśli parametrowi `posts_per_page` zostanie przypisana wartość `-1`, zostaną wyświetlone wszystkie wpisy spełniające kryteria pętli. Uważaj jednak z tym ustawieniem w witrynach zawierających wiele tysięcy wpisów, ponieważ na stronie głównej spowoduje ono wyświetlenie wszystkich wpisów, a to dla bazy danych nie będzie przyjemne przeżycie. To może teraz wyświetlimy wszystkie wpisy opublikowane przez autora TDH w 2009 roku?



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php
  // Argumenty
  $args = array(
    'author_name' => 'TDH',
    'year' => 2009,
    'posts_per_page' => -1
  );

  // Przekazanie argumentów
  query_posts( $args );
?>
```

Jak widać, funkcja `query_posts()` przyjmuje argumenty w postaci tablicy, dzięki czemu można zdefiniować ich dużą liczbę bez obawy, że kod będzie nieczytelny.

## CO ZAMIAST PĘTLI

Gdy opanujesz obsługę pętli, możesz w swoich motywach zacząć tworzyć wiele różnych pętli albo konstruować skomplikowane zapytania pętlowe. Podczas gdy w wielu przypadkach rozwiązania te mogą spełniać Twoje oczekiwania, czasami warto zastanowić się nad użyciem innych metod. Modyfikowanie pętli na różne sposoby, zazwyczaj przy użyciu funkcji `query_posts()`, czasami jest kompletnie niepotrzebne. Chcesz wiedzieć dlaczego? Czytaj dalej.



Po pierwsze: zastanów się, czy na pewno potrzebujesz kolejnej pętli, aby zrobić to, co chcesz. W wielu sytuacjach wystarczy tylko użyć odpowiedniego tagu szablonowego, co prawie zawsze jest lepszym rozwiązaniem. Tworzenie własnych niestandardowych pętli powinno być zarezerwowane dla sytuacji, w których nie ma innego wyjścia.

Po drugie: sprawdź, czy nie ma jakiejś wtyczki, która robiłaby to, czego potrzebujesz. Ludzie należący do społeczności WordPressa opublikowali już mnóstwo genialnych rozwiązań często spotykanych problemów i mimo iż czasami lepiej byłoby niektóre sprawy załatwić bezpośrednio w plikach motywu, to warto pamiętać, że wtyczki rozszerzają funkcjonalność systemu i są łatwiejsze w użyciu. Wiele wtyczek służących do manipulowania sposobem wyświetlania list wpisów modyfikuje działanie pętli w taki sam sposób, w jaki moglibyśmy to zrobić samodzielnie w naszych plikach. Normalnie zalecam to drugie rozwiązanie, ale co jeśli nie możemy zajmować się utrzymaniem kodu motywu przez długi czas, co jest typowe w sytuacjach, gdy robimy stronę dla klienta i oddajemy mu ją do użytkowania? Jeśli klient nie chce płacić za poprawki albo uzna, że ma wystarczające umiejętności, aby dokonywać ich we własnym zakresie, lepiej jest zainstalować wtyczkę, niż obawiać się, że coś zostanie zepsute z powodu grzebania w szablonach. Oczywiście wtyczki też mają wady, jak choćby to, że ich twórcy często je nagle porzucają albo wykorzystują w nich wycofywane funkcje WordPressa. Mimo to jednak warto poważnie rozważyć taką możliwość zamiast modyfikować pętlę.

---

*W rozdziale 8. dokładniej wyjaśniam, kiedy należy korzystać z wtyczek, aby zmienić funkcjonowanie systemu.*

---

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W końcu trzeba też wziąć pod uwagę obciążenie serwera i bazy danych. Pętla za pętlą powodują, że wykonywanych jest wiele instrukcji PHP i zapytań MySQL, które spowalniają działanie witryny. Proste rozwiązania nie zawsze są najlepsze, ale zawsze warto dążyć do osiągania zamierzonych celów przy użyciu jak najprostszych środków. Zarówno usługi hostingowe, jak i WordPress od czasu powstania przeszły długą drogę ewolucji, ale to nie znaczy, że trzeba stosować bardziej skomplikowane rozwiązania, niż to konieczne.

Krótko mówiąc, chodzi mi o to, abyś przed utworzeniem nowej pętli zawsze zastanowił się, czy na pewno jest ona niezbędna. Dzięki temu możesz uniknąć wielu potencjalnych problemów.

# TWORZENIE WIELU PĘTLI

Czasami na jednej stronie chcemy mieć kilka pętli. Jest to przydatne np. gdy mamy bardzo ważną kategorię wpisów i na stronie głównej chcemy wyświetlać tylko wpisy należące do niej albo gdy chcemy wyświetlić najnowsze wpisy na pasku bocznym. Niezależnie od tego, co planujesz, jeśli chcesz pobrać wpisy po raz drugi, musisz utworzyć nową pętlę. Umiejętność tworzenia dodatkowych pętli jest niezbędna, aby tworzyć witryny nie będące zwykłymi blogami.

Zaczniemy od najprostszego przypadku, czyli po prostu utworzymy drugą zwykłą pętlę na stronie. Potrzebna nam będzie funkcja `rewind_posts()` resetująca licznik pętli i umożliwiająca ponowne jej uruchomienie:



```
<?php
  rewind_posts();
  while ( have_posts() ) : the_post();
?>
  <!-- Dalsza część pętli... -->
```

Oczywiście ta pętla wyświetliłaby dokładnie to samo co poprzednia, co przecież nie jest naszym celem. Aby funkcja `rewind_posts()` do czegoś nam się przydała, musimy coś zmienić w nowej pętli. Możemy też używać funkcji `wp_reset_postdata()` przywracającej wartość zmiennej `$post` do stanu z oryginalnego zapytania — co może być Ci potrzebne, gdy będziesz działać we własnej pętli. Więcej informacji na temat tej funkcji znajduje się w dokumentacji WordPressa pod adresem *http://codex.wordpress.org/Function_Reference/wp_reset_postdata*.

Powiedzmy, że na dole strony chcemy wyświetlić ramkę pokazującą pięć najnowszych wpisów z kategorii *News*. Można to zrobić przy użyciu obiektu klasy `WP_Query`, który przyjmuje takie same argumenty jak funkcja `query_posts()`:

```
<?php
  // Argumenty dla obiektu klasy WP_Query
  $args = array(
    'category_name' => 'news',
    'showposts' => 5
  );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
  // Nowa pętla zapisana w zmiennej $new_loop
  $new_loop = new WP_Query( $args );
  while ( $new_loop->have_posts() ) : $new_loop->the_post(); ?>

    <!-- Dane zwrócone przez pętlę  -->
<?php
  // Koniec pętli
  endwhile;
  // Reset zapytania
  wp_reset_postdata();
?>
```

Ten kod spowodowałby wyświetlenie pięciu najnowszych wpisów z kategorii *News*, które można by było umieścić we wspomnianej ramce.

## WYŚWIETLANIE PROPONOWANYCH ARTYKUŁÓW

Innym często spotykanym zastosowaniem dodatkowych pętli jest wyświetlanie proponowanego artykułu na górze strony. Jest to jeden ze sposobów na zmianę WordPressa z tradycyjnego bloga w coś innego i ostatnio zyskuje on coraz to większą popularność, zwłaszcza w motywach czasopism, które wyglądem naśladują swoje drukowane odpowiedniki.



Najpierw musimy utworzyć pętlę pobierającą jeden wpis — oczywiście najnowszy — z kategorii *Featured*. Następnie tworzymy drugą, standardową pętlę wyświetlającą wszystkie wpisy z wszystkich kategorii. Aby to zrobić, musimy zapytanie pierwszej pętli zapisać w jego własnym obiekcie zapytania. W tym celu wywołujemy konstruktor klasy `WP_Query` i otrzymany obiekt zapisujemy w zmiennej. Klasa `WP_Query` to bardzo ważny element systemu, na którym opiera się działanie pętli. Mimo iż nie widać jej w podstawowej pętli, jest ona tam np. w wywołaniu `have_posts()`, które w istocie wygląda tak: `$wp_query->have_posts()`. Zapis `have_posts()` jest po prostu wygodnym skrótem. Klasa `WP_Query` jest ogromna i skomplikowana, przez co praca z nią wymaga dużych umiejętności programistycznych albo dużo cierpliwości do prób i błędów. Doświadczony programista powie Ci nawet, że najlepiej jest połączyć obie te cechy. Praca z klasą `WP_Query` często jednak polega też na używaniu różnych szablonowych i warunkowych tagów.

Przypomnijmy sobie tag szablonowy `query_posts()`. W jego opisie można znaleźć informację, że służy on do modyfikowania tylko pętli głównej, a więc nie możemy używać go w naszej nowej pętli. Zamiast tego przekażemy do obiektu `WP_Query` kilka zmiennych. Poniżej znajduje się odpowiedni kod:

```
<?php
    // Ustawienie tymczasowej wartości, aby uniknąć błędów
    $do_not_duplicate = null;
    // Argumenty pętli
    $args = array(
        'category_name' => 'featured',
        'showposts' => 1
    );
    // Zapytanie i pętla polecanego artykułu
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    $featured_query   = new WP_Query($args);
    while ($featured_query->have_posts()):
        $featured_query->the_post();
        // Zapisanie identyfikatora wpisu w zmiennej $do_not_duplicate
        $do_not_duplicate = $post->ID;
?>
<!-- Kod formatowania polecanego artykułu -->
<?php
    // Koniec pętli polecanego artykułu
    endwhile;
    // Resetowanie
    wp_reset_postdata();
?>
<!-- Treść mająca się znajdować między polecanym artykułem
a normalną listą wpisów  -->
<?php
    // Argumenty pętli
    $args         = array(
        'showposts' => 9
    );
    // Druga pętla
    $second_query = new WP_Query($args);
    while ($second_query->have_posts()):
        $second_query->the_post();
        // Nie wyświetlamy wpisu polecanego
        if ($post->ID == $do_not_duplicate)
            continue;
        update_post_caches($posts);
?>
<!-- Kod formatowania normalnej listy wpisów -->
<?php
    // Koniec drugiej pętli
    endwhile;
?>
```

Przyjrzyj się dokładnie temu kodowi. Na samym początku ustawiamy zmienną `$do_not_duplicate` na `null`, aby uniknąć błędów, gdyby w pętli polecanych artykułów nie było żadnych wpisów. Następnie zdefiniowana została tablica argumentów dla obiektu klasy `WP_Query`, za pomocą której określamy, jaka ma być zawartość pętli. Obiekt ten zapisany został w zmiennej `$featured_query`, która następnie została użyta w pętli `while`. Już to widzieliśmy. Na koniec wynik pętli, czyli identyfikator jednego wpisu, zapisujemy w zmiennej `$do_not_duplicate`. Zwróć uwagę, że mimo iż zmieniając argument `showposts` w tablicy argumentów dla obiektu klasy `WP_Query`, można łatwo wyświetlić dwa i więcej polecanych artykułów, identyfikatory tych wpisów trzeba by było zapisać w tablicy.

W przykładzie tym nie pokazano kodu wyświetlającego treść. Może to być cokolwiek, np. obraz, atrakcyjny nagłówek i fragment tekstu. Można wyświetlić wszystko, co można by było wyświetlić w zwykłej pętli. Na końcu zamykamy pętlę instrukcją `endwhile` i resetujemy dane wpisów za pomocą funkcji `wp_reset_postdata()`.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Druga pętla, zawierająca pozostałe wpisy, które mają być wyświetlone za polecanym artykułem, ma podobną budowę jak pierwsza. Obiekt klasy `WP_Query` jest w tym przypadku przechowywany w zmiennej `$second_query`. Najważniejsza różnica w stosunku do poprzedniej pętli polega na obecności kodu sprawdzającego, czy identyfikator każdego wpisu nie pokrywa się z identyfikatorem wpisu użytego w poprzedniej pętli. Jeśli tak, wpis ten zostaje pominięty.

## TO BYŁO NIEZŁE, ALE CZTERY PĘTLE TO DOPIERO COŚ

W tym podrozdziale sprawdzisz nabyte przed chwilą umiejętności tworzenia dodatkowych pętli. W tym przykładzie w jednej pętli będziemy pobierać trzy polecane artykuły, a pod nimi wyświetlimy trzy kolumny zawierające najnowsze wpisy z trzech różnych kategorii. W ten sposób utworzymy stronę w typowym portalowym stylu. Jeśli chodzi o pliki szablonowe, to najlepiej to zrealizować w pliku *home.php*, który jest wczytywany tylko na stronie głównej witryny. Można też użyć szablonu strony należącego do strony ustawionej jako główna dla witryny.

W tym przypadku wykorzystamy tę samą zasadę co poprzednio, tylko z czterema zapytaniami. Pierwsze będzie pobierało trzy polecane artykuły, a pod nim będą znajdować się trzy zapytania, każde dla innej kategorii. Zaczniemy od polecanych artykułów:



```
<div id="featured">
<?php
    // Ustawienie tymczasowej wartości, aby uniknąć błędów
    $do_not_duplicate = null;
    // Argumenty pętli
    $args = array(
        'category_name' => 'featured',
        'showposts' => 3
    );
    // Pętla i zapytanie dla polecanych artykułów
    $featured_query = new WP_Query( $args );
    while ( $featured_query->have_posts() ) : $featured_query->the_post();
    // Zapisanie identyfikatora wpisu w zmiennej $do_not_duplicate
    $do_not_duplicate[] = $post->ID;
?>
    <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
        <h1>
            <a href="<?php the_permalink(); ?>"
                title="<?php the_title_attribute(); ?>">
                    <?php the_title(); ?>
            </a>
        </h1>
        <?php the_excerpt(); ?>
    </div>
<?php
    // Koniec pętli polecanych artykułów
    endwhile;
    // Resetowanie
    wp_reset_postdata();
?>
</div>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Ten kod jest bardzo podobny do pokazanego w poprzednim podrozdziale, chociaż występują między nimi pewne istotne różnice. Przede wszystkim tym razem wyświetlamy trzy polecane artykuły, a nie tylko jeden, w związku z czym zamiast zwykłej zmiennej mamy tablicę `$do_not_duplicate[]`, w której zapisujemy identyfikatory tych trzech wpisów. Będziemy ich potrzebować w trzech następnych zapytaniach.

Reszta tego kodu służy już tylko do wyświetlania tytułów wpisów itp. Ponadto do kodu HTML zostały dodane identyfikatory i klasy, aby można było go odpowiednio sformatować za pomocą kaskadowych arkuszy stylów.

Teraz przyjrzymy się drugiej pętli:

```
<div class="column-left">
    <h2>Najnowsze w kategorii <span>Jabłka</span></h2>
    <ul>
    <?php
        // Argumenty pętli
        $args = array(
            'category_name' => 'apples',
            'showposts' => 10
        );
        // Zapytanie i pętla
        $apple_query = new WP_Query( $args );
        while ( $apple_query->have_posts() ) : $apple_query->the_post();
        // Wykluczenie artykułów polecanych
        if ( in_array( $post->ID, $do_not_duplicate ) ) continue;
            update_post_caches( $posts );
    ?>
        <li>
            <h3>
                <a href="<?php the_permalink(); ?>"
                    title="<?php the_title_attribute(); ?>">
                        <?php the_title(); ?>
                </a>
            </h3>
            <?php the_excerpt(); ?>
        </li>
    <?php
        // Koniec pętli
        endwhile;
        // Resetowanie
        wp_reset_postdata();
    ?>
    </ul>
</div>
```

Ta pętla niewiele różni się od pierwszej. Jedyna różnica polega na tym, że wyświetla 10 wpisów z kategorii `apples`. Mamy tu, podobnie jak poprzednio, kod sprawdzający identyfikatory wpisów, ale tym razem sprawdzamy tablicę. Kod wyświetlający wpisy jest prosty — każdy kolejny wpis jest prezentowany jako element listy. Ważnym elementem tej pętli jest wywołanie



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

funkcji `wp_reset_postdata()`, ponieważ musimy mieć czyste pole do działania przed rozpoczęciem kolejnej pętli.

Pozostałe dwie pętle są prawie takie same jak powyższa. Różnią się jedynie kategorią i nazwą zmiennej użytej do przechowywania obiektu klasy `WP_Query`. Poniżej przedstawiony jest kod w całości:

```
<div id="featured">
<?php
    // Ustawienie tymczasowej wartości, aby uniknąć błędów
    $do_not_duplicate = null;
    // Argumenty pętli
    $args = array(
        'category_name' => 'featured',
        'showposts' => 3
    );
    // Pętla polecanych artykułów
    $featured_query = new WP_Query( $args );
    while ( $featured_query->have_posts() ) : $featured_query->the_post();
    // Zapisanie identyfikatora wpisu w zmiennej $do_not_duplicate
    $do_not_duplicate[] = $post->ID;
?>
    <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
        <h2>
            <a href="<?php the_permalink(); ?>"
                title="<?php the_title_attribute(); ?>">
                    <?php the_title(); ?>
            </a>
        </h2>
        <?php the_excerpt(); ?>
    </div>
<?php
    // Koniec pętli polecanych artykułów
    endwhile;
    // Resetowanie
    wp_reset_postdata();
?>
</div>
<div class="column-left">
    <h2>Najnowsze w kategorii <span>Jabłka</span></h2>
    <ul>
        <?php
        // Argumenty pętli
        $args = array(
            'category_name' => 'apples',
            'showposts' => 10
        );
        // Pętla
        $apple_query = new WP_Query( $args );
        while ( $apple_query->have_posts() ) : $apple_query->the_post();
        // Eliminacja wpisów polecanych
        if ( in_array( $post->ID, $do_not_duplicate ) ) continue;
            update_post_caches( $posts );
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    ?>
        <li>
            <h3>
                <a href="<?php the_permalink(); ?>"
                  title="<?php the_title_attribute(); ?>">
                    <?php the_title(); ?>
                </a>
            </h3>
            <?php the_excerpt(); ?>
        </li>
    <?php
        // Koniec pętli
        endwhile;
        // Resetowanie
        wp_reset_postdata();
    ?>
    </ul>
</div>
<div class="column-left">
    <h2>Najnowsze w kategorii <span>Pomarańcze</span></h2>
    <ul>
    <?php
        // Argumenty pętli
        $args = array(
            'category_name' => 'oranges',
            'showposts' => 10
        );
        // Pętla
        $orange_query = new WP_Query( $args );
        while ( $orange_query->have_posts() ) : $orange_query->the_post();
        // Eliminacja artykułów polecanych
        if ( in_array( $post->ID, $do_not_duplicate ) ) continue;
            update_post_caches( $posts );
    ?>
        <li>
            <h3>
                <a href="<?php the_permalink(); ?>"
                  title="<?php the_title_attribute(); ?>">
                    <?php the_title(); ?>
                </a>
            </h3>
            <?php the_excerpt(); ?>
        </li>
    <?php
        // Koniec pętli
        endwhile;
        // Resetowanie
        wp_reset_postdata();
    ?>
    </ul>
</div>
<div class="column-right">
    <h2>Najnowsze w kategorii <span>Cytryny</span></h2>
    <ul>
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    <?php
        // Argumenty pętli
        $args = array(
            'category_name' => 'lemons',
            'showposts' => 10
        );
        // Pętla
        $lemon_query = new WP_Query( $args );
        while ( $lemon_query->have_posts() ) : $lemon_query->the_post();
        // Eliminacja artykułów polecanych
        if ( in_array( $post->ID, $do_not_duplicate ) ) continue;
            update_post_caches( $posts );
    ?>
        <li>
            <h3>
                <a href="<?php the_permalink(); ?>"
                  title="<?php the_title_attribute(); ?>">
                    <?php the_title(); ?>
                </a>
            </h3>
            <?php the_excerpt(); ?>
        </li>
    <?php
        // Koniec pętli
        endwhile;
        // Resetowanie
        wp_reset_postdata();
    ?>
    </ul>
</div>
```



Mamy gotowe cztery pętle: jedną wyświetlającą artykuły polecane i trzy wyświetlające najnowsze wpisy z różnych kategorii. Kod ten można z łatwością zastosować w każdym motywie WordPressa, tylko trzeba by było zmienić nazwy kategorii.

## OSWAJANIE SIĘ Z PĘTLĄ

Pętla przy pierwszym zetknięciu na pewno może się wydawać skomplikowana. Ale na początek wystarczy nauczyć się tworzyć podstawową wersję tej konstrukcji, co przecież nie jest trudne. Przy odrobinie praktyki można szybko zrozumieć zasadę działania tego mechanizmu i nawet nie spostrzeżesz, kiedy zaczniesz go dostosowywać do własnych potrzeb w tworzonych przez siebie motywach. Pętla jest bardzo ważna między innymi dlatego, że wielu wtyczek i tagów szablonowych można używać tylko w niej. Jednak sztuczki w rodzaju tworzenia dodatkowych pętli, aby wyświetlać różne treści na jednej stronie, są potrzebne, dopiero gdy zdecydujemy się na rozszerzanie możliwości WordPressa.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jeśli po lekturze tego rozdziału nadal nie masz pewności, czy dobrze rozumiesz działanie pętli, nie martw się. Im więcej przykładów przeanalizujesz i im więcej modyfikacji wykonasz samodzielnie, tym łatwiej Ci będzie wszystko pojąć. A będziesz miał okazję to zrobić już w następnym rozdziale, w którym zaczynamy tworzenie motywów. Będziemy się tym zajmować przez kilka kolejnych rozdziałów, a więc zobaczysz jeszcze niejedną pętlę. Ponadto lepiej poznasz tagi szablonowe i inne narzędzia służące w WordPressie do kontrolowania treści.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

CZĘŚĆ II

# PROJEKTOWANIE I PROGRAMOWANIE MOTYWÓW WORDPRESSA

Rozdział 4. Motywy do WordPressa — wiadomości podstawowe

Rozdział 5. Motywy potomne

Rozdział 6. Motywy dla zaawansowanych

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 4

# MOTYWY DO WORDPRESSA — WIADOMOŚCI PODSTAWOWE

**JEŚLI CHCESZ NAUCZYĆ SIĘ** zaawansowanej obsługi WordPressa, najpierw musisz poznać podstawy budowy motywów. Możesz zacząć od przerabiania domyślnego motywu albo dowolnego innego wysokiej jakości motywu z listy przedstawionej w dodatku B. Jednak wielu rzeczy trudno się domyślić, tylko modyfikując istniejący kod. Dlatego w tym rozdziale opisane są podstawowe wiadomości dotyczące budowy motywów WordPressa i sposobu ich działania. Nauczysz się je dostosowywać do własnych potrzeb. Wiedza nabyta w tym rozdziale będzie Ci potrzebna w kolejnych rozdziałach.

W tym rozdziale utworzysz nowy motyw od podstaw i dowiesz się więcej na temat różnych plików szablonowych. Wśród nich będzie plik arkuszy stylów oraz obszerny plik *functions.php*, a więc krótko mówiąc, nauczysz się posługiwać wszystkimi narzędziami, o których była mowa w poprzednich rozdziałach.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PODSTAWY BUDOWY MOTYWU

Wiesz już mniej więcej, jak są zbudowane motywy WordPressa i co zawierają. Teoretycznie ich podstawowym zadaniem jest oddzielenie warstwy projektowej od kodu, jednak w istocie w bardziej skomplikowanych motywach i tak można znaleźć mnóstwo kodu.

Każdy motyw mieści się w jednym folderze zawierającym pliki szablonowe i inne potrzebne do jego działania pliki. Dwa pliki, które są absolutnie niezbędne, to *style.css* i *index.php*. Pierwszy zawiera nagłówek identyfikujący motyw (i ogólne arkusze stylów), a drugi reprezentuje podstawowy układ stron. Jeśli lubisz wyzwania, utwórz motyw do WordPressa zawierający tylko te dwa pliki i sprawdź, co przy ich użyciu uda Ci się osiągnąć. Będzie to bardzo pouczające ćwiczenie.

## PODSTAWOWE ELEMENTY MOTYWU

Na rysunku 4.1 przedstawiony jest typowy układ bloga. Na górze znajduje się nagłówek rozciągający się na całą szerokość strony. Zawartość nagłówka znajduje się w pliku *header.php*. Poniżej po lewej stronie mamy obszar właściwej treści (plik *index.php*), a po prawej pasek boczny (*sidebar.php*). Na samym dole znajduje się stopka zawarta w pliku *footer.php*.

Rysunek 4.1. Typowy układ bloga

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Oczywiście blog nie musi tak wyglądać. Można zmienić układ elementów, coś usunąć, coś dodać itd. Niemniej jednak jest to bardzo często spotykana konfiguracja, którą można domyślnie spotkać także w standardowych motywach Twenty Ten i Twenty Eleven. Znajdziesz wiele motywów o podobnym rozkładzie plików, ale to nie oznacza, że jest to jedyna możliwość.

Mimo iż do budowy motywu wystarczą jedynie pliki *style.css* i *index.php*, poprzestanie tylko na nich nie jest dobrym pomysłem. Oprócz nich powinno się jeszcze utworzyć przynajmniej pliki *header.php* i *footer.php*. Dołącza się je do pliku *index.php* za pomocą tagów `get_header()` i `get_footer()`. Warto też dodać plik *comments.php* do obsługi komentarzy, który dołącza się za pomocą tagu `comments_template()`. To samo dotyczy paska bocznego. Dla niego również powinno się utworzyć szablon dołączany za pomocą tagu `get_sidebar()`.

W pliku *header.php* powinna znajdować się deklaracja typu dokumentu itp. rzeczy. Bliżej przyjrzymy się temu w dalszej części rozdziału. Ponadto w pliku tym ustawia się favikonę, generuje tytuły stron i wpisuje słowa kluczowe, aby ułatwić wyszukiwarkom znalezienie stron. Ponadto nie może w nim zabraknąć łącza do arkusza stylów oraz wywołania funkcji `wp_head()`, od której rozpoczyna się działanie całego systemu.

Plik *footer.php* powinien zawierać wywołanie funkcji `wp_footer()` wyłączającej WordPressa i oczywiście wszystkie potrzebne znaczniki zamykające wcześniej otwartych elementów. Należy pamiętać, aby zamknąć elementy `body` i `html`, a także wszystkie inne otwarte, lecz nie zamknięte w nagłówku.



W końcu w pliku *comments.php* powinien znajdować się kod wyświetlający komentarze i formularz dodawania komentarzy. Obecnie wystarczy użyć tylko kilku tagów szablonowych, dzięki czemu zawartość plików *comments.php* nie jest już tak zagmatwana, jak była kiedyś. Najprostszy z wszystkich plików jest *sidebar.php*, który wyświetla zawartość paska bocznego.

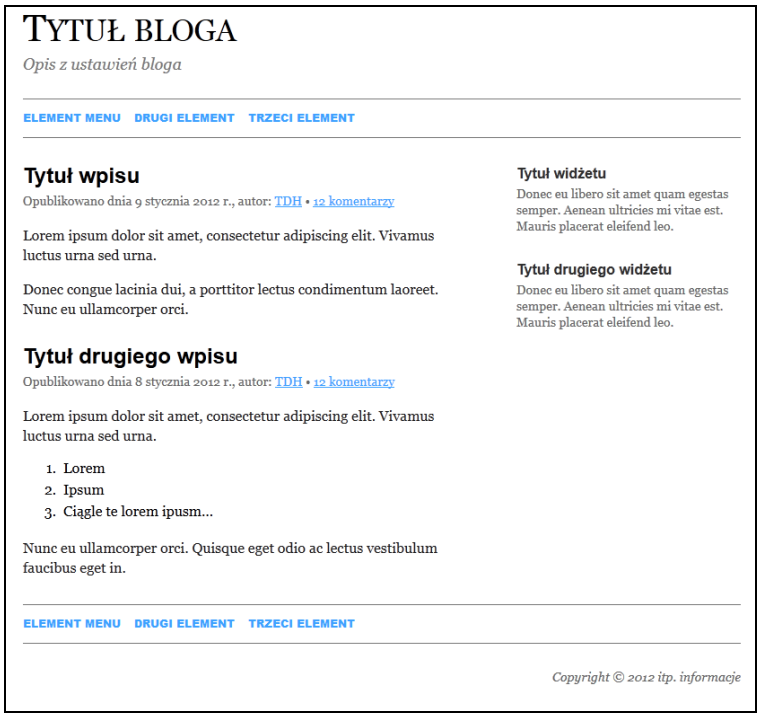
Jak zapewne zauważyłeś, wszystko kręci się wokół pliku *index.php*. Jednak w niektórych sytuacjach plik ten może zostać zastąpiony przez inne pliki. Gdy na przykład jesteśmy na stronie głównej, kontrolę przejmuje plik *home.php*, a *index.php* w takich sytuacjach jest używany tylko wówczas, gdy pliku *home.php* nie ma. To samo dotyczy widoku pojedynczych wpisów. Jeśli istnieje, to do wyświetlania wpisów używany jest plik *single.php*, a jeśli go nie ma, to brany jest *index.php*. Tak naprawdę to jeśli utworzysz wszystkie możliwe pliki szablonowe, plik *index.php* nigdy nie będzie potrzebny.

Skoro ustaliliśmy podstawowe zagadnienia, czas zakasać rękawy i wziąć się do roboty!

# CO BĘDZIEMY ROBIĆ

To, jak zaprojektujesz swój motyw, zależy tylko od Ciebie. W tym rozdziale będziemy posługiwać się prostym projektem bloga widocznym na rysunku 4.2, który został utworzony jedynie w oparciu o pliki HTML. Czasami do rozpoczęcia pracy wystarczy tylko to.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974






Rysunek 4.2. Prosty projekt strony HTML, który będzie stanowił podstawę naszego motywu

Sam wygląd projektu może nie powala na kolana, bo jest to w zasadzie proste odzwierciedlenie układu przedstawionego na rysunku 4.1. Projekt ten będziemy jednak dzielić na pliki szablonowe, z których powstanie nasz pierwszy motyw do WordPressa.

Jeśli chcesz obejrzeć kod HTML tego projektu, możesz pobrać pliki do tej książki ze strony *ftp://ftp.helion.pl/przyklady/podwsm.zip*. Nie przedstawiam go w książce, bo zajęłby bardzo dużo miejsca.

## KILKA SŁÓW NA TEMAT JĘZYKA HTML5

Projekt, na którym będziemy pracować, jest napisany przy użyciu języka HTML5, przez co nie da się go poprawnie wyświetlić w starszych przeglądarkach, z Internet Explorerem do wersji 8 włącznie. Jeśli chcesz, aby Twój motyw współpracował także ze starszymi przeglądarkami, a jednocześnie chciałbyś móc używać nowych elementów semantycznych HTML5, takich jak `header`, `article` i `footer`, to musisz coś z tym zrobić.

Jest kilka rozwiązań tego problemu. Można wszystkie elementy umieścić w dodatkowych elementach `div` i w ten sposób poradzić sobie z problemem przestarzałych przeglądarek, ale to spowoduje bałagan w kodzie i wówczas stosowanie elementów semantycznych będzie się mijać z celem. Innym rozwiązaniem jest użycie JavaScriptu, aby zmusić przeglądarkę Internet Explorer do poprawnego interpretowania kodu — i na to rozwiązanie ja się zdecydowałem. Remy Sharp napisał pewien skrypt JavaScript, który można zapisać na swoim serwerze albo dołączać do stron z serwerów Google, jeśli nie przeszkadza Ci jedno żądanie HTTP więcej:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Jest to tzw. znacznik warunkowy (nie ma nic wspólnego z tagami warunkowymi WordPressa) sprawdzający, czy przeglądarka użyta do otwarcia strony to Internet Explorer w wersji starszej od 9. Jeśli tak, następuje wczytanie skryptu *html5.js* z serwisu Google. Więcej na temat tego narzędzia można przeczytać na blogu Remiego Sharpa we wpisie z 2009 roku, w którym ogłasza on powstanie tego skryptu (*http://remysharp.com/2009/01/07/html5-enabling-script*). Później pokażę Ci, jak prawidłowo dołącza się skrypty w WordPressie.

Teraz możemy „pociąć" nasz plik HTML do motywu WordPressa!

# TWORZENIE PLIKÓW SZABLONOWYCH

Przedstawiony plik HTML potniemy kawałek po kawałku i przeniesiemy do plików szablonowych WordPressa, aby utworzyć kompletny motyw. Szczegółowo opiszę wszystkie te pliki, abyś zobaczył, jak wygląda proces powstawania motywu, chociaż — jak się wkrótce przekonasz — różne rzeczy można robić na różne sposoby. Na razie postaram się, aby wszystko było jak najprostsze.



Pracę należy zacząć od utworzenia folderu na pliki motywu. Nazwiemy go *simple-blog* (więcej na temat tej nazwy napiszę przy okazji opisywania pliku *style.css*). Znajdować się w nim będą same pliki PHP i jeden plik *style.css*.

## DEKLARACJA MOTYWU W PLIKU STYLE.CSS

Motyw musi zawierać plik *style.css*. Bez niego ani rusz. Na początku tego pliku musi się znajdować deklaracja motywu, a pod nią mogą znajdować się zwykłe reguły CSS. Można wszystkie style zapisać w tym pliku bezpośrednio albo importować je z innych plików — wybór należy do Ciebie.

Poniżej przedstawiona jest tylko część pliku *style.css* zawierająca deklarację motywu. Reguły stylistyczne dla projektu HTML nie są pokazane. Poniższy kod znajduje się na samym początku tego pliku:

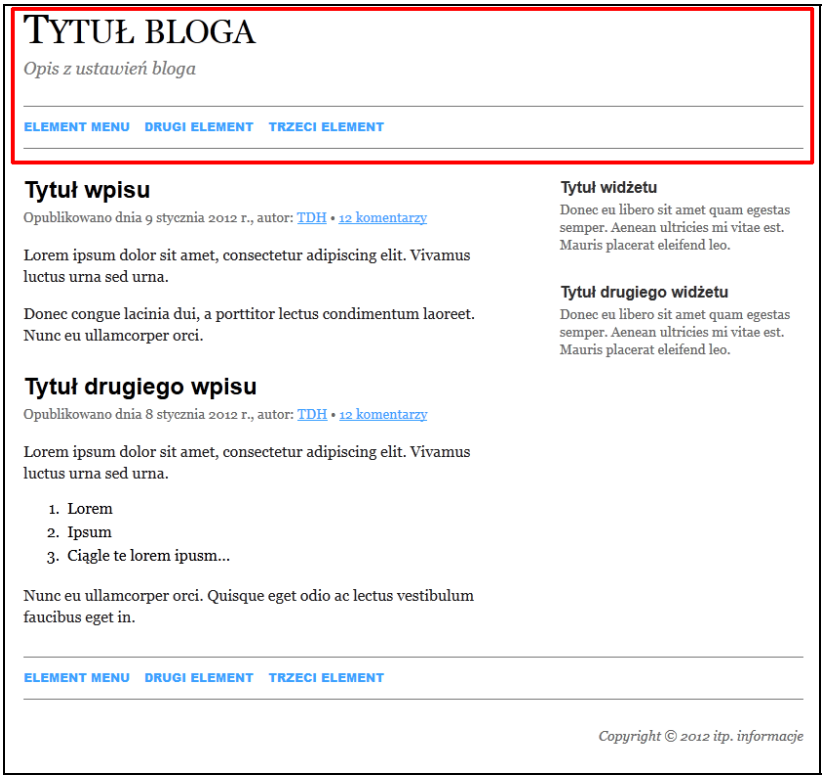
```
/*
Theme Name: Simple Blog
Theme URI: http://tdh.me/wordpress/simple-blog/
Author: Thord Daniel Hedengren
Author URI: http://tdh.me/
Description: A simple light weight blog theme.
Version: 1.0
License: GNU General Public License
License URI: license.txt
*/
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Pod tym komentarzem mogą znajdować się dowolne reguły CSS. Zwróć uwagę na napisy `License` i `License URI`. Pierwszy z nich informuje, na jakiej licencji udostępniany jest motyw, a drugi, że pełny tekst tej licencji można znaleźć w pliku *license.txt*. Najlepiej jedną z tych informacji usunąć, aby nie wywoływać nieporozumień.

## PLIK HEADER.PHP

Cięcie projektu rozpoczniemy od nagłówka, którego zawartość będzie zapisana w pliku *header.php*. Na rysunku 4.3 pokazano, co dokładnie ma zostać przeniesione do tego pliku.




Rysunek 4.3. Część w ramce zostanie przeniesiona do pliku header.php



Ponieważ plik *header.php* w tym motywie będzie służył jako nagłówek we wszystkich sytuacjach (aczkolwiek w razie potrzeby można utworzyć dowolną liczbę nagłówków), musi być wystarczająco ogólny, aby zawsze można było go użyć. Poniższy kod skopiowałem z pliku HTML do pliku *header.php*:

```
<!DOCTYPE HTML>
<html>

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <title>Tytuł bloga</title>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    <link rel="stylesheet" type="text/css" media="all" href="style.css" />

    <!-- Fix for Internet Explorer prior to version 9
         by Remy Sharp http://remysharp.com/2009/01/07/html5-enabling-script/ -->
    <!--[if lt IE 9]>
    <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>

<body>

    <div id="outer-wrap">
        <div id="inner-wrap">

            <header id="header-container">
                <hgroup>
                    <h1 id="site-title">Tytuł bloga</h1>
                    <h2 id="site-description">Opis z ustawień bloga</h2>
                </hgroup>

                <nav>
                    <div class="menu">
                        <ul>
                            <li><a href="#">Element menu</a></li>
                            <li><a href="#">Drugi element</a></li>
                            <li><a href="#">Trzeci element</a></li>
                        </ul>
                    </div>
                </nav>
            </header> <!-- Koniec #header-container -->
```



To oczywiście nie wystarczy. Teraz musimy ożywić ten nagłówek za pomocą funkcji WordPressa. Teoretycznie wystarczyłoby dodanie wywołania tylko funkcji `wp_head()`, aby uruchomić WordPressa, jednak mamy większe ambicje.

Przede wszystkim musimy zadbać o ustawienie w tym pliku odpowiedniego kodowania znaków, wstawienie poprawnych tytułów stron w zależności od tego, która strona jest w danej chwili przeglądana, dołączenie właściwych arkuszy stylów, wstawienie adresu URL do pingowania, co ułatwia komunikację między blogami. Musimy też załadować potrzebne pliki JavaScript i na końcu wywołać funkcję `wp_head()`. Poniżej znajduje się przerobiony kod nagłówka zawierający wszystkie wymienione dodatki:

```
<!DOCTYPE HTML>
<html <?php language_attributes(); ?>>
<head>
    <meta charset="<?php bloginfo( 'charset' ); ?>" />
    <title>
    <?php
        // Tytuł
        wp_title( '|', true, 'right' );
        // Dodanie nazwy bloga
        bloginfo( 'name' );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    ?>
    </title>
    <link rel="profile" href="http://gmpg.org/xfn/11" />
    <link rel="stylesheet" type="text/css" media="all" href="<?php
      bloginfo( 'stylesheet_url' ); ?>" />
    <link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>" />
    <!-- Naprawia stronę w przeglądarkach Internet Explorer starszych od wersji 9
      — Remy Sharp, http://remysharp.com/2009/01/07/html5-enabling-script/ -->
    <!--[if lt IE 9]>
    <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
    <?php
        // Wywołanie funkcji wp_head() powinno znajdować się przed znacznikiem zamykającym nagłówek
        wp_head();
    ?>
</head>
```

Zwróć uwagę, że zawartość elementu `title` jest tworzona przy użyciu instrukcji `if`. Jej użycie jest konieczne, jeśli chcemy, aby tytuł strony był zawsze właściwy, ponieważ nigdy nie wiadomo, na jakiej stronie będzie znajdował się użytkownik. Za pomocą tagów warunkowych sprawdzamy, czy otwarta strona to archiwum, pojedynczy wpis, czy strona, i na podstawie tej informacji podejmujemy odpowiednie czynności. Jak widać, także skrypt włączający obsługę HTML5 w Internet Explorerze jest dołączany warunkowo, zamiast być dodawany w normalny sposób za pomocą funkcji `wp_enqueue_script()`, której powinno się do tych celów używać. Skrypt ten oddzieliłem od reszty strony także po to, aby nie zakłócać działania mechanizmu sprawdzania kodu HTML przeglądarek oraz dlatego, ponieważ skrypt ten powinien być wczytywany tylko przez wersje Internet Explorera starsze od 9. Dołączony jest też skrypt JavaScript obsługujący odpowiedzi na komentarze. Jest on wczytywany tylko dla pojedynczych wpisów i stron i tylko w taki sposób można go używać. Na końcu oczywiście mamy niezbędne wywołanie funkcji `wp_head()` znajdujące się bezpośrednio nad zamykającym znacznikiem `</head>`. W tym miejscu też wiele wtyczek i innych dodatków do motywów wstawia kod, który ma działać w nagłówku.



Teraz zajmiemy się pozostałą częścią pliku *header.php*. Musimy zadbać o przypisanie elementowi `body` odpowiednich klas (przydatnych do formatowania stron za pomocą CSS), aby zostały wyświetlone odpowiedni tytuł i opis witryny oraz istniała możliwość tworzenia menu za pomocą narzędzi WordPressa:

```
<body <?php body_class(); ?>>

    <div id="outer-wrap">
        <div id="inner-wrap">
            <header id="header-container">
                <hgroup>
                <?php if (is_home() || is_front_page()) { ?>
                    <h1 id="site-title">
                        <a href="<?php echo home_url(); ?>" title="<?php
                          bloginfo( 'name' ); ?>">
                            <?php bloginfo( 'name' ); ?>
                        </a>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            </h1>
            <h2 id="site-description">
                <?php bloginfo( 'description' ); ?>
            </h2>
        <?php } else { ?>
            <div id="site-title">
                <a href="<?php echo home_url(); ?>" title="<?php
                  bloginfo( 'name' ); ?>">
                    <?php bloginfo( 'name' ); ?>
                </a>
            </div>
            <div id="site-description">
                <?php bloginfo( 'description' ); ?>
            </div>
        <?php } ?>
        </hgroup>
        <nav>
        <?php
            // Górne menu nawigacyjne
            wp_nav_menu( array(
                 'theme_location' => 'top-navigation'
            )
        ); ?>
        </nav>
    </header> <!-- #header-container - koniec -->
```



Jest to bardzo prosty kod. Gdy zajmiemy się właściwą treścią, dowiesz się, że podobnie jak tag `body_class()` służy do dodawania klas do elementu `body`, istnieje też tag dodający klasy dla wpisów. Ogólnie chodzi o to, aby mieć jak największą kontrolę nad projektem i móc formatować w dowolny sposób różne części witryny. Może zechcesz, aby strony należące do określonej kategorii miały żółte tło. Nic prostszego. Wystarczy tylko dodać odpowiedni arkusz stylów do pliku CSS i gotowe.

Tytuł witryny jest zwracany przez funkcję `bloginfo()`. Uważam, że w elemencie `h1` powinien się on znajdować tylko na stronie głównej, ponieważ jest to jedyne miejsce, w którym tytuł witryny jest najważniejszy. Dlatego właśnie umieściłem w tym miejscu dodatkowy warunek odpowiednio umieszczający tytuł w nagłówku `h1` lub elemencie `div`. To samo oczywiście dotyczy opisu witryny, który również znajduje się w tym kodzie warunkowym.

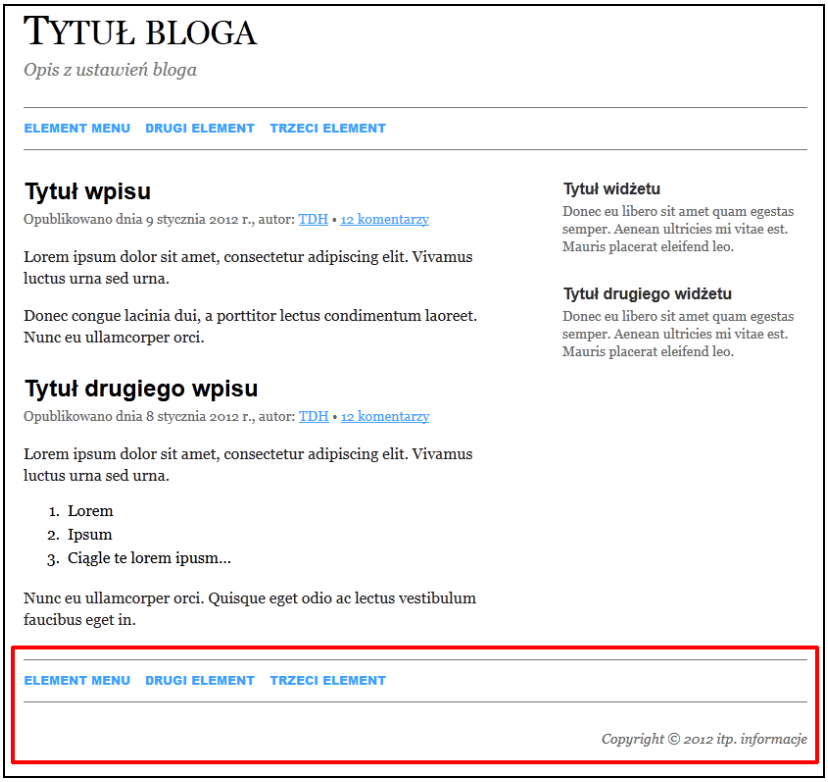
Na końcu znajduje się menu. Elementy `div.menu` i `ul` z szablonu HTML zastąpiliśmy wywołaniem funkcji `wp_nav_menu()`. Jest to tag wyświetlający menu, ale żeby działał, należy zaimplementować jego obsługę w pliku *functions.php*. Zajmiemy się tym później. Jeśli będziesz chciał przetestować swój motyw, zanim zaczniesz pisać kod w pliku *functions.php*, możesz ten fragment umieścić w komentarzu, aby nie przeszkadzał.

Plik *header.php* jest już ukończony! Nie było trudno, prawda?

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## PLIK FOOTER.PHP

Stopka w tym motywie jest bardzo prosta, więc zajmiemy się nią teraz, aby mieć ją już z głowy. Pomijamy zatem na razie całą treść główną strony i przechodzimy na sam dół. Podobnie jak plik *header.php*, również *footer.php* będzie dołączany na każdej stronie witryny, chociaż gdybyśmy chcieli, to moglibyśmy też utworzyć kilka stopek. Na rysunku 4.4 pokazano, nad czym dokładnie teraz pracujemy.




Rysunek 4.4. Oznaczona część zostanie umieszczona w pliku footer.php

Z pliku HTML do pliku *footer.php* skopiujemy poniższy kod:

```
<footer id="footer-container">
    <nav>
        <div class="menu">
            <ul>
                <li><a href="#">Element menu</a></li>
                <li><a href="#">Drugi element</a></li>
                <li><a href="#">Trzeci element</a></li>
            </ul>
        </div>
    </nav>

    <p>Copyright &copy; 2012 itp. informacje</p>
</footer> <!-- #footer-container - koniec -->
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
        </div> <!-- #inner-wrap - koniec -->
    </div> <!-- #outer-wrap - koniec -->

</body>

</html>
```

Nie ma tu wiele do zrobienia. Mamy kolejne menu, które musimy udostępnić do edytowania użytkownikom za pomocą funkcji `wp_nav_menu()`, oraz tekst dotyczący praw autorskich. Na koniec bezpośrednio przed zamykającym znacznikiem `</body>` wpiszemy wywołanie funkcji `wp_footer()`, analogicznie jak w nagłówku dodaliśmy funkcję `wp_header()`.

Poniżej znajduje się kod z opisanymi dodatkami i zwykłą instrukcją PHP wyświetlającą rok obok tekstu określającego prawa autorskie:

```
        <footer id="footer-container">
            <nav>
            <?php
                // Dolne menu nawigacyjne
                wp_nav_menu( array(
                     'theme_location' => 'bottom-navigation' )
                );
            ?>
            </nav>
            <p>
                Copyright &copy; <?php echo date( 'Y' ); ?>
                <a href="<?php echo home_url(); ?>" title="<?php
                    bloginfo( 'name' ); ?>">
                        <?php bloginfo( 'name' ); ?>
                    </a>
                </p>
            </footer> <!-- #footer-container - koniec -->
        </div> <!-- #inner-wrap - koniec -->
    </div> <!-- #outer-wrap - koniec -->
<?php
    // Zamknięcie WordPressa przed zamknięciem elementu body
    wp_footer();
?>
</body>
</html>
```

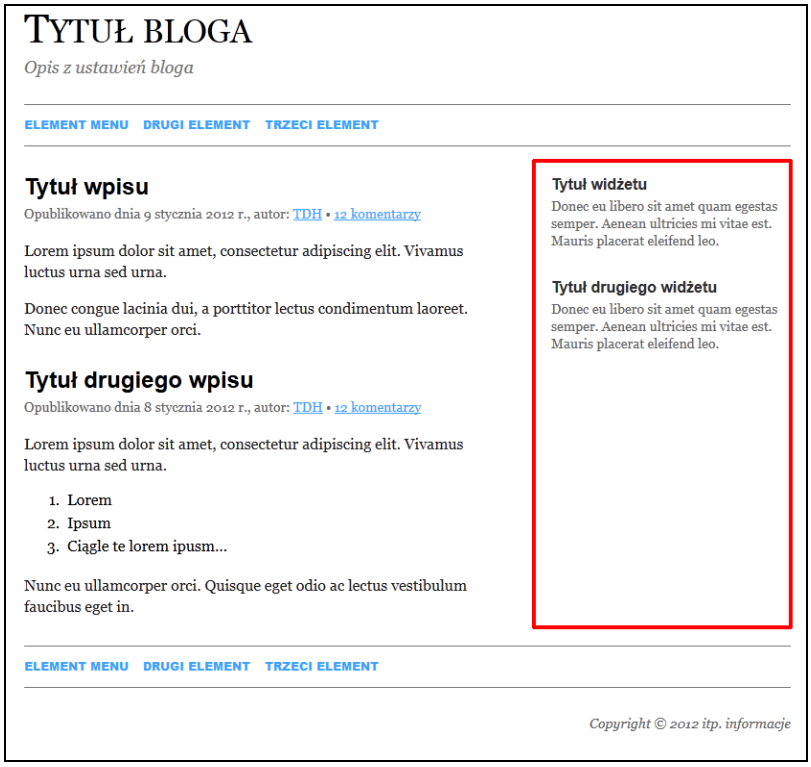
Obsługę menu trzeba zaimplementować w pliku *functions.php*, a więc na razie to pominiemy. Jest jeszcze tylko jedna rzecz, o której chciałbym tu wspomnieć, zanim przejdę dalej, do ciekawszych zagadnień. Chodzi o informację o prawach autorskich. Umieściłem ją tutaj tylko po to, aby pokazać, jak się to robi, ale pamiętaj, że nie każdy użytkownik motywu może chcieć taką informację wyświetlać. Wiele osób chce udostępniać swoje treści za darmo albo na jednej z licencji Creative Commons (*http://creativecommons.org/*). Najlepiej jest dodać do motywu opcję pozwalającą to ustawić, do czego jeszcze wrócimy w dalszej części książki. Na razie tylko zwracam Twoją uwagę na tę kwestię.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## PRAWA KOLUMNA: PLIK SIDEBAR.PHP

Zanim przejdziemy do treści głównej, zajmiemy się prawą kolumną, czyli tzw. **paskiem bocznym**. Interesujący nas obszar strony jest zaznaczony ramką na rysunku 4.5.




Rysunek 4.5. Prawa kolumna

Gdybyśmy chcieli, moglibyśmy wczytywać różne prawe kolumny w zależności od otwartej strony, ale na potrzeby tej prezentacji wystarczy nam jedna. Poniższy kod HTML wkleiłem do pliku o nazwie *sidebar.php*:

```
<aside id="sidebar-container">
    <ul id="sidebar">
        <li class="widget-container">
            <h3 class="widget-title">Tytuł widżetu</h2>
            <p>Donec eu libero sit amet quam egestas semper. Aenean ultricies mi
            vitae est. Mauris placerat eleifend leo.</p>
        </li>
        <li class="widget-container">
            <h3 class="widget-title">Tytuł drugiego widżetu</h2>
            <p>Donec eu libero sit amet quam egestas semper. Aenean ultricies mi
            vitae est. Mauris placerat eleifend leo.</p>
        </li>
    </ul>
</aside> <!-- #sidebar-container - koniec -->
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Oczywiście w pasku bocznym nie powinno być treści zastępczej, a więc będziesz musiał ją usunąć. Prawdziwa treść będzie generowana dynamicznie i wstawiana w postaci widżetów w miejscu nazywanym **obszarem widżetów**. W większości motywów dostępnych jest od jednego do kilku takich obszarów, w których właściciel witryny może umieszczać standardowe widżety WordPressa i widżety dodawane przez różne wtyczki. Cała prawa kolumna tego motywu jest obszarem widżetów, co oznacza, że trzeba wprowadzić pewne zmiany w kodzie:

```
<aside id="sidebar-container">
    <ul id="sidebar">
    <?php
        // Jeśli pasek boczny jest pusty, wyświetlana jest statyczna treść
        if ( !dynamic_sidebar( 'right-column' ) ) : ?>
        <li>Umieść jakieś widżety w obszarze widżetów w <em>prawej kolumnie</em>!</li>
    <?php endif; ?>
    </ul>
</aside> <!-- #sidebar-container - koniec -->
```

Treść zastępcza została zamieniona na kod PHP szukający paska bocznego o nazwie *Right column*, a jeśli jest on pusty — wyświetlający element `li` z instrukcjami.

Aby paski boczne działały, muszą zostać zadeklarowane w pliku *functions.php*. Zrobimy to wkrótce, a na razie coś innego.



## TREŚĆ GŁÓWNA: PLIK INDEX.PHP

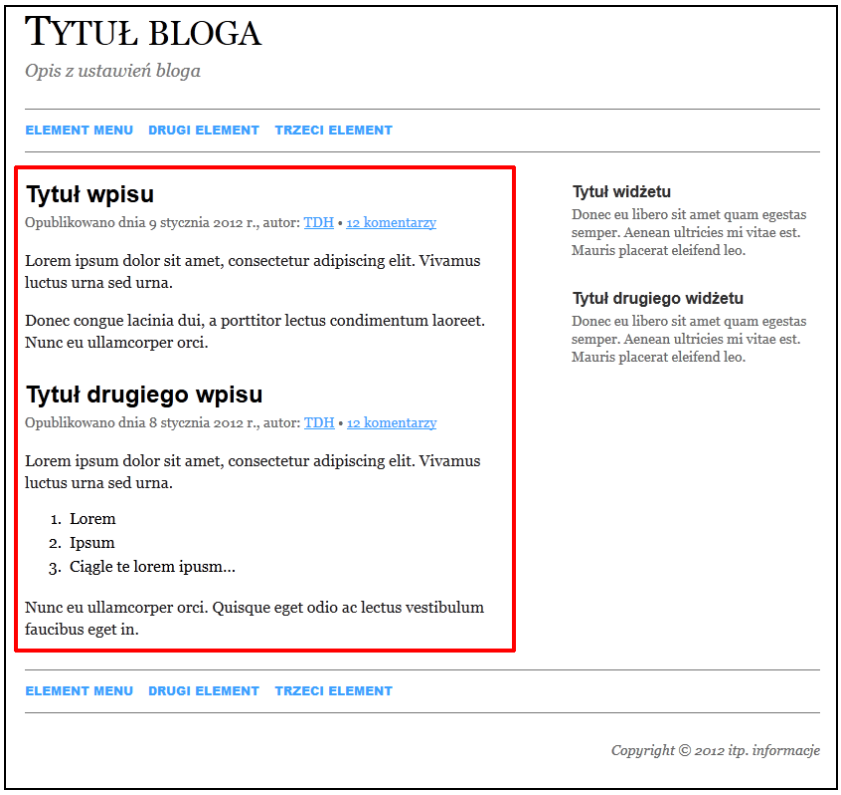
Czas okiełznać właściwą treść witryny. Na rysunku 4.6 zaznaczono interesującą nas część w szablonie HTML, która w tym przypadku składa się z dwóch wpisów, ale oczywiście powinna być możliwość zwiększenia tej liczby.

Wszystkie poprzednie pliki były szablonami i plik *index.php* nie będzie tu wyjątkiem, mimo że jest on nieco inny od tamtych plików. Plik *index.php* jest dodatkowo szablonem awaryjnym używanym do wyświetlania stron, dla których nie ma specjalnego szablonu. Później wnikliwiej zbadamy tę sprawę, ale warto już teraz wiedzieć, że plik ten może Cię uratować w sytuacji, gdy zapomnisz utworzyć szablon albo dodasz jakieś nowe funkcje.

Poniżej znajduje się kod HTML, który skopiowałem do pliku *index.php*. Znajduje się w nim też kod prawej kolumny, która została już umieszczona w pliku *sidebar.php*. Nie przejmuj się tym, wkrótce to naprawimy.

```
<div id="main-container">
    <section id="content-container">
        <article class="post">
            <header>
                <h2 class="entry-title">Tytuł wpisu</h2>
                <p class="entry-meta">Opublikowano dnia <time datetime="2012-01-
                09">9 stycznia 2012 r.</time>, autor: <a href="#">TDH</a> &bull; <a
                href="#comments">12 komentarzy</a></p>
            </header>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974






Rysunek 4.6. Plik index.php obsługuje wyświetlanie treści głównej i nie tylko

```
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
            luctus urna sed urna.</p>
            <p>Donec congue lacinia dui, a porttitor lectus condimentum laoreet.
            Nunc eu ullamcorper orci.</p>
        </article>

        <article class="post">
            <header>
                <h2 class="entry-title">Tytuł drugiego wpisu</h2>
                <p class="entry-meta">Opublikowano dnia <time datetime="2012-01-
                08">8 stycznia 2012 r.</time>, autor: <a href="#">TDH</a> &bull; <a
                href="#comments">12 komentarzy</a></p>
            </header>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
            luctus urna sed urna.</p>
            <ol>
                <li>Lorem</li>
                <li>Ipsum</li>
                <li>Ciągle te lorem ipusm...</li>
            </ol>
            <p>Nunc eu ullamcorper orci. Quisque eget odio ac lectus vestibulum
            faucibus eget in.</p>
        </article>
    </section> <!-- #main-container - koniec -->
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    <aside id="sidebar-container">
        <ul id="sidebar">
            <li class="widget-container">
                <h2 class="widget-title">Tytuł widżetu</h2>
                <p>Donec eu libero sit amet quam egestas semper. Aenean ultricies
                mi vitae est. Mauris placerat eleifend leo.</p>
            </li>
            <li class="widget-container">
                <h2 class="widget-title">Tytuł drugiego widżetu</h2>
                <p>Donec eu libero sit amet quam egestas semper. Aenean ultricies
                mi vitae est. Mauris placerat eleifend leo.</p>
            </li>
        </ul>
    </aside> <!-- #sidebar-container - koniec -->
</div>
```

Wszystko po kolei. Najpierw musimy dołączyć pliki *header.php*, *footer.php* i *sidebar.php*. Do tego celu użyjemy funkcji `get_header()`, `get_footer()` i `get_sidebar()`. Pierwszą z nich należy umieścić na samym początku pliku, a drugą na końcu. Natomiast funkcję `get_sidebar()` wpisujemy w miejsce kodu prawej kolumny w następujący sposób:

```
<?php get_header(); ?>

<div id="main-container">
    <section id="content-container">
        <article class="post">
            <header>
                <h2 class="entry-title">Tytuł wpisu</h2>
                <p class="entry-meta">Opublikowano dnia <time datetime="2012-01-
                09">9 stycznia 2012 r.</time>, autor: <a href="#">TDH</a> &bull; <a
                href="#comments">12 komentarzy</a></p>
            </header>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
            luctus urna sed urna.</p>
            <p>Donec congue lacinia dui, a porttitor lectus condimentum laoreet.
            Nunc eu ullamcorper orci.</p>
        </article>

        <article class="post">
            <header>
                <h2 class="entry-title">Tytuł drugiego wpisu</h2>
                <p class="entry-meta">Opublikowano dnia <time datetime="2012-01-
                08">8 stycznia 2012 r.</time>, autor: <a href="#">TDH</a> &bull; <a
                href="#comments">12 komentarzy</a></p>
            </header>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus
            luctus urna sed urna.</p>
            <ol>
                <li>Lorem</li>
                <li>Ipsum</li>
                <li>Ciągle te lorem ipusm...</li>
            </ol>
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            <p>Nunc eu ullamcorper orci. Quisque eget odio ac lectus vestibulum
            faucibus eget in.</p>
        </article>
    </section> <!-- #main-container - koniec -->

<?php get_sidebar(); ?>

</div>

<?php get_footer(); ?>
```

Te trzy tagi dołączą do strony zawartość plików *header.php*, *footer.php* i *sidebar.php*.

Teraz musimy przystosować plik *index.php* do wyświetlania treści. Do tego potrzebna nam będzie opisana w poprzednim rozdziale pętla. Treść zastępcza posłuży nam jako wzór, do którego osiągnięcia powinniśmy dążyć. Dodajemy pętlę, usuwamy treść zastępczą i sprawdzamy, jak to wygląda:

```
<?php get_header(); ?>

<div id="main-container">
    <section id="content-container">

    <?php
        // Początek pętli
        if ( have_posts() ) : while ( have_posts() ) : the_post();
    ?>

    <!-- TREŚĆ WYŚWIETLONA PRZEZ PĘTLĘ -->

    <?php
        // Koniec pętli
        endwhile;
        // Pętla nic nie zwróciła?
        else :
    ?>

            <article id="post-0" class="post no-results not-found">
                <header>
                    <h2 class="entry-title">Nic nie znaleziono </h2>
                </header>
                <p>Przepraszamy, ale nic dla Ciebie nie znaleźliśmy. Spróbuj znaleźć
                to, czego szukałeś, korzystając z wyszukiwarki.</p>
                <?php get_search_form(); ?>
            </article>

    <?php
        // Koniec
        endif;
    ?>
    </section> <!-- #main-container - koniec -->

<?php get_sidebar(); ?>
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
</div>

<?php get_footer(); ?>
```

Ponieważ pętla została szczegółowo omówiona w poprzednim rozdziale, na pewno bez trudu rozpoznałeś konstrukcję `if ( have_posts() ) : while ( have_posts() ) : the_post();`. Za nią znajduje się komentarz informujący, że w jego miejscu będzie wyświetlona treść. Dalej znajduje się jeszcze klauzula `else` obsługująca przypadki nieznalezienia wpisów do wyświetlenia. Oczywiście w takich sytuacjach musi być też wyświetlona odpowiednia wiadomość, która została już napisana. Nie ma tu nic niezwykłego. Jeśli potrzebujesz odświeżenia pamięci, jeszcze raz przeczytaj poprzedni rozdział.

Aby wyświetlić prawdziwą treść, musimy w miejsce komentarza wstawić odpowiedni kod. Korzystając ze wzoru, jakim jest treść zastępcza, użyjemy różnych tagów szablonowych, aby wyświetlić samą treść, jej tytuł, datę i godzinę publikacji oraz autora:

```
<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
    <header>
        <h2 class="entry-title">
            <a href="<?php the_permalink(); ?>" title="<?php the_title_attribute();
            ?>" rel="bookmark">
                <?php the_title(); ?>
            </a>
        </h2>
        <p class="entry-meta">
            Opublikowano dnia <time datetime="<?php echo get_the_date(); ?>">
                <?php the_time(); ?>,</time>
            autor: <?php the_author_link(); ?>
        <?php
            // Czy komentarze są otwarte?
            if ( comments_open() ) : ?>
            &bull; <?php comments_popup_link( 'Brak komentarzy', '1 komentarz',
            'komentarzy: %' ); ?>
        <?php endif; ?>
        </p>
    </header>
    <?php
        // Treść
        the_content();
    ?>
</article>
```

Nie ma co tu opisywać. Sprawdzamy, czy komentarze są włączone, a jeśli tak, wyświetlamy także już dodane komentarze.

W całości zawartość pliku *index.php* wygląda teraz następująco:

```
<?php get_header(); ?>
<div id="main-container">
    <section id="content-container">
    <?php
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
        // Początek pętli
        if ( have_posts() ) : while ( have_posts() ) : the_post();
        // Wyświetlenie daty w jednym miejscu na każdej stronie
        the_date( '', '<h3 class="the_date">', '</h3>' );
    ?>
    <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
        <header>
            <h2 class="entry-title">
                <a href="<?php the_permalink(); ?>" title="<?php
                the_title_attribute(); ?>" rel="bookmark">
                    <?php the_title(); ?>
                </a>
            </h2>
            <p class="entry-meta">
                Opublikowano dnia <time datetime="<?php echo get_the_date(); ?>">
                    <?php the_time(); ?>,</time>
                autor: <?php the_author_link(); ?>
            <?php
                // Czy komentarze są otwarte?
                if ( comments_open() ) : ?>
                &bull; <?php comments_popup_link( 'Brak komentarzy', '1 komentarz',
                'komentarzy: %' );
                endif;
            ?>
            </p>
        </header>
        <?php
            // Treść
            the_content();
        ?>
    </article>
    <?php
        // Wczytanie komentarzy, jeśli strona zawiera pojedynczy artykuł
        if ( is_singular() ) {
            comments_template( '', true );
        }
        // Koniec pętli
        endwhile;
        // W pętli nie ma nic do wyświetlenia?
        else :
    ?>
        <article id="post-0" class="post no-results not-found">
            <header>
                <h2 class="entry-title">Nic nie znaleziono</h2>
            </header>
            <p>Przepraszamy, ale nic dla Ciebie nie znaleźliśmy. Spróbuj znaleźć
            to, czego szukałeś, korzystając z wyszukiwarki.</p>
            <?php get_search_form(); ?>
        </article>
    </section> <!-- #main-container - koniec -->
<?php get_sidebar(); ?>
</div>
<?php get_footer(); ?>
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W kodzie tym znajdują się dwa drobne dodatki, o których nie było jeszcze mowy. Pierwszym z nich jest funkcja `the_date()` wyświetlająca każdą datę tylko raz, tzn. jeśli dwa wpisy mają tę samą datę, zostanie ona wyświetlona tylko raz. Poniższy wiersz kodu znajduje się na samym początku pętli:

```
// Wyświetlenie daty w jednym miejscu na każdej stronie

the_date( '', '<h3 class="the_date">', '</h3>' )
```

Druga nowość to tag warunkowy sprawdzający, czy otwarta strona to pojedynczy wpis lub strona pojedynczego artykułu, a jeśli tak, wywołujący funkcję `comments_template()`:

```
// Wczytanie komentarzy, jeśli strona zawiera pojedynczy artykuł
if ( is_singular() ) {
    comments_template( '', true );
}
```

Na tym etapie nasz motyw jest już prawie w pełni funkcjonalny. Pozostało jeszcze tylko zdefiniować kilka elementów w pliku *functions.php*, czym wkrótce się zajmiemy. Jednak wcześniej dopracujemy to, co mamy.

## PRZENOSZENIE PĘTLI DO OSOBNEGO PLIKU



Mimo że nie jest to obowiązkowe, wygodniej jest mi pracować, gdy kod pętli umieszczę w osobnym pliku. Czasami przenoszę całą pętlę, a czasami tylko sam wynik jej działania. W tym przykładzie przeniesiemy do osobnego pliku tylko część odpowiedzialną za wyświetlanie treści. Spójrz na poprzedni przykład kodu. Chcemy wyjąć z niego blok elementu `article` i przenieść go do nowego pliku. Ponieważ kod ten określa domyślny sposób prezentacji wpisów, umieścimy go w pliku o nazwie *content.php*. Plik ten będziemy dołączać do pliku *index.php* za pomocą poniższej instrukcji:

```
get_template_part( 'content', get_post_format() );
```

Powyższa funkcja pobiera treść pliku *content-X.php*, gdzie *X* to format wpisu zwrócony przez `get_post_format()`. W przypadku zwykłego wpisu pobierana jest zawartość pliku *content.php*, natomiast w przypadku formatu Galeria najpierw będzie szukany szablon o nazwie *content-gallery.php*, a jeśli nie zostanie on znaleziony, zostanie użyty plik *content.php*. Ma to umożliwić wielokrotne wykorzystanie pliku *content.php* dla różnych formatów wpisów, jeśli tego chce projektant motywu.

Poniżej znajduje się kompletny kod źródłowy pliku *index.php*:

```
<?php get_header(); ?>
<div id="main-container">
    <section id="content-container">
    <?php
        // Początek pętli
        if ( have_posts() ) : while ( have_posts() ) : the_post();
        // Pobranie odpowiedniego typu treści
        get_template_part( 'content', get_post_format() );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
        // Wczytanie komentarzy, jeśli strona zawiera pojedynczy artykuł
        if ( is_singular() ) {
            comments_template( '', true );
        }
        // Koniec pętli
        endwhile;

        // W pętli nie ma nic do wyświetlenia?
        else :
    ?>
        <article id="post-0" class="post no-results not-found">
            <header>
                <h2 class="entry-title">Nic nie znaleziono</h2>
            </header>
            <p>Przepraszamy, ale nic dla Ciebie nie znaleźliśmy. Spróbuj znaleźć
            to, czego szukałeś, korzystając z wyszukiwarki.</p>
            <?php get_search_form(); ?>
        </article>
    </section> <!-- #main-container - koniec -->
<?php get_sidebar(); ?>
</div>
<?php get_footer(); ?>
```



Poniżej natomiast przedstawiony jest kod źródłowy pliku *content.php*:

```
<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
    <header>
        <h2 class="entry-title">
            <a href="<?php the_permalink(); ?>" title="<?php the_title_attribute();
            ?>" rel="bookmark">
                <?php the_title(); ?>
            </a>
        </h2>
        <p class="entry-meta">
            Opublikowano dnia <time datetime="<?php echo get_the_date(); ?>">
                <?php the_time(); ?>,</time>
            autor: <?php the_author_link(); ?>
        <?php
            // Czy komentarze są otwarte?
            if ( comments_open() ) : ?>
            &bull; <?php comments_popup_link( 'Brak komentarzy', '1 komentarz',
            'komentarzy: %' ); ?>
        <?php endif; ?>
        </p>
    </header>
    <?php
        // Treść
        the_content();
    ?>
</article>
```

Mając to za sobą, możemy utworzyć kilka kolejnych plików szablonowych!

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## POJEDYNCZE WPISY I STRONY

W większości motywów używa są zarówno pliku *single.php*, jak i *page.php*. Pierwszy z nich służy do wyświetlania treści pojedynczych wpisów, a drugi do wyświetlania stron. W prostym motywie, jak ten, który tutaj budujemy, nie ma potrzeby stosowania takiego podziału. Wystarczy użyć kilku tagów warunkowych, aby uzyskać odpowiedni efekt bez komplikowania pliku *index.php*. Jednak mimo to utworzymy oba te pliki, aby nauczyć się ich obsługi.

Zaczniemy od pliku *single.php*, który w istocie niewiele będzie się różnił od pliku *index.php*. Tak naprawdę jest to jego uproszczona wersja, pozbawiona kodu „awaryjnego" i zawierająca wywołanie `get_template_part( 'content', 'single' )` zamiast wywołania dla konkretnego typu wpisu, ponieważ pojedynczy wpis powinien wyglądać zawsze tak samo, przynajmniej w tym przypadku.

Poniżej znajduje się kompletny kod źródłowy pliku *single.php*:

```
<?php get_header(); ?>

<div id="main-container">
    <section id="content-container">

    <?php
        // Początek pętli
        while ( have_posts() ) : the_post();

        // Pobranie odpowiedniego typu treści
        get_template_part( 'content', 'single' );

        // Komentarze
        comments_template( '', true );

        // Koniec pętli
        endwhile;
    ?>

    </section> <!-- #main-container - koniec -->

    <?php get_sidebar(); ?>

    </div>

    <?php get_footer(); ?>
```

Funkcja `get_template_part()` najpierw poszuka pliku *content-single.php*, a jeśli go nie znajdzie, wczyta zawartość pliku *content.php*. Teraz utworzymy zatem plik *content-single.php*, którego zawartość będzie bardzo podobna do pliku *content.php*, lecz bez łącza w tytule, z elementem `h1` zamiast `h2` oraz bez sprawdzenia, czy otwarta jest pojedyncza strona, czy pojedynczy wpis (bo już to wiemy):

```
<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
    <header>
        <h1 class="entry-title">
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            <?php the_title(); ?>
        </h1>
        <p class="entry-meta">
            Opublikowano dnia <time datetime="<?php echo get_the_date(); ?> <?php
            the_time(); ?>"><?php the_date(); ?> <?php the_time(); ?>,</time>
            autor: <?php the_author_link(); ?>
        <?php
            // Czy komentarze są otwarte?
            if ( comments_open() ) : ?>
            &bull; <?php comments_popup_link( 'Brak komentarzy', '1 komentarz',
            'komentarzy: %' ); ?>
        <?php endif; ?>
        </p>
    </header>
</article>
```

Oprócz opisanych zmian, które łatwo znaleźć, gdy porówna się ten kod z kodem z pliku *content.php*, dodałem też listę kategorii, do których przypisany jest dany wpis.

Przechodzimy do pliku *page.php*, którego zawartość będzie bardzo podobna do poprzedniego. W zasadzie moglibyśmy po prostu skopiować treść pliku *single.php*, wkleić ją do *page.php* i byłoby po sprawie. No może poza jednym drobnym szczegółem. Strony nie mają kategorii ani tagów. Musimy to poprawić w pliku *content-single.php*, ponieważ teraz jego zawartość jest używana w pliku *page.php*. Gdybyśmy chcieli, aby strony wyglądały inaczej niż wpisy, najprawdopodobniej utworzylibyśmy plik *content-page.php* i dołączalibyśmy go za pomocą funkcji `get_template_part( 'content', 'page' )`. Jednak tak nie jest w tym przypadku. Zamiast tego w pliku *content-single.php* część dotyczącą kategorii i tagów umieścimy w tagu warunkowym `is_single()`, który zwraca `true` tylko wtedy, gdy otwarty jest pojedynczy wpis dowolnego typu. Do wpisów zaliczają się także załączniki, ale nie strony.



Poniżej znajduje się poprawiony kod pliku *content-single.php*:

```
<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
    <header>
        <h1 class="entry-title">
            <?php the_title(); ?>
        </h1>
    <?php if ( is_single() ) : ?>
        <p class="entry-meta">
            Opublikowano dnia <time datetime="<?php echo get_the_date(); ?> <?php
            the_time(); ?>"><?php the_date(); ?> <?php the_time(); ?>,</time>
            autor: <?php the_author_link(); ?>
        <?php
            // Czy komentarze są otwarte?
            if ( comments_open() ) : ?>
            &bull; <?php comments_popup_link( 'Brak komentarzy', '1 komentarz',
            'Komentarzy: %' ); ?>
        <?php endif;
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            // Wyświetlenie kategorii i tagów w pojedynczych wpisach
            if ( is_singular( 'post' ) ) :
        ?>
            <br />Kategorie: <?php the_category( ', ' ); ?>
            <?php the_tags( ' Tagi: ', ', ', '' ); ?>
        </p>
    <?php
        endif;
    endif; ?>
    </header>
    <?php
        // Treść
        the_content();
    ?>
</article>
```

Co w takim razie z komentarzami? Musimy utworzyć plik *comments.php*, który jest szablonem do prezentacji komentarzy. Poniżej znajduje się zmodyfikowana i gotowa do formatowania za pomocą CSS wersja standardowego pliku *comments.php* z motywu Twenty Eleven:

```
<div id="comments">
<?php
    // Czy wpis jest chroniony hasłem?
    if ( post_password_required() ) : ?>
    <p class="nopassword">
        Ten wpis jest chroniony hasłem. Wpisz hasło.
    </p>
</div>
<?php
        // Powrót
        return;
    endif;

    // Sprawdzenie, czy są komentarze
    if ( have_comments() ) : ?>
    <h2 id="comments-title">
        Ten wpis <?php comments_number( 'nie ma komentarzy', 'ma jeden komentarz',
        'ma % komentarzy' ); ?>
    </h2>

    <?php
        // Nawigacja komentarzy
        if ( get_comment_pages_count() > 1 && get_option( 'page_comments' ) ) : ?>
        <nav id="comment-nav-above">
            <div class="nav-previous">
                <?php previous_comments_link( '&larr; Starsze komentarze' ); ?>
            </div>
            <div class="nav-next">
                <?php next_comments_link( 'Nowsze komentarze &rarr;' ); ?>
            </div>
        </nav>
    <?php endif; ?>
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    <ol class="commentlist">
    <?php
        // Wyświetlanie listy komentarzy
        wp_list_comments();
    ?>
    </ol>
<?php
    // Nawigacja komentarzy
    if ( get_comment_pages_count() > 1 && get_option( 'page_comments' ) ) : ?>
    <nav id="comment-nav-above">
        <div class="nav-previous">
            <?php previous_comments_link( '&larr; Starsze komentarze' ); ?>
        </div>
        <div class="nav-next">
            <?php next_comments_link( 'Nowsze komentarze &rarr;' ); ?>
            </div>
        </nav>
    <?php endif;

// Skończone
endif; ?>

<?php
    // Formularz dodawania komentarzy
    comment_form();
?>

</div>
```



W zrozumieniu tego kodu powinny pomóc Ci zamieszczone w nim komentarze. Przede wszystkim należy zauważyć, że na wyjście komentarze wysyła funkcja `wp_list_comments()`. Funkcja ta przyjmuje kilka parametrów i można nawet dostosować jej wynik za pomocą funkcji zwrotnej zdefiniowanej w pliku *functions.php*. My w tym przykładzie nie modyfikujemy sposobu jej działania. Jeśli chcesz dowiedzieć się więcej na temat możliwości tagu szablonowego `wp_list_comments()`, zajrzyj na stronę *http://codex.wordpress.org/Function_Reference/wp_list_comments*.

## SZABLONY ARCHIWÓW

W motywie można utworzyć kilka szablonów archiwów. W istocie jeśli plik *index.php* jest używany do wyświetlania listy wpisów, to stanowi archiwum. Typ archiwów potrzebnych w motywie zależy od tego, jaka treść jest prezentowana w witrynie. W większości motywów tworzy się ogólny szablon domyślny *archive.php* i na nim można poprzestać. Dla kategorii jest plik *category.php*, a dla tagów — *tag.php*. Jest też plik *date.php* dla archiwów według dat itd. Dostępne szablony zostały wypisane w dalszej części tego rozdziału, w podrozdziale „Pliki szablonowe”. Wszystkie one działają mniej więcej tak samo.

W naszym motywie utworzymy tylko szablon *archive.php*, który będzie używany do wyświetlania wszystkich archiwów (gdyby go nie było, używany byłby plik *index.php*). Odpowiedni nagłówek w zależności od typu archiwum będziemy wyświetlać przy użyciu tagów warunkowych:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php get_header(); ?>

<div id="main-container">
    <section id="content-container">
        <header class="page-header">
            <h1 class="page-title">
                <?php if ( is_day() ) : ?>
                    Archiwum dzienne dla <span><?php echo get_the_date(); ?></span>
                <?php elseif ( is_month() ) : ?>
                   Archiwum miesięczne dla <span><?php echo get_the_date( 'F Y' ); ?>
                      </span>
                <?php elseif ( is_year() ) : ?>
                    Archiwum roczne dla <span><?php echo get_the_date( 'Y' ); ?>
                        </span>
                <?php elseif ( is_category() ) : ?>
                    <?php single_cat_title('Aktualnie przeglądane '); ?>
                <?php elseif ( is_tag() ) : ?>
                    <?php single_tag_title('Aktualnie przeglądane '); ?>
                <?php else : ?>
                    Archiwa
                <?php endif; ?>
            </h1>
        </header>

    <?php
        // Początek pętli
        while ( have_posts() ) : the_post();

        // Pobranie odpowiedniego typu treści
        get_template_part( 'content', get_post_format() );

        // Koniec pętli
        endwhile;
    ?>

    </section> <!-- #main-container - koniec -->

<?php get_sidebar(); ?>

</div>

<?php get_footer(); ?>
```

Jest to w zasadzie plik *index.php* pozbawiony kodu awaryjnego, ale za to zawierający instrukcję `if-else` wyświetlającą odpowiedni nagłówek w zależności od sytuacji. Obsługiwane są kategorie dni, miesięcy, lat, kategorii i tagów oraz dodatkowo na wszelki wypadek została dodana ogólna obsługa pozostałych przypadków. Powinieneś już dobrze rozumieć budowę tych plików.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## BŁĘDY 404, WYSZUKIWARKA I ZRZUTY EKRANU

Jest jeszcze kilka szablonów, które powinny znajdować się w każdym motywie. Dwa najważniejsze z nich to plik *404.php* wykorzystywany w sytuacjach, gdy użytkownik wpisze adres URL strony, która nie istnieje, i *search.php* używany do wyświetlania wyników wyszukiwania. Poniżej znajduje się kod prostego pliku *404.php*. Powinien on zawierać bardziej pomocne informacje, ale wrócimy do tego nieco później.

```
<?php get_header(); ?>

<div id="main-container">
    <section id="content-container">

        <article id="post-0" class="post no-results not-found">
            <header>
                <h1 class="entry-title">404 Nie znaleziono strony</h2>
            </header>
            <p>Chyba nie ma takiej strony. Może łącze, którego użyłeś, zawiera
            błąd?</p>
            <p>Jeśli chcesz, możesz spróbować znaleźć to, czego tu szukałeś.</p>
            <?php get_search_form(); ?>
        </article>

    </section> <!-- #main-container - koniec -->

<?php get_sidebar(); ?>

</div>

<?php get_footer(); ?>
```



Jest to prosty plik szablonowy zawierający wiadomość o błędzie. Zostanie on użyty przez system, gdy użytkownik spróbuje wejść na nieistniejącą stronę.

Plik *search.php* jest bardzo podobny do pliku *index.php*. Podobieństwo to jest tak duże, że gdy nie ma pliku *search.php*, to zamiast niego używany jest właśnie *index.php*. Mamy dwie możliwości do wyboru. Możemy skopiować zawartość pliku *index.php* i odpowiednio ją dostosować albo dodać kilka tagów do obsługi wyświetlania wyników wyszukiwania. W tym przykładzie zastosujemy drugie z wymienionych podejść. Dodaj poniższy kod do pliku *index.php*, bezpośrednio pod elementem `section#content-container`:

```
<?php
    // Jeśli jest to strona wyników wyszukiwania
    if ( is_search() ) :
?>
    <header class="page-header">
        <h1 class="page-title">
            Szukana fraza:<br />
            <span><?php the_search_query(); ?></span>
        </h1>
    </header>
<?php endif; ?>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Powyższy kod jest zamknięty w prostym tagu warunkowym sprawdzającym, czy dana strona wyświetla wyniki wyszukiwania. Jeśli tak, następuje wyświetlenie szukanej frazy w elemencie `h1`. Czasami to wystarczy.

I jeszcze na koniec: nie zapomnij umieścić w folderze motywu pliku *screenshot.png* przedstawiającego ten motyw. Plik ten powinien mieć wymiary 300×225 pikseli, ponieważ będzie wyświetlany w sekcji motywów w panelu administracyjnym WordPressa. Zadbaj o to, aby był to obraz dobrej jakości i aby wiernie przedstawiał wygląd motywu. W przeciwnym razie wprowadzisz użytkowników w błąd i nie dodasz motywu do zbiorów WordPress.org, gdybyś chciał to zrobić.

# PLIK FUNCTIONS.PHP

Podstawowym miejscem pracy projektanta motywów jest plik *functions.php*. Plik ten jest ładowany wraz z motywem i można w nim robić prawie wszystko, jak we wtyczkach. W pliku tym można zaimplementować różne funkcje, takie jak ustawianie tła i nagłówków, wyświetlanie polecanych artykułów, tworzenie menu i wiele więcej. Bardziej szczegółowy opis tego pliku znajduje się w dalszej części rozdziału. Poniżej przedstawiony jest tylko prosty kod znajdujący się obecnie w naszym motywie:



```
<?php
// Ustawienie szerokości treści w motywie
if ( ! isset( $content_width ) )
    $content_width = 500;
// Konfiguracja motywu
add_action( 'after_setup_theme', 'simpleblog_themesetup' );
function simpleblog_themesetup() {
    // Automatyczne linki kanałów RSS
    add_theme_support( 'automatic-feed-links' );
    // Dodanie funkcji menu nawigacyjnych do zaczepu init
    add_action( 'init', 'simpleblog_register_menus' );
    // Dodanie funkcji pasków bocznych do zaczepu widgets_init
    add_action( 'widgets_init', 'simpleblog_register_sidebars' );
    // Dodanie do kolejki plików JavaScript w zaczepie wp_enqueue_scripts
    add_action( 'wp_enqueue_scripts', 'simpleblog_load_scripts' );
}
// Rejestracja menu
function simpleblog_register_menus() {
    register_nav_menus(
        array(
            'top-navigation' => 'Top navigation',
            'bottom-navigation' => 'Bottom navigation'
        )
    );
}
// Zarejestrowanie obszarów widżetów
function simpleblog_register_sidebars() {
    // Obszar widżetów w prawej kolumnie
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    register_sidebar( array(
        'name' => 'Right column',
        'id' => 'right-column',
        'before_widget' => '<li id="%1$s" class="widget-container %2$s">',
        'after_widget' => '</li>',
        'before_title' => '<h3 class="widget-title">',
        'after_title' => '</h3>',
    ) );
}
// Kolejkowanie skryptów JavaScript
function simpleblog_load_scripts() {
    // Kolejkowanie JavaScriptu dla komentarzy dzielonych na wątki, jeśli funkcja ta zostanie włączona
    if ( is_singular() && get_option( 'thread_comments' ) && comments_open() )
        wp_enqueue_script( 'comment-reply' );
}
?>
```

Zmienna `$content_width` oznacza, że podstawowa szerokość treści w tym motywie wynosi 500 pikseli. Dalej znajduje się funkcja konfiguracyjna motywu. Została ona podłączona do zaczepu `after_setup_theme`, co oznacza, że wszystkie funkcje zapisane w funkcji `simpleblog_themesetup()` służą do konfiguracji motywu. Definicje tych funkcji znajdują się niżej. Pierwsza z nich to `simpleblog_register_menus()` pozwalająca utworzyć dwa menu przy użyciu funkcji `register_nav_menus()`. Druga to funkcja `simpleblog_register_sidebars()` tworząca obszar widżetów w prawej kolumnie. I w końcu trzecia funkcja to `simpleblog_load_scripts()` ładująca skrypty JavaScript. Wszystkie te funkcje są wymienione w funkcji `simpleblog_theme_setup()` podłączonej do haka `after_setup_theme`.



Nie jest to skomplikowane, prawda? Może jednak takie być, gdy dodamy opcje motywu i różne inne rzeczy. Więcej na ten temat dowiesz się z rozdziału 6., w którym będziemy zajmować się bardziej zaawansowanymi zagadnieniami dotyczącymi motywów. Na razie wystarczy nam to, co mamy.

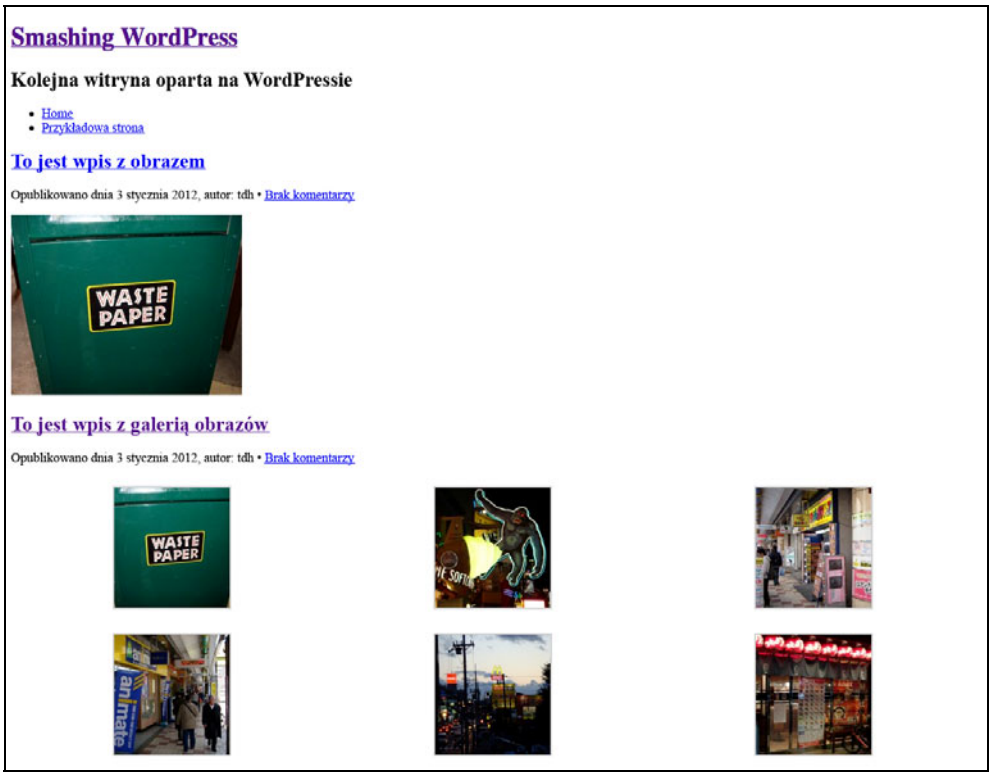
Nasz prosty motyw jest już gotowy. Pozostało jeszcze tylko napisać arkusze stylów, ponieważ bez odpowiedniego kodu w pliku *style.css* nie wygląda on zbyt zachęcająco, co bez wątpienia widać na rysunku 4.7.

# PLIKI SZABLONOWE

Motyw składa się z plików szablonowych, przynajmniej jednego arkusza stylów i ewentualnie dodatkowych plików, takich jak *functions.php*. Czasami powstają różne nieporozumienia, ponieważ niektórzy **szablonami** nazywają motywy. Jest to błąd. Motyw to folder (który musi znajdować się w folderze *wp-content/themes/*) zawierający różne pliki, z których niektóre są szablonami.

Plików szablonowych można używać do różnych celów. Wszystko zależy od tego, co konkretnie się w nich umieści. Można w nich dowolnie modyfikować pętlę, a nawet ją całkiem usunąć, można używać tagów szablonowych, aby uzyskać określone efekty.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 4.7. Możesz nie uwierzyć, ale temu motywowi potrzeba tylko trochę arkuszy stylów, aby zaczął dobrze wyglądać



Analogicznie za pomocą tagów warunkowych można sprawić, aby strony były bardziej dynamiczne. A jeśli chcesz naprawdę zamieszać w systemie, możesz użyć wtyczek rozszerzających jego funkcjonalność poza standard.

Jest kilka rzeczy, które warto wiedzieć o plikach szablonowych. Przede wszystkim należy wiedzieć, które z nich są niezbędne do działania podstawowych funkcji systemu. Po drugie: trzeba wiedzieć, kiedy każdy z tych plików jest używany. I w końcu po trzecie: trzeba wiedzieć, których szablonów należy użyć, aby zrealizować określony cel.

Przypuśćmy, że chcemy, aby strona główna wyglądała tak samo jak strony *index.php*, tylko dodatkowo zawierała jeszcze jakiś obraz lub wiadomość powitalną. Czy w takim przypadku obok pliku *index.php* należy utworzyć plik *home.php*? O wiele wygodniej byłoby po prostu użyć w pliku *index.php* tagu warunkowego `is_home()`, aby wyświetlić dodatki na stronie głównej. To samo dotyczy pojedynczych wpisów. Jeśli nie różnią się znacząco od widoku strony głównej, można użyć tagu `is_single()`, aby wprowadzić zmiany charakterystyczne dla pojedynczych wpisów (np. brak łącza w tytule wpisu).

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## KIEDY UŻYWANE SĄ POSZCZEGÓLNE PLIKI SZABLONOWE

Poniżej znajduje się opis zasad używania różnych plików szablonowych.

- Strona główna i frontowa: pierwszeństwo ma plik *front-page.php*. Jeśli jako strona główna ustawiona jest strona statyczna, to użyta zostanie ta ustawiona strona, a następnie *home.php* i *index.php*.
- Pojedyncze wpisy: *single-X.php*, gdzie *X* to typ wpisu (np. *single-book.php*, jeśli typ wpisu to książka), a następnie *single.php* i *index.php*.
- Strony: pierwszeństwo ma wybrany szablon strony, następnie plik *page-X.php*, gdzie *X* to uproszczona nazwa strony, plik *page-Y.php*, gdzie *Y* to identyfikator strony, później *page.php* i *index.php*.
- Archiwa kategorii: *category-X.php*, gdzie *X* to uproszczona nazwa kategorii, następnie *category-Y.php*, gdzie *Y* to identyfikator kategorii, później *category.php*, *archive.php* i *index.php*.
- Archiwa tagów: *tag-X.php*, gdzie *X* to uproszczona nazwa tagu, następnie *tag-Y.php*, gdzie *Y* to identyfikator tagu, później *tag.php*, *archive.php* i *index.php*.
- Archiwa własnych taksonomii: *taxonomy-X-Y.php*, gdzie *X* to uproszczona nazwa taksonomii, a *Y* to uproszczona nazwa terminu (np. w przypadku *taxonomy-person-waldo.php* mamy termin *waldo* w taksonomii *person*), następnie *taxonomy-X.php*, gdzie *X* to uproszczona nazwa taksonomii, a później *taxonomy.php*, *archive.php* i *index.php*.
- Archiwa typów wpisów użytkownika: *archive-X.php*, gdzie *X* to uproszczona nazwa typu wpisu użytkownika, następnie *archive.php* i *index.php*.
- Archiwa autorów: *author-X.php*, gdzie *X* to uproszczona nazwa, następnie *author-Y.php*, gdzie *Y* to identyfikator, a potem *author.php*, *archive.php* i *index.php*.
- Archiwa dat: *date.php*, *archive.php* i *index.php*.
- Wyniki wyszukiwania: *search.php* i *index.php*.
- 404 (nie znaleziono): *404.php* i *index.php*.
- Załączniki: według typów MIME, tzn. *image.php* dla obrazów, *video.php* dla filmów, *text.php* itd. Następnie *attachment.php*, *single-attachment.php*, *single.php* i *index.php*.



Pamiętaj, że motyw nie musi zawierać wszystkich możliwych plików szablonowych. Należy tworzyć tylko te, które są naprawdę potrzebne, ponieważ gdy jest ich za dużo, to powstaje bałagan i trudno jest cokolwiek zmieniać w projekcie.

Zastosowanie niektórych szablonów może być niejasne. Weźmy na przykład szablon *category.php*. Domyślnie wyświetla on listę wpisów należących do określonej kategorii. Jeśli jednak zostanie utworzony szablon *category-X.php*, gdzie *X* jest identyfikatorem kategorii, to zostanie użyty ten szablon zamiast *category.php*. To samo dotyczy tagów — plik *tag-X.php* ma pierwszeństwo przed plikiem *tag.php* — i uproszczonych nazw tagów i kategorii, tzn. kategoria *Małpy* o uproszczonej nazwie `malpy` i identyfikatorze 42 może być zaprezentowana w plikach *category-malpy.php* i *category-42.php*.

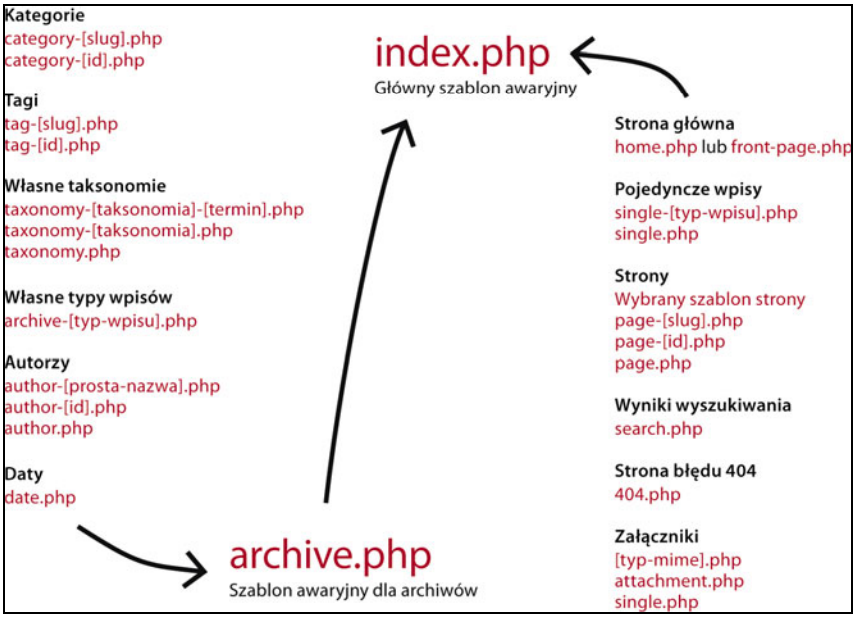
ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Mówiąc krótko: WordPress najpierw szuka najbardziej szczegółowych szablonów, a gdy ich nie znajduje, szuka innych najlepiej pasujących, aż w końcu jakiś znajdzie.

Pamiętaj, że plik *page.php* i szablony stron to dwie różne rzeczy. Zapewne zauważyłeś, że podczas tworzenia strony w WordPressie można wybrać szablon. Szablony te nie są kontrolowane w pliku *page.php*, lecz są osobnymi plikami zawierającymi nagłówek podobny do tego, który znajduje się w pliku *style.css*. Szablony stron bywają bardzo przydatne i dlatego jeszcze do nich wrócimy w dalszej części książki.

## HIERARCHIA SZABLONÓW

Wiesz już, jakie pliki szablonowe są dostępne, musisz zatem też wiedzieć, kiedy każdy z nich jest ładowany i co się dzieje, gdy nie ma w ogóle żadnego szablonu. Jak wiesz, plik *index.php* jest awaryjnym szablonem dla wszystkiego i można sobie poradzić, używając tylko jego. Na rysunku 4.8 przedstawiona jest hierarchia szablonów WordPressa. Jeśli potrzebujesz kompletnych informacji z opisem tagów warunkowych, zajrzyj na stronę *http://codex.wordpress.org/Template_Hierarchy*.




Rysunek 4.8. Hierarchia plików szablonowych

## SZABLONY STRON

Strony w WordPressie standardowo służą do przedstawiania statycznej treści, która jest mniej zależna od czasu niż typowy wpis na blogu (który również zwykle jest statyczny). Strony mogą mieć podstrony i są zazwyczaj używane do prezentowania informacji o witrynie, danych kontaktowych itp. Jeśli jednak chcesz, możesz ich używać do wielu innych celów. Po pierwsze: szablonowi *page.php* możesz nadać styl odpowiadający rodzajowi treści



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

prezentowanej na stronach (zamiast pozostać domyślnie przy takim samym wyglądzie jak wygląd wpisów). Po drugie: możesz tworzyć szablony stron, które można wybierać przy tworzeniu stron w panelu administracyjnym.

Szablony stron są w zasadzie normalnymi plikami szablonowymi, z tym wyjątkiem, że na ich początku musi znajdować się krótki kod pozwalający systemowi je zidentyfikować (jest to mechanizm podobny do używanego w pliku *style.css*).

Na początku pliku szablonu strony powinien znajdować się poniższy komentarz. Sam plik może mieć dowolną nazwę, ale musi mieć rozszerzenie *.php*:

```
<?php
/*
Template Name: Mój szablon strony
*/
?>
```

Plik zawierający ten komentarz mógłby się nazywać np. *mypagetemplate.php*, chociaż lepiej jest plikom tym nadawać znaczące nazwy typu *pagetemplate-[coś].php*. Dodatkowo mimo że nie ma ścisłych zasad dotyczących nazw plików szablonów, należy unikać nazw typu *page-[coś].php*, ponieważ WordPress może szukać szablonu *page-[uproszczona-nazwa].php*, co może spowodować konflikt.



Gdy umieścisz powyższy fragment kodu na początku pliku, a pod nim dowolny kod, szablon ten zostanie wyświetlony na liście szablonów do wyboru w panelu administracyjnym w sekcji tworzenia nowej strony. Wystarczy go wybrać i zapisać stronę.

Szablony stron często wykorzystywane są do tworzenia stron archiwalnych. Powiedzmy, że chcemy utworzyć przy użyciu tagów szablonowych stronę wyświetlającą autorów, kategorie, tagi i 50 najnowszych wpisów. Nie da się tego zrobić za pomocą normalnej strony, ponieważ nie da się wstawiać tagów szablonowych do wpisów poprzez panel administracyjny (przynajmniej nie bez jakiejś wtyczki). Do tego potrzebny jest szablon strony. Poniżej znajduje się kod szablonu strony wyświetlający archiwum bloga:

```
<?php
/*
Template Name: Archiwum
*/

get_header(); ?>

<div id="main-container">
    <section id="content-container">

    <?php
        // Początek pętli
        while ( have_posts() ) : the_post();
    ?>

        <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            <header>
                <h1 class="entry-title">
                    <?php the_title(); ?>
                </h1>
            </header>
            <?php
                // Treść
                the_content();
            ?>
            <h2>Przeglądaj wg miesiąca</h2>
            <ul>
            <?php
                // Argumenty
                $args = array(
                     'type' => 'monthly'
                );

                // Archiwum
                wp_get_archives( $args );
            ?>
            </ul>
            <h2>Przeglądaj wg kategorii</h2>
            <ul>
            <?php
                // Argumenty
                $args = array(
                     'title_li' => ''
                );

                // Kategorie
                wp_list_categories( $args ); ?>
            </ul>
            <h2>Przeglądaj wg tagów</h2>
            <?php
                // Argumenty
                $args = array(
                     'smallest' => 8,
                     'largest' => 28,
                     'number' => 0,
                     'orderby' => 'name',
                     'order' => 'ASC'
                );

                // Chmura tagów
                wp_tag_cloud( $args ); ?>
        </article>

    <?php
        // Koniec pętli
        endwhile;
    ?>

    </section> <!-- #main-container - koniec -->
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php get_sidebar(); ?>

</div>

<?php get_footer(); ?>
```

Innym częstym zastosowaniem dla szablonów stron jest wyświetlanie treści przy użyciu specjalnych pętli. Można na przykład skopiować zawartość pliku *index.php*, wkleić ją do pliku szablonu strony zawierającego na początku specjalny komentarz, a następnie zmodyfikować pętlę w dowolny sposób za pomocą funkcji `query_posts()`. Można także utworzyć stronę zawierającą kilka pętli, jak również kod PHP w ogóle nie związany z WordPressem.

Wykorzystanie szablonów stron do własnych celów jest wielkim krokiem w kierunku utworzenia witryny dokładnie dostosowanej do własnych potrzeb.

# KORZYSTANIE Z PLIKU FUNCTIONS.PHP

Do tej pory niewiele pisałem na temat pliku *functions.php*. Jest on dość tajemniczy i większość osób, gdy tylko do niego zajrzy, od razu z przerażeniem go zamyka. Nie wszystkie motywy zawierają ten plik, a te, w których się on znajduje, zazwyczaj mają własne widżety i własną stronę opcji w panelu administracyjnym. Plik ten jest niezbędny, gdy chce się dodać widżety i panel administracyjny.



Do czego w takim razie konkretnie służy plik *functions.php*? W zasadzie można powiedzieć, że może służyć do wszystkiego, co sobie wymyślimy, ponieważ działa on jak wtyczka wywoływana w czasie inicjacji WordPressa — zarówno gdy użytkownik przegląda strony, jak i wtedy, gdy administrator wchodzi do panelu administracyjnego. Dzięki temu za pomocą pliku *functions.php* można dodawać opcje administracyjne.

Widżetami zajmiemy się nieco dalej, natomiast teraz chciałbym, abyś dodał poniższą prostą funkcję do swojego pliku *functions.php*:

```
<?php
    function hellomate() {
        echo 'Cześć, jak leci?';
    }
?>
```

Jeśli umieścisz tę funkcję w pliku *functions.php*, a następnie wywołasz ją w dowolnym miejscu w swoim motywie, zostanie tam wyświetlony napis **Cześć, jak leci?**. Funkcje takie wywołuje się w sposób typowy dla języka PHP:

```
<?php hellomate(); ?>
```

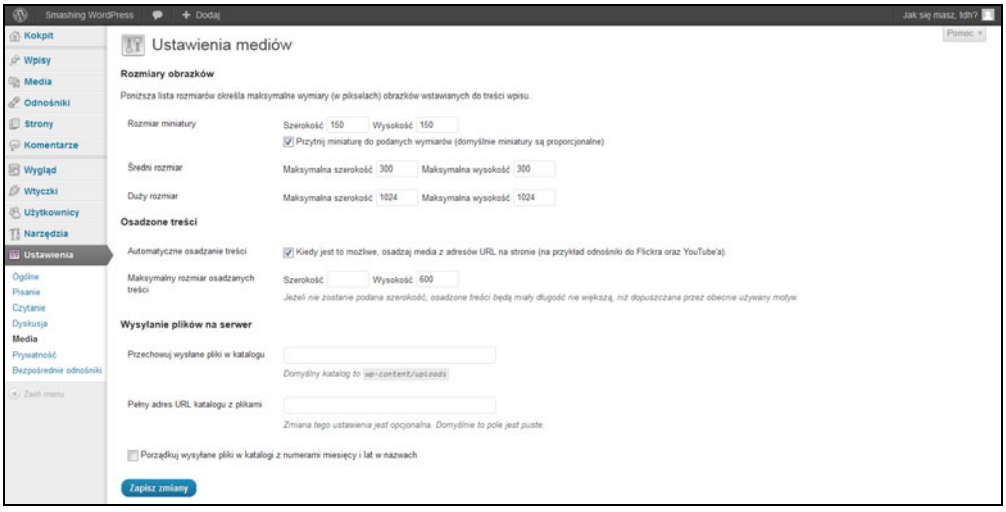
To wywołanie spowodowałoby wyświetlenie tekstu na stronie. Może nie jest to szczyt sztuki programistycznej, ale pokazuje, jakie są możliwości pliku *functions.php*. Jeśli masz jakieś fragmenty kodu, których często używasz, i chcesz mieć do nich łatwy dostęp, to właśnie znalazłeś rozwiązanie.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Wiele motywów ma strony z opcjami, na których użytkownik może ustawić kolorystykę witryny, style pisma, zmienić nagłówek itp. Wszystko to jest możliwe dzięki plikowi *functions.php*. Bardziej szczegółowy opis tego pliku znajduje się w rozdziale 6. i dalszych.

## USTAWIANIE DOMYŚLNEJ SZEROKOŚCI

Często zapominanym ustawieniem jest szerokość treści. Parametr ten, mający postać krótkiego fragmentu kodu, informuje WordPressa o tym, jaka jest maksymalna dozwolona szerokość treści w motywie, dzięki czemu np. wszystkie obrazy będą odpowiednio przeskalowywane. Oczywiście w panelu administracyjnym WordPressa też są ustawienia mediów i można za ich pomocą kontrolować rozmiary obrazów (rysunek 4.9), ale użytkownik może zapomnieć je zmienić.




Rysunek 4.9. Ustawienia mediów w panelu administracyjnym WordPressa nie są jedynym sposobem na kontrolowanie szerokości obrazów

Do ustawiania szerokości treści służy zmienna `$content_width`, która określa szerokość dużych obrazów. Pamiętaj, że gdy wysyłasz obraz do WordPressa, w istocie otrzymujesz cztery obrazy: te, które są wyświetlone w panelu administracyjnym, i oryginał. Dzięki zmiennej `$content_width` duży rozmiar obrazu będzie idealnie pasował do Twojego motywu.

Zmienną tę bardzo łatwo się dodaje. Wystarczy umieścić poniższy fragment kodu w pliku *functions.php* (oczywiście w bloku kodu PHP):

```
$content_width = 580;
```

W tym przykładzie została ustawiona szerokość 580 pikseli. Jeśli w Twoim motywie ma być inna, po prostu zmień tę liczbę na inną.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## DODAWANIE ELEMENTÓW PROMOCYJNYCH ZA POMOCĄ PLIKU FUNCTIONS.PHP

W niektórych blogach i witrynach internetowych na końcu każdego wpisu wyświetlane są elementy promocyjne, mające np. na celu zachęcenie czytelników do subskrypcji kanału RSS. Elementy takie można z łatwością wstawić bezpośrednio w plikach szablonowych, ale można także wykorzystać czarodziejskie właściwości pliku *functions.php*.

Załóżmy, że chcemy zachęcić czytelników do subskrypcji naszego kanału RSS. W tym celu na stronę wstawimy element `div` przypisany do klasy `promotion`, a w nim umieścimy nagłówek `h4` i tekst. Dzięki CSS możemy z tymi elementami zrobić praktycznie wszystko, np. to:

```
div.promotion {
    background: #eee;
    border: 1px solid #bbb;
    padding: 10px; }

div.promotion h4 {
    color: #f00;
    font-size: 14px;
    margin: 0 0 5px 0;
    padding: 0; }

div.promotion p {
    font-size: 12px;
    color: #444;
    margin-bottom: 0; }
```



To powinno wyglądać nieźle. Gdybyś jednak chciał osiągnąć oszałamiający efekt, zapewne przydałoby się dodać w tle elementu `div` jakąś efektowną grafikę przedstawiającą kanał RSS. Na razie jednak nie to jest dla nas najważniejsze. Poniżej pokazany jest cały kod HTML, jaki chcemy wstawiać za każdym naszym wspaniałym wpisem:

```
<div class="promotion">
    <h4>Nie pozwól, żeby cokolwiek Cię ominęło!</h4>
    <p>
        Nasz <a href="http://tdh.me/feed/">kanał RSS</a> na pewno Ci w tym pomoże!
    </p>
</div>
```

Jak w takim razie teraz sprawić, aby to coś znalazło się na stronach bez modyfikowania plików szablonów? Nic prostszego. Wystarczy użyć w pliku *functions.php* haka `the_content`. Hak ten wyświetla treść, za którą chcemy umieścić nasz kod. Oto kod, który należy wpisać w pliku *functions.php*:

```
function Promotion($content) {
        if( !is_feed() && !is_home() ) {
            $content.= '<div class="promotion">';
            $content.= '<h4>Nie pozwól, żeby cokolwiek Cię ominęło!</h4>';
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            $content.= '<p>Nasz <a href=/"http://tdh.me/feed/">kanał RSS</a> na
            pewno Ci w tym pomoże!</p>';
            $content.= '</div>';
        }
        return $content;
}
add_filter ( 'the_content', 'Promotion' );
```

W kodzie tym tworzona jest funkcja o nazwie `Promotion`, w której tworzony jest nasz kod HTML. Oczywiście cały kod HTML można by było zapisać w jednym długim wierszu, zamiast stosować kilkukrotnie instrukcję przypisania do zmiennej `$content`, ale tak jest prościej i bardziej przejrzyście. Funkcja zwraca zawartość zmiennej `$content`. Następnie za pomocą funkcji `add_filter` wywołujemy naszą funkcję po wykonaniu funkcji `the_content`.

Gotowe. Jeśli użytkownik otworzy stronę, która nie jest główną ani kanałem RSS (w razie potrzeby możesz w instrukcji `if` dodać jeszcze inne warunki), zostanie wyświetlona ramka promocyjna zachęcająca do subskrypcji kanału RSS.

Czemu miałbyś posługiwać się tą techniką, skoro łatwiej byłoby dodać odpowiedni kod HTML w pliku szablonu? Jedynym powodem jest to, że metoda ta jest całkowicie niezależna od motywu, tzn. wystarczy skopiować ten kod, wkleić go do pliku *functions.php* dowolnego motywu, dodać odpowiednie style w pliku *style.css* i gotowe. Tworząc zestaw funkcji do wstawiania na strony najczęściej używanych treści i podłączając je do haków, gdy to tylko możliwe, sprawisz, że Twój motyw będzie łatwiejszy w użytkowaniu.



Jak widzisz, plik *functions.php* jest bardzo przydatny i z pewnością pełni o wiele ważniejszą funkcję, niż tylko bycie miejscem do deklarowania widżetów, które mają praktycznie wszystkie nowe motywy. Przy okazji: tworzenie widżetów to jedno z najpopularniejszych zastosowań pliku *functions.php*, a więc teraz przyszedł czas przyjrzeć się także im.

# WIDŻETY — CZYM SĄ I DO CZEGO SŁUŻĄ

Widżety to elementy, które administrator może umieszczać w wybranych miejscach, przeciągając je tam myszą. W widżecie może znajdować się dowolna treść, od zwykłego tekstu po listę wpisów z wybranej kategorii, najnowsze komentarze czy najświeższe wiadomości z kanału RSS. Te rzeczy można zrobić za pomocą podstawowych widżetów WordPressa. Jeśli jednak dodamy do tego możliwość tworzenia wtyczek tworzących widżety, to nasze możliwości stają się ogromne.

Dla administratora witryny widżety są bardzo wygodnym rozwiązaniem, ponieważ sposób ich obsługi jest bardzo łatwy, a możliwości są bardzo duże. W dalszych rozdziałach dowiesz się, jak wykorzystać obszary na widżety do zadań innych niż tylko wyświetlanie bałaganu na paskach bocznych, w stylu podrzędnych blogów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## DEKLAROWANIE OBSZARÓW NA WIDŻETY

Przygotowanie motywu pod kątem widżetów jest bardzo łatwe. Polega to na zadeklarowaniu obszarów widżetów w pliku *functions.php*, a następnie dodaniu odpowiedniego kodu do plików szablonowych (zazwyczaj *sidebar.php*) w miejscach, w których te widżety mają być wyświetlane.

Najprostszym sposobem jest utworzenie domyślnego paska bocznego w pliku *functions.php*:

```
register_sidebar();
```

Powyższy kod zadziała, chociaż poprawny sposób rejestrowania paska bocznego polega na podłączeniu się do haka `widgets_init`. W tym celu trzeba napisać funkcję zawierającą wywołanie funkcji `register_sidebar()` i przekazać ją do haka:

```
add_action( 'widgets_init', 'smashing_register_sidebars' );
function smashing_register_sidebars() {
    // Rejestracja pasków bocznych
    register_sidebar()
}
```

Powyższy kod podłącza funkcję `smashing_register_sidebars()` do haka `widgets_init`, w wyniku czego zostanie wykonana funkcja `register_sidebar()`.



Następnie w pliku *sidebar.php* należy dodać poniższy kod w miejscu, w którym mają być wyświetlane widżety. Całość powinna znajdować się w elemencie `ul`:

```
<ul id="sidebar">
<?php dynamic_sidebar(); ?>
</ul>
```

Bardziej szczegółowy opis podstawowej zawartości pliku *sidebar.php* znajduje się w podrozdziale „Prawa kolumna: plik sidebar.php”.

## DEKLAROWANIE WIELU OBSZARÓW NA WIDŻETY

W niektórych motywach jest więcej obszarów widżetów niż jeden. Obszary te również są deklarowane w pliku *functions.php*, ale w nieco inny sposób. Poniżej znajduje się kod deklarujący dwa obszary widżetów w pasku bocznym, jeden w nagłówku i jeden w stopce:

```
add_action( 'widgets_init', 'smashing_register_sidebars' );

function smashing_register_sidebars() {
    // Pierwszy pasek boczny
    register_sidebar( array(
        'name' => 'First sidebar',
        'id' => 'first-sidebar',
        'before_widget' => '<li id="%1$s" class="widget-container %2$s">',
        'after_widget' => '</li>',
        'before_title' => '<h3 class="widget-title">',
        'after_title' => '</h3>'
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    ) );
    // Drugi pasek boczny
    register_sidebar( array(
        'name' => 'Second sidebar',
        'id' => 'second-sidebar',
        'before_widget' => '<li id="%1$s" class="widget-container %2$s">',
        'after_widget' => '</li>',
        'before_title' => '<h3 class="widget-title">',
        'after_title' => '</h3>'
    ) );
    // Pasek boczny w nagłówku
    register_sidebar( array(
        'name' => 'Header widget area',
        'id' => 'header-sidebar',
        'before_widget' => '<li id="%1$s" class="widget-container %2$s">',
        'after_widget' => '</li>',
        'before_title' => '<h3 class="widget-title">',
        'after_title' => '</h3>'
    ) );
    // Pasek boczny w stopce
    register_sidebar( array(
        'name' => 'Footer widget area',
        'id' => 'footer-sidebar',
        'before_widget' => '<li id="%1$s" class="widget-container %2$s">',
        'after_widget' => '</li>',
        'before_title' => '<h3 class="widget-title">',
        'after_title' => '</h3>'
    ) );
}
```



Ten kod mówi sam za siebie. Są to prawie takie same deklaracje obszarów na widżety jak poprzednio, tylko każdy z nich ma określoną nazwę. Te dodatkowe informacje wykorzystamy w plikach szablonowych. Nie jest to nic skomplikowanego, po prostu do definicji obszaru na widżety należy dodać identyfikator (preferowany) tego obszaru.

```
<?php dynamic_sidebar( 'id-obszaru-widżetów' ); ?>
```

W związku z tym obszar na widżety w stopce zostałby zdefiniowany następująco:

```
<?php dynamic_sidebar( 'footer-sidebar' ); ?>
```

Proste. Oczywiście wszystko, co dotyczy pojedynczego obszaru widżetów, ma zastosowanie także wtedy, gdy tych obszarów jest więcej, a więc możesz w nich umieszczać, co tylko chcesz.

## DOSTOSOWYWANIE WIDŻETÓW

Czasami nie odpowiada nam domyślny sposób wyświetlania widżetów i chcielibyśmy to zmienić, np. umieszczając je w dodatkowym elemencie `div`. W tym celu należy zdefiniować odpowiednie ustawienia w tablicy:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php
    register_sidebar( array(
         'before_widget' => '',
         'after_widget' => '',
         'before_title' => '',
         'after_title' => '',
    ) );
?>
```

Kod HTML należy wpisać w pojedynczych cudzysłowach znajdujących się na końcu każdego wiersza.

W ramach przykładu umieścimy nasz widżet w elemencie `div` przypisanym do klasy `customwidget`, a jego tytuł zapiszemy w elemencie `div.customtitle`:

```
<?php
    register_sidebar( array(
         'before_widget' => '<div class="customwidget">',
         'after_widget' => '</div>',
         'before_title' => '<div class="customtitle"',
         'after_title' => '</div>',
    ) );
?>
```



Z możliwości tych należy korzystać ostrożnie. Większość motywów i wtyczek tworzących widżety jest dostosowana do list nieuporządkowanych.

Teraz gdy już dogłębnie poznałeś tajniki plików szablonowych i wiesz, jak tworzyć obszary widżetów, możesz zająć się komentarzami. Tak, chodzi o plik *comments.php*, o którym była mowa we wcześniejszej części rozdziału. Komentarze nie tylko powinny znajdować się na stronie, lecz również powinny dobrze wyglądać.

# UPIĘKSZANIE KOMENTARZY

Nie we wszystkich witrynach potrzebna jest funkcja komentarzy, ale istnieje duża szansa, że w większości witryn, które będziesz tworzyć w oparciu o WordPressa, będzie ona jednak potrzebna. Zazwyczaj czytelnicy blogów mają możliwość komentowania wpisów. To samo dotyczy witryn różnych czasopism, od gazet codziennych po kolorowe magazyny. Komentarze są dobrym sposobem na zaangażowanie czytelników i mimo że na różnych stronach możliwość komentowania może być włączona z różnych powodów oraz mogą obowiązywać różne zasady dodawania komentarzy, podstawowy mechanizm ich działania jest zawsze taki sam.

Dla projektanta motywów do WordPressa komentarze mogą być problemem, ponieważ trudno jest sprawić, aby dobrze wyglądały. Ich kod nie jest jednak skomplikowany, a jeśli spodoba Ci się domyślny wygląd, to możesz nawet zrezygnować z tworzenia pliku *comments.php*.

Poniżej znajduje się lista najważniejszych czynników, jakie należy wziąć pod uwagę przy projektowaniu sekcji komentarzy na stronie:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Wyraźnie oddziel sekcję komentarzy od pozostałej treści, aby nie pomyliły się użytkownikom z właściwą treścią.
- Komentarze powinny być łatwe do przeczytania, tak jak pozostała treść witryny.
- Odpowiednie odstępy między poszczególnymi komentarzami i naprzemienne kolory lub linie rozdzielające pomagają wizualnie oddzielić komentarze od siebie. Każdy sposób na uzyskanie tego efektu jest dobry.
- Imię autora komentarza powinno być dobrze widoczne.
- Formularz dodawania komentarzy powinien być prosty w użyciu, poprawnie skonstruowany oraz zawierać odpowiednio duży i czytelny tekst. Pomyśl: jeśli chcesz, aby użytkownicy pisali długie i wartościowe komentarze, powinieneś zapewnić im jak najlepsze warunki do pracy.

Można wymienić jeszcze kilka innych, choć mniej ważnych zasad:

- Co zrobić z zasadami dodawania komentarzy? Dodać łącze do strony, gdzie są wypisane, czy wypisać je drobnym drukiem obok przycisku zatwierdzania komentarza?
- Czy zezwolić na używanie elementów HTML? Jeśli tak, to jakich?
- Czy komentarze mają być moderowane? Jeśli tak, należy wyświetlić dobrze widoczne powiadomienie, że komentarz oczekuje na moderację.
- Czy aby dodać komentarz, trzeba się zarejestrować lub zalogować? Jeśli tak, to proces rejestracji i logowania powinien być jak najprostszy.



Dobrze przemyśl wszystkie kwestie związane z działaniem funkcji komentarzy, a nie będziesz mieć żadnych problemów. Gdy to zrobisz, także projektowanie i późniejsze modyfikowanie komentarzy będzie łatwiejsze.

## PODZIAŁ KOMENTARZY NA WĄTKI

Możliwość wyświetlania komentarzy w wątkach wprowadzono w WordPressie 2.7. Funkcja ta musi zostać włączona na stronie *Ustawienia/Dyskusja*. Można z niej korzystać w każdym motywie, w którym do wyświetlania komentarzy użyty jest tag szablonowy `wp_list_comments()`.

Gdy opcja zakorzeniania komentarzy jest włączona, obok każdego komentarza pojawia się odnośnik *Odpowiedz*. Jego kliknięcie spowoduje dodanie łącza *Anuluj pisanie odpowiedzi*. Wszystko to jest standardową funkcją WordPressa i nie trzeba nic dodatkowo robić, aby działało.

Należy jednak zastanowić się nad następującymi kwestiami:

- Na ile poziomów będą mogły być zagnieżdżane wątki? Jest to ustawienie dostępne w sekcji administracyjnej i powinno być obsługiwane w motywie.
- Jak ma wyglądać łącze *Odpowiedz*?
- Jak ma wyglądać łącze *Anuluj pisanie odpowiedzi*?

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Odpowiedzi do komentarzy są umieszczane w tym samym elemencie `li` w elemencie `ul` przypisanym do klasy `children`. Ogólnie hierarchia komentarzy wygląda następująco (część nieistotnego kodu usunąłem):

```
<li>
    [Komentarz pierwszego poziomu]
    <ul class="children">
        <li>
             [Pierwszy poziom odpowiedzi]
            <ul class="children">
                <li>
                     [Drugi poziom odpowiedzi]
                </li>
            </ul>
        </li>
    </ul>
</li>
<li>
     [Następny komentarz najwyższego poziomu]
</li>
```

Liczbę dozwolonych elementów `ul` przypisanych do klasy `children` ustawia się w panelu administracyjnym za pomocą opcji zakorzeniania komentarzy. Domyślnie jest ona ustawiona na 5, a więc tyle przynajmniej powinien obsługiwać każdy motyw. Cała koncepcja zakorzeniania komentarzy opiera się na hierarchii, a więc dopełnienie lub margines klasy `children` powinno się ustawić na około 10 pikseli. Konkretna wartość zależy od motywu, ale należy pamiętać, że każda odpowiedź powinna być trochę wcięta względem komentarza, do którego się odnosi.



Z łączem *Odpowiedz* jest łatwiej. Jest ono tworzone przez element `a` z klasą `comment-reply-link`, a więc wystarczy tylko napisać dowolny arkusz stylów dla tej klasy. Poniższa reguła na przykład zamienia nasze łącze w element spływający w prawo i ustawia w nim rozmiar pisma na 12 pikseli:

```
a.comment-reply-link {
    float:right;
    font-size: 12px; }
```

To samo dotyczy łącza *Anuluj pisanie odpowiedzi* wyświetlanego zaraz pod nagłówkiem *Skomentuj* w sekcji *Odpowiedz*. Tak naprawdę wszystko zależy od ustawień w pliku *comments.php*, ale zazwyczaj wygląda to tak, jak opisałem. Łącze jest elementem `a` z identyfikatorem `cancel-comment-reply`. Można je np. pogrubić:

```
a#cancel-comment-reply { font-weight: bold; }
```

Zakorzenianie komentarzy jest doskonałym sposobem na uporządkowanie dłuższych dyskusji, a więc warto rozważyć możliwość włączenia tej opcji, zwłaszcza gdy planuje się publikować kontrowersyjne materiały.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### WYRÓŻNIANIE AUTORA WPISU

Dobrym pomysłem jest też wyróżnienie komentarzy napisanych przez autora wpisu, zwłaszcza w witrynach edukacyjnych. Czytelnicy mogą zadawać pytania i wówczas dobrze by było, gdyby komentarze autora wpisu były łatwo rozpoznawalne.

Komentarze znajdują się w elementach `li` listy uporządkowanej (`ol`). Można to wykorzystać podobnie jak fakt, że funkcja `wp_list_comments()` do każdego elementu `li` dodaje różne klasy. Wśród tych klas znajduje się klasa `bypostauthor`, która jest dodawana tylko do komentarzy napisanych przez autora wpisu. Oczywiście aby autor wpisu był rozpoznany, musi przed dodaniem komentarza zalogować się w systemie.

Poniżej znajduje się przykładowa reguła CSS zmieniająca kolor tła komentarzy autora wpisu na żółty:

```
li.bypostauthor { background: yellow; }
```

Możesz też wymyślić coś bardziej wyszukanego, ale zmiana koloru tła jest bardzo dobrym pomysłem, podobnie jak powiększenie czcionki i zmiana jej koloru. Jeśli chcesz, możesz pójść na całość, wykorzystując fakt, że cały komentarz znajduje się w elemencie `li.bypostauthor`. Możesz na przykład zmienić sposób wyświetlania awatara (potrzebny do tego selektor to `img.avatar`), zmodyfikować metadane komentarza (`div.comment-author` i `div.comment-meta`) albo zmodyfikować wygląd samego tekstu komentarza. Na przykład dla zabawy poniżej zmieniamy rozmiar czcionki komentarzy autora wpisu na 18 pikseli i pozostawiamy żółte tło:

```
li.bypostauthor { background: yellow; }
li.bypostauthor div.comment-body p { font-size: 18px; }
```

Korzystaj z możliwości wyróżniania komentarzy autora wpisu ostrożnie. Nie zawsze muszą one być bardzo widoczne, chociaż mniej rzucające się w oczy wyróżnienie nigdy nie zaszkodzi.



## DODAWANIE WŁASNYCH PÓL

Własne pola otwierają przed projektantami wtyczek i motywów jeszcze jedno pole możliwości. Pola pozwalają na przechowywanie informacji w bazie danych, co można wykorzystać na wiele różnych sposobów. Szczegółowy opis działania własnych pól znajduje się w podrozdziale „Używanie własnych pól” w rozdziale 6. W tym podrozdziale pokażę Ci tylko, do czego można je wykorzystać podczas projektowania motywu.

### NAJCZĘSTSZE ZASTOSOWANIE WŁASNYCH PÓL

Początkowo własne pola miały służyć do przechowywania metadanych wpisów i taki ich opis nadal można znaleźć w dokumentacji WordPressa. Jednak obecnie pola te są wykorzystywane do całkiem innych celów. Najczęściej używa się ich do dodawania obrazów w wybranych miejscach we wpisach i na listach wpisów oraz do tworzenia tzw. **nagłówków w stylu czasopism**. Oczywiście możliwości własnych pól nie ograniczają się tylko do wyświetlania

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

atrakcyjnych nagłówków czy miniatur na listach wpisów. Można ich użyć do różnych celów, np. dodawania własnych arkuszy stylów w zależności od wpisu albo tworzenia i identyfikowania serii wpisów (kluczem mogłoby być słowo `Seria`, a wartościami różne nazwy serii), które następnie wyświetlałoby się w specjalnych szablonach ze specjalnie skonstruowaną pętlą.

Za pomocą pól własnych można zmienić nie tylko nagłówek czy obrazy na listach wpisów, ale nawet tło całego dokumentu!

Za pomocą pól własnych można naprawdę dużo zdziałać, jeśli więc musisz posłużyć się czymś innym niż zwykłe tagi szablonowe i funkcje WordPressa, pola te mogą być właściwym wyborem.

### KWESTIA UŻYTECZNOŚCI

W polach własnych najbardziej nie podoba mi się to, że wyglądają jakoś tak nieporządnie. Wystarczy spojrzeć na pole do ich dodawania w panelu administracyjnym. Ten interfejs nie jest tak przyjazny dla użytkownika jak reszta panelu. Nie podobają mi się nazwy **klucz** i **wartość**. Raczej nie zaryzykowałbym przekazania takiego czegoś do użytku klientowi.



To według mnie największa wada pól własnych, bo przecież gdy już utworzymy dane pole, to później nie będziemy mieli problemu, aby np. wybrać je ponownie z listy rozwijanej oraz skopiować i wkleić obraz w polu wartości. A jednak mimo że Tobie może się to wydawać proste, dla klienta wcale nie musi takie być.

Projektując witrynę internetową, należy zawsze mieć na uwadze, czy osoba, która będzie się nią opiekować, będzie umiała sobie z tym poradzić. Pola własne przecież najczęściej są używane do dodawania obrazów do nagłówków i robienia podobnych rzeczy, a to oznacza, że zazwyczaj trzeba znaleźć adres URL obrazu graficznego, a następnie go skopiować i wkleić w polu wartości odpowiedniego klucza. Czy klient sobie z tym poradzi?

Pola własne są świetnym narzędziem, ale dopóki ich obsługa nie zostanie uproszczona, pozostaną tylko niszowym dodatkiem dostępnym wyłącznie lepiej obeznanym technicznie użytkownikom nie bojącym się ręcznego konfigurowania różnych rzeczy. Zapewne należysz do tej grupy, ale czy Twoi klienci (współpracownicy, partnerzy itd.) też? Musisz sam sobie odpowiedzieć na to pytanie. Jeśli nie, to lepiej poszukać innych rozwiązań. Na szczęście istnieje kilka pomocnych wtyczek (dojdziemy do nich później). A w razie potrzeby możesz też utworzyć własne pola do wprowadzania metadanych.

## TWORZENIE MOTYWU BAZOWEGO

Jeśli jesteś projektantem motywów lub aspirujesz do tego miana, powinieneś utworzyć sobie bazowy motyw. Oto kilka powodów, dla których warto to zrobić:

- Oszczędność czasu. Rozpoczynając kolejny projekt, masz już gotowy podstawowy i łatwy w użyciu motyw, od którego możesz zacząć pracę.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Znajomość środowiska pracy. Jeśli na pracę nad jakimś motywem poświęcisz wiele godzin, to przy każdym kolejnym projekcie będzie Ci coraz łatwiej.
- Łatwość aktualizowania. Jeśli swój bazowy motyw będziesz na bieżąco aktualizować, nie będziesz musiał ciągle zmagać się z nowymi funkcjami. Wystarczy nanieść poprawki w bazowym motywie i po wszystkim.
- Łatwość aktualizowania witryn klientów. Jeśli będziesz tworzyć witryny jako motywy potomne swojego bazowego motywu, nie powinieneś mieć problemów z aktualizowaniem serwisów klientów.

Jak się przekonasz w rozdziale 5., motywy potomne to bardzo przydatna rzecz. Jeśli utworzysz solidny motyw bazowy, na którym będziesz opierać wszystkie swoje projekty, znacznie ułatwisz sobie pracę.

Jako bazy można użyć dowolnego motywu, niezależnie od tego, czy będziesz wprost modyfikować ten motyw, aby dostosować go do indywidualnych potrzeb, czy skorzystasz z koncepcji motywów potomnych. Istnieje wiele darmowych motywów, których można używać do dowolnego celu, a więc zarówno do tworzenia witryn osobistych, jak i komercyjnych. Jeśli jeszcze nie masz ulubionego takiego motywu, to w dodatku B znajdziesz kilka propozycji. Zanim zdecydujesz się na użycie któregokolwiek motywu, koniecznie dokładnie przeczytaj jego licencję, ponieważ potrzebujesz takiego, którego można używać w dowolnym celu bez uiszczania opłat przy każdym użyciu. Jeśli na bazowy wybierzesz jakiś płatny motyw, to zapewne ma on także licencję dla programistów, która zezwala na jego wykorzystanie w opisywany sposób. Ale skoro czytasz tę książkę, to myślę, że lepiej na tym wyjdziesz, jeśli poświęcisz trochę czasu na utworzenie własnego motywu bazowego.



Jeśli nie chcesz wykorzystywać żadnego istniejącego motywu, możesz utworzyć własny od podstaw (albo użyć funkcji kopiuj i wklej, tylko z rozwagą). Wówczas wszystko będzie dokładnie tak, jak sobie to wymyśliłeś, ale dojście do tego zajmie Ci więcej czasu, bo wszystko musisz zrobić samemu. Skoro jednak czytasz tę książkę, to mniemam, że możesz być zainteresowany takim rozwiązaniem!

Co w takim razie powinien zawierać taki bazowy motyw? W zasadzie wszystko, czego potrzebujesz w każdym projekcie, i nic więcej. Ostatnie czego Ci potrzeba to rozdęty motyw bazowy o może i ładnym wyglądzie i pasujący może do jednej witryny, ale o wiele za bardzo skomplikowany dla innej. Lepiej jest utworzyć sobie bibliotekę dodatków, które są potrzebne tylko od czasu do czasu, np. fragmentów kodu i plików szablonowych, i używać ich w miarę potrzeb. Twoim celem jest przede wszystkim otrzymanie zgrabnego motywu bez zbędnych funkcji utrudniających jego używanie.

Podsumujmy:

- Zastanów się, czego potrzebujesz, i na podstawie tej analizy swoich potrzeb utwórz bazowy motyw.
- Jeśli to możliwe, użyj istniejącego motywu, aby zaoszczędzić na czasie.
- Uważnie czytaj licencje motywów!

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Powiedzmy, że lubisz dzielić się z innymi i chcesz udostępnić swój znakomity bazowy motyw albo przynajmniej jakąś jego wersję do użytku za darmo. Bardzo dobrze, rozumiesz, na czym polega filozofia otwartego kodu. Mimo to nawet w takim przypadku trzeba pamiętać o pewnych kwestiach!

# PUBLIKOWANIE MOTYWU

Społeczność WordPressa zawsze chętnie przyjmie nowy motyw. W panelu administracyjnym WordPressa jest nawet opcja pozwalająca instalować motywy z oficjalnego zbioru, a więc warto umieścić w nim też swój motyw. Dzięki temu system będzie automatycznie przypominał użytkownikom o konieczności zainstalowania aktualizacji, gdy je udostępnisz.

Motyw przekazany do użytku powinien być w pełni funkcjonalny, mieć poprawny kod HTML i CSS oraz nie powinien być wierną kopią innego motywu. W następnym podrozdziale zamieściłem listę punktów do sprawdzenia przed opublikowaniem motywu. Przeczytaj ją uważnie.

Możesz też zechcieć sprzedać swój motyw. Komercyjne motywy (często z angielska nazywane **premium**) jak najbardziej istnieją, są nawet takie, których licencja przewiduje dodatkową pomoc dla kupującego, jak również inne rozwiązania oparte na licencji GPL, na której udostępniany jest WordPress. Jakie to ma znaczenie dla Ciebie? Musisz pamiętać, że WordPress jest na licencji GPL, a to oznacza, że wszystko, co na nim bazuje, również jest na tej licencji. Sprzedając motywy, stąpasz po grząskim gruncie, powinieneś więc dokładnie przemyśleć sposób licencjonowania swojego produktu. Warto też wiedzieć, że do zbioru na WordPress.org przyjmowane są wyłącznie motywy na licencji zgodnej z GPL, co (szczęśliwie) przyczyniło się do udostępnienia wielu płatnych motywów na tej licencji.



## LISTA PUNKTÓW DO SPRAWDZENIA PRZED PUBLIKACJĄ MOTYWU

Przed publikacją motywu, jak również zanim się go odda do użytku klientowi lub użyje we własnym projekcie, obowiązkowo należy przemyśleć kilka spraw. Oczywiście motyw musi działać, a więc musi zawierać plik *style.css* z podstawowymi informacjami. Ponadto konieczny jest plik *index.php* i inne pliki szablonowe.

To jednak nie wszystko. Zanim opublikujesz motyw, upewnij się, że spełnia on wszystkie wymogi opisane na poniższych listach. Listy te pomogą Ci uniknąć sytuacji, w której opublikujesz motyw i zaraz po tym będziesz musiał go poprawiać.

**Kwestie programistyczne:**

- Czy kod motywu jest poprawny?
- Czy motyw zawiera prawidłową deklarację `DOCTYPE`?
- Czy wywoływane są funkcje `wp_head()` i `wp_footer()` (naprawdę powinny!)?

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Czy w deklaracji nagłówka znajduje się łącze do kanału RSS? Jeśli przeglądarki znajdą ten odnośnik, to po prawej stronie adresu URL w pasku adresu umieszczają ikonę RSS.
- Czy zostały usunięte wszystkie elementy lokalne, np. lokalne odwołania do obrazów, kod odwołujący się do lokalnego repozytorium SVN itd.
- Czy poprawnie używasz JavaScriptu? Pamiętaj, że wiele motywów do WordPressa dołącza różne skrypty JavaScript i istnieje nawet specjalna funkcja służąca do ich dołączania, o nazwie `wp_enqueue_script()`. Więcej szczegółów na ten temat znajduje się w rozdziale 11.
- Czy obszary widżetów działają prawidłowo i czy wyświetlana jest w nich domyślna treść? Jeśli tak, upewnij się, że treść ta jest odpowiednia. Jeśli taka nie jest, to lepiej żeby nic nie wyświetlały.
- Przyjrzyj się obszarom na menu. Czy są w ogóle jakieś i jak są obsługiwane? Funkcja menu w WordPressie jest bardzo pomocna dla użytkowników, a więc należy jej używać, kiedy tylko się da!
- Czy utworzyłeś łącza do edycji wpisów, stron i komentarzy widoczne tylko dla administratora? Jest to bardzo przydatna rzecz.
- Czy Gravatry działają prawidłowo?
- Czy napisałeś arkusze stylów dla zakorzenionych komentarzy, nawet jeśli nie planujesz ich używać? Obsługiwane powinny być wszystkie funkcje, które użytkownik może włączyć w panelu administracyjnym.
- Czy motyw jest przystosowany do lokalizacji? Czy powinien być?
- Czy wszystkie daty i godziny są poprawnie wyświetlane? Lepiej nie kodować ich formatów na sztywno za pomocą parametrów funkcji `the_date()` i `the_time()`. Użytkownik powinien mieć możliwość kontrolowania tych ustawień w panelu administracyjnym.
- Czy ustawiłeś zmienną określającą szerokość treści w pliku *functions.php*?
- Czy użytkownik może ustawiać własne nagłówki i tła? Czy powinna być taka możliwość?
- Czy włączona jest obsługa formatów wpisów? Czy jest to potrzebne?
- Pomyśl o innych funkcjach, które mogłyby być przydatne, np. własnym arkuszu stylów dla wizualnego edytora w panelu administracyjnym.
- Jeśli dodałeś standardową obsługę jakichś wtyczek, to upewnij się, że motyw działa także, gdy nie są one zainstalowane.
- Czy informacje w plikach *readme.txt* i *style.css* są aktualne? Czy motyw spełnia wszystkie warunki licencji?
- Czy przeprowadziłeś podstawowe testy, aby sprawdzić, czy na listach wyświetlane są odpowiednie wpisy, czy da się dodawać komentarze itd.? Nie zapomnij o podstawowych rzeczach: w swoim motywie mogłeś uszkodzić wiele różnych funkcji WordPressa, a więc przetestuj go od podstaw!



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

**Kwestie, na które zwrócą uwagę użytkownicy:**

- Czy błędy 404 są poprawnie obsługiwane?
- Czy dostępny jest formularz wyszukiwania i czy strona wyników wyszukiwania jest przejrzysta?
- Czy motyw zawiera wszystkie potrzebne pliki szablonowe lub czy są przewidziane inne rozwiązania w ich miejsce? Upewnij się, że archiwa tagów, kategorii, autorów, dat itd. działają zgodnie z oczekiwaniami.
- Czy zagnieżdżone kategorie i strony są wyświetlane prawidłowo? Jeśli w jakichś obszarach na widżety nie powinny one być wyświetlane, to czy użytkownik jest o tym poinformowany?
- Czy widok pojedynczego wpisu jest odpowiednio sformatowany za pomocą CSS?
- Czy widok stron jest odpowiednio sformatowany za pomocą CSS?
- Czy tam, gdzie powinna być wywołana funkcja `the_content()`, nigdzie nie ma wywołania funkcji `the_excerpt()`?
- Czy działa dzielenie list wpisów na strony i czy wyświetlane są odnośniki do następnego i poprzedniego wpisu na stronach wpisów?
- Czy imię autora jest wyświetlane tak, jak planowałeś?
- Czy wszystkie załączniki (obrazy, filmy itp.) są poprawnie wyświetlane?
- Czy galerie obrazów dobrze wyglądają?
- Czy włączyłeś możliwość dodawania ikon dla wpisów i stron?
- Co się dzieje po wyłączeniu możliwości komentowania? Upewnij się, że odpowiedzi wyglądają dobrze i są wyświetlane tak, jak byś chciał.



**Formatowanie:**

- Czy zdefiniowałeś odpowiedni styl dla każdego elementu dostępnego w edytorze wizualnym w panelu administracyjnym WordPressa? Dotyczy to m.in. cytatów blokowych, tabel oraz tabel uporządkowanych i nieuporządkowanych.
- Czy cytaty blokowe, listy itp. działają w komentarzach?
- Czy komentarze i trackbacki mają różne formatowanie? Czy komentarze autora wpisu są wyróżnione?
- Czy przyklejone wpisy mają specjalny styl? Czy jest to potrzebne?
- Czy nagłówki `h1`-`h6` dobrze wyglądają (nawet jeśli nie przewidujesz, że wszystkie będą używane)?
- Czy obrazy wstawiane za pomocą WordPressa są poprawnie wyświetlane? Dotyczy to zarówno obrazów pływających w lewo i prawo, jak i wyśrodkowanych.
- Czy podpisy pod obrazkami są wyświetlane?
- Co się dzieje, gdy użytkownik opublikuje za szeroki obraz? Czy psuje on cały projekt?

Oczywiście jest jeszcze wiele innych kwestii dotyczących bezpośrednio samego motywu, które należy sprawdzić. Należy sprawdzić, czy łącza w menu działają i czy tekst jest wszędzie czytelny. Dzięki powyższym listom unikniesz niektórych najczęściej spotykanych problemów

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

z motywami. Możesz do nich dodać jeszcze własne punkty, aby mieć jeszcze większą pewność, że Twoje motywy będą wysokiej jakości.

## MOTYWY KOMERCYJNE A LICENCJA GPL

Aktualnie do repozytorium WordPress.org nie są przyjmowane motywy płatne. Jeśli jednak zajmujesz się sprzedażą motywów, możesz dostać się na stronę komercyjnych motywów, która zawiera tylko odnośniki i zrzuty ekranu do popularnych motywów, ale nie są one przechowywane na serwerach WordPressa. Innymi słowy, komercyjnych motywów na licencji GPL nie da się automatycznie aktualizować z poziomu panelu administracyjnego WordPressa, ponieważ nie są one przyjmowane do zbiorów WordPress.org. Oczywiście zawsze możesz udostępnić motyw za darmo i wtedy możliwe, że zostanie on przyjęty. Wówczas możesz zarabiać na udzielaniu pomocy i dostosowywaniu motywu do potrzeb użytkowników.

Strona z komercyjnymi motywami jest dość nowa w witrynie WordPress.org i dyskusja nad tym, jak te motywy traktować, nadal trwa. Jeśli planujesz w jakikolwiek sposób zarabiać na płatnych motywach na licencji GPL, śledź rozwój sytuacji i zapoznaj się ze stroną *http://wordpress.org/extend/themes/commercial*.

## ZGŁASZANIE MOTYWÓW DO WORDPRESS.ORG



Jeśli przejrzałeś listy potencjalnych problemów i nie masz żadnych zastrzeżeń do swojego motywu oraz udostępniasz go na licencji zgodnej z GPL, możesz go zgłosić do zbioru motywów WordPress.org. Warto tam wysłać swój motyw z kilku powodów. Przede wszystkim w ten sposób dotrzesz do potencjalnych użytkowników poprzez oficjalny kanał, który jest dostępny także poprzez panel administracyjny. Ponadto będziesz mógł wygodnie publikować nowe wersje i otrzymasz połączenie z forami WordPress.org.

Zanim zgłosisz swój motyw do zbiorów WordPressa, zainstaluj wtyczkę Theme Check (*http://wordpress.org/extend/plugins/theme-check*), która pozwala przeprowadzić podobne testy, jakie przeprowadzają pracownicy zajmujący się recenzowaniem zgłaszanych motywów. Dzięki tej znakomitej wtyczce dowiesz się, jakie problemy występują w Twoim motywie, i otrzymasz wskazówki na temat ich możliwego rozwiązania. Jest to zatem nie tylko doskonała pomoc dla Ciebie, ale i dla recenzentów. Ponadto powinieneś włączyć opcję `WP_DEBUG`, tzn. ustawić ją na `true` w pliku *wp-config.php*:

```
define( 'WP_DEBUG', true );
```

Po skończeniu pracy zmień ją z powrotem na `false`. Dzięki temu otrzymasz więcej informacji o różnych błędach, zwłaszcza tych, które WordPress tłumi, ponieważ nie są decydujące dla działania witryny. Jeśli chcesz tylko sprawdzić skrypty JavaScript w WordPressie, użyj poniższego kodu:

```
define( 'SCRIPT_DEBUG', true );
```

Tak jak w przypadku `WP_DEBUG` po skończeniu pracy zmień to ustawienie na `false`.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Motyw powinien być kompletny i spakowany w formacie ZIP. W archiwum powinny znajdować się wszystkie pliki, a najważniejszym z nich jest *style.css*. To w tym pliku wpisuje się wersję motywu i tagi, które zostaną wykorzystane do klasyfikacji. Ponadto trzeba dołączyć plik o nazwie *screenshot.png* przedstawiający zrzut ekranu motywu w czasie, gdy jest używany (nie powinno to być logo ani nic w tym rodzaju). Wśród innych zasad jest jeszcze obsługa awatarów (zazwyczaj realizowana przy użyciu Gravatara — *http://gravatar.com*), obsługa widżetów, dostępność kanałów RSS, wyświetlanie tytułu i sloganu bloga oraz domyślne wyświetlanie list kategorii i tagów.

Przypominasz sobie etykietę *Tags*, która znajduje się na początku pliku *style.css*? To jest właśnie miejsce, w którym wpisujesz tagi, według których Twój motyw zostanie sklasyfikowany w serwisie WordPress.org. Tagi należy oddzielać od siebie przecinkami, na przykład:

```
Tags: light, two-columns, right-sidebar, fixed-width, custom-menus
```

Ten kod powinien znajdować się na samym początku pliku *style.css* wraz z innymi informacjami.

Poniżej znajduje się lista słów kluczowych, według których można wyszukiwać motywy w zbiorach WordPress.org (najbardziej aktualna wersja zawsze znajduje się pod adresem *http://wordpress.org/extend/themes/about*).



**Kolory:**

- black,
- blue,
- brown,
- gray,
- green,
- orange,
- pink,
- purple,
- red,
- silver,
- tan,
- white,
- yellow,
- dark,
- light.

**Kolumny:**

- one-column,
- two-columns,
- three-columns,

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- four-columns,
- left-sidebar,
- right-sidebar.

**Szerokość:**

- fixed-width,
- flexible-width.

**Funkcje:**

- blavatar,
- buddypress,
- custom-background,
- custom-colors,
- custom-header,
- custom-menu,
- editor-style,
- featured-image-header,
- featured-images,
- flexible-header,
- front-page-post-form,
- full-width-template,
- microformats,
- post-formats,
- rtl-language-support,
- sticky-post,
- theme-options,
- threaded-comments,
- translation-ready.

**Tematyka:**

- holiday,
- photoblogging,
- seasonal.

Postaraj się jak najlepiej dobrać tagi, ponieważ to według nich Twój motyw będzie znajdowany przez przyszłych użytkowników, niezależnie od tego, czy skorzystają z wyszukiwarki na stronie WordPress.org, czy użyją funkcji instalowania motywów w panelu administracyjnym (*Wygląd/Motywy/Zainstaluj motywy*).

Motywy wysyła się za pośrednictwem strony *http://wordpress.org/extend/themes/upload*.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# W NASTĘPNYM ROZDZIALE

Umiesz już posługiwać się składnią WordPressa (poznałeś ją w rozdziale 2.), wiesz, jak działa pętla (dowiedziałeś się tego w rozdziale 3.) oraz z czego składa się motyw.

Możliwe, że już zastanawiasz się nad tworzeniem własnego motywu bazowego dostosowanego do Twoich potrzeb. To bardzo dobry pomysł, pamiętaj tylko, aby wziąć wszystko co najlepsze z innych znanych Ci motywów, a następnie dostosować kod do własnych potrzeb. Nie ma powodu, dla którego miałbyś jeszcze się wstrzymywać z dłubaniem we własnych motywach, chociaż lektura dwóch kolejnych rozdziałów na pewno otworzy przed Tobą jeszcze większe możliwości.

W następnym rozdziale opisane są motywy potomne. Dowiesz się, jak utworzyć motyw na bazie innego motywu, nie modyfikując go. Pomyśl o tym przez chwilę i zastanów się, jakie możliwości to stwarza, albo po prostu zacznij czytać następną stronę.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 5 MOTYWY POTOMNE

**JAKO PROJEKTANT MOTYWÓW** do WordPressa musisz ciągle pamiętać o tym, że do systemu mogą być dodawane nowe funkcje, a stare mogą być wycofywane. Motyw, który został utworzony kilka lat temu, powinien działać także i dzisiaj, ale będzie pozbawiony wielu nowych funkcji. Pozostaje też otwarte pytanie, czy będzie działał także za kilka lat? Twórcy WordPressa dbają o to, aby kolejne wersje były zgodne z poprzednimi, ale wszystko ma swoje granice.

Tworzenie motywów bazowych w WordPressie i opieranie na nich motywów potomnych jest korzystne z wielu powodów, wśród których najważniejszym jest łatwiejsze utrzymanie zgodności z kolejnymi wersjami systemu.

W pewnym sensie koncepcja motywu potomnego dotyczy oddzielenia warstwy prezentacyjnej witryny o jeden krok od zaplecza, ponieważ motyw potomny zwykle składa się głównie z wizualnych dodatków do motywu głównego. Dzięki temu użytkownik, aktualizując podstawowy motyw, ma pewność, że niczego nie zepsuje w swojej witrynie.

W tym rozdziale znajduje się opis genialnej koncepcji motywów potomnych oraz sposobów ich efektywnego wykorzystania we własnych projektach.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# GENIALNOŚĆ MOTYWÓW POTOMNYCH

**Motywy potomne** to technika tworzenia motywów w oparciu o inne motywy, zwane **motywami nadrzędnymi** (ang. *parent theme*), polegająca na modyfikowaniu wybranych części motywu nadrzędnego tak, aby dostosować je do indywidualnych potrzeb. Przyjmijmy na przykład, że bardzo podoba się nam pewien motyw, ale wolelibyśmy zmienić w nim czcionki i kolorystykę. Ponadto przydałoby mu się kilka szablonów stron. Problemy te możemy rozwiązać na dwa sposoby. Pierwszy jest oczywisty: możemy po prostu bezpośrednio zmodyfikować pliki motywu. W takim przypadku musielibyśmy wprowadzić zmiany w pliku *style.css* (aby zmienić kolory i czcionki) oraz utworzyć kilka szablonów stron. Nic wielkiego, prawda?

Nieprawda! Co się stanie, gdy zostanie opublikowana aktualizacja motywu z nowymi funkcjami, a Ty, przeszczęśliwy, że jest nowa wersja, ją zainstalujesz? Wówczas wszystkie Twoje modyfikacje zostaną utracone, bo przecież autor motywu nie dodał ich do oryginalnego projektu. Teraz musisz wszystko robić od nowa, aby znowu dostosować wygląd witryny do swoich potrzeb. Oczywiście zawsze możesz sobie zapisać, co zmieniasz, i robić kopie zapasowe zmienionych plików, ale w nowej wersji motywu może być sporo zmian i ponowne wprowadzanie własnych modyfikacji w najlepszym wypadku może być tylko uciążliwe, a w najgorszym — trudne i czasochłonne.



Bezpośrednia modyfikacja plików motywu to dobre rozwiązanie, ale jeśli chcesz móc go bezproblemowo aktualizować, jest na to lepszy sposób. Możesz utworzyć motyw potomny tego motywu (który będzie w takim przypadku **szablonem**, na podstawie którego utworzysz nową wersję). Motyw potomny znajduje się w osobnym folderze, w którym należy zapisać wszystkie jego pliki. Dzięki temu plikom w motywie potomnym nic się nie stanie, gdy podczas aktualizacji oryginału zostaną podmienione pliki, a więc i Twoje zmiany pozostaną nienaruszone. Ogólnie rzecz biorąc, chodzi o to, aby oddzielić główny motyw od własnych adaptacji. A dzięki temu, że te adaptacje są przechowywane w osobnym folderze, aktualizacja motywu nadrzędnego nie ma na nie wpływu. Koncepcja ta jest dodatkowo przedstawiona na rysunku 5.1.

Rysunek 5.1. Koncepcja motywów potomnych

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## JAK DZIAŁAJĄ MOTYWY POTOMNE

Każdy motyw może być motywem nadrzędnym. Zarówno motyw nadrzędny, jak i potomny muszą znajdować się w folderze *wp-content/themes/*, jak zwykłe motywy. Aby na przykład jako nadrzędnego użyć motywu Notes Blog, należy umieścić go w folderze *wp-content/themes/*, a następnie również w folderze *wp-content/themes/* umieścić folder z motywem potomnym, np. Small Notes.

W pliku *style.css* motywu potomnego należy poinformować WordPressa, że jest to motyw potomny oraz wskazać jego rodzica. Za każdym razem, gdy system otrzyma żądanie jakiegoś pliku szablonowego, najpierw poszuka go w folderze motywu potomnego, a jeśli go tam nie znajdzie, pobierze go z folderu motywu nadrzędnego.

Poniżej znajduje się przykładowy fragment typowego pliku *style.css*:

```
/*
Theme Name: nazwa motywu
Theme URI: http://strona-motywu.pl
Description: ciekawy i wciągający opis motywu!
Author: Twoje imię i nazwisko
Author URI: http://twoja-strona.pl
Version: numer wersji
*/
```



Aby zaznaczyć, że motyw jest potomkiem innego motywu, należy użyć instrukcji `Template` składającej się ze słowa `Template:` i nazwy folderu, w którym znajduje się motyw nadrzędny:

```
Template: nazwa folderu motywu nadrzędnego (tylko motywy potomne)
```

Poniżej znajduje się drugi przykład kodu z pliku *style.css* definiujący motyw o nazwie Small Notes jako potomka motywu Notes Blog:

```
/*
Theme Name: Small Notes
Theme URI: http://notesblog.com/blog/small-notes/
Description: motyw Small Notes, potomek motywu Notes Blog
Author: Thord Daniel Hedengren
Author URI: http://tdh.me
Template: notes-blog
Version: 1.0

Aby motyw ten działał, w folderze wp-content/themes/ muszą znajdować się motywy Small Notes i Notes Blog.

*/
```

Jest to zawartość pliku *style.css* motywu potomnego, który można włączyć tak samo jak każdy inny motyw w panelu administracyjnym.

W pliku tym możesz teraz zmienić czcionki i kolory, które Ci się nie podobały. Najpierw musisz zdecydować, czy całkowicie zastąpisz plik *style.css* z motywu nadrzędnego, czy będziesz na nim bazować. W tym przykładzie użyjemy oryginalnego pliku jako bazy. W tym celu zaimportujemy go za pomocą dyrektywy `@import`:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
@import url("../notes-blog/style.css");
```

Wpisz ten wiersz kodu pod nagłówkiem informacyjnym w pliku *style.css*, a wszystkie reguły mające coś zmienić w wyglądzie stron wpisz pod nim. Na przykład poniżej zmieniamy czcionki i kolory:

```
@import url("../notes-blog/style.css");
div#content { font-family: Georgia, Times New Roman, serif; }
ul.sidebar { color: #444; }
```

Teraz tekst elementów `div` o identyfikatorze `content` będzie priorytetowo drukowany czcionką Georgia, a tekst w elementach `ul` należących do klasy `sidebar` będzie miał kolor ciemnoszary. Te reguły zostaną zastosowane **po** wszystkich regułach znajdujących się w arkuszu stylów motywu Notes Blog, ponieważ najpierw zostanie wykonana dyrektywa `@import`.

Cała zawartość pliku *style.css* z uwzględnieniem opisanych zmian wygląda teraz tak:

```
/*
Theme Name: Small Notes
Theme URI: http://notesblog.com/blog/small-notes/
Description: motyw Small Notes, potomek motywu Notes Blog
Author: Thord Daniel Hedengren
Author URI: http://tdh.me
Template: notes-blog
Version: 1.0
.
Aby motyw ten działał, w folderze wp-content/themes/ muszą znajdować się motywy Small Notes i Notes Blog.
.
*/

@import url("../notes-blog/style.css");

div#content { font-family: Georgia, Times New Roman, serif; }

ul.sidebar ( color: #333; }
```



Teraz przejdziemy do szablonów stron, o których była już wcześniej mowa. Ich utworzenie nie sprawi nam problemu, ponieważ należy to zrobić dokładnie tak samo, jak gdybyśmy modyfikowali bezpośrednio motyw Notes Blog, tylko pliki zapisujemy w folderze motywu Small Notes. Będą one używane tak, jakby należały do zwykłego motywu.

Każdy plik szablonowy znajdujący się w motywie potomnym jest klasyfikowany wyżej niż jego odpowiednik w motywie nadrzędnym. Oznacza to, że nawet jeśli w folderze motywu Notes Blog znajduje się plik *sidebar.php*, to i tak ładowany będzie plik o tej samej nazwie z folderu motywu Small Notes. Jeśli nie chcesz niczego zmieniać w oryginalnym pliku *sidebar.php*, po prostu nie dodawaj go do motywu potomnego.

Jest jednak jeden wyjątek od tej reguły: plik *functions.php*. Plik ten zostanie wczytany z motywu nadrzędnego, nawet jeśli w motywie potomnym również będzie znajdował się plik *functions.php*, przy czym system najpierw wczyta plik z motywu potomnego, a potem z nadrzędnego.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jest to korzystne rozwiązanie, ponieważ dzięki temu w pliku *functions.php* motywu potomnego możemy modyfikować funkcje motywu nadrzędnego. Innymi słowy, jeśli chcesz w motywie potomnym coś zmienić, możesz nadpisać części pliku *functions.php* motywu nadrzędnego, zmieniając tylko ten kod, który Ci nie odpowiada, i pozostawiając resztę pliku *functions.php* motywu nadrzędnego w pełni funkcjonalną. Wielką zaletą opisywanego rozwiązania jest to, że wszelkie zmiany są wprowadzane w pliku w motywie potomnym, dzięki czemu gdy zostanie zaktualizowany motyw nadrzędny, nic z nich nie zostanie utracone. Co ciekawe, motyw potomny również odniesie korzyści z tej aktualizacji, mimo że pozostanie nietknięty.

## PIĘKNO TECHNIKI PRZESŁANIANIA PLIKÓW SZABLONOWYCH

Dzięki temu, że większość plików szablonowych znajdujących się w motywie potomnym przesłania pliki o takich samych nazwach w motywie nadrzędnym, można bezpiecznie modyfikować tylko wybrane części motywu. Oznacza to, że w motywie potomnym tworzy się nowe pliki szablonowe, które mają wyższy priorytet od plików z motywu nadrzędnego. Załóżmy na przykład, że chcemy zmienić styl archiwów kategorii. W tym celu wystarczy w motywie potomnym utworzyć plik *category.php*, a zostanie on użyty w miejsce pliku *category.php* motywu nadrzędnego. Możliwe, że będzie trzeba też napisać jakieś reguły CSS, ale je wystarczy dodać do arkusza stylów motywu potomnego.

Jednak o pięknie rozwiązania, jakim są motywy potomne, nie stanowi proste przesłanianie szablonów. Powiedzmy, że chcemy, aby tylko wybrana kategoria wyglądała inaczej niż inne. Jeśli do motywu potomnego dodamy plik o nazwie *category-awesome.php*, to w nim będziemy mogli zmienić wygląd kategorii, której uproszczona nazwa brzmi **awesome**. Plik ten można by było dodać również do motywu nadrzędnego, ale mogłoby to powodować problemy z aktualizacją, a poza tym nie zawsze chcemy, aby zmiany były globalne.



## WSPANIAŁY SZABLON LOOP.PHP

Doskonałą mieszanką jest połączenie motywów potomnych i dodatkowych plików szablonowych zawierających pętle. W wielu motywach pętla w całości jest umieszczana w pliku *loop.php*, dzięki czemu jest oddzielona od projektowego kodu HTML. Jest to bardzo pomocne przy budowie motywów potomnych, ponieważ można zmodyfikować tylko samą pętlę, zamiast przesłaniać cały szablon.

Przypuśćmy, że chcemy zmodyfikować pętlę archiwów kategorii. Nasz motyw nadrzędny zawiera plik *category.php*, który z kolei dołącza pętlę z pliku *loop-category.php* za pomocą poniższego znajomo wyglądającego kodu:

```
<?php get_template_part( 'loop', 'category' ); ?>
```

Jak już wiemy, system najpierw poszuka pliku *loop-category.php*, a jeśli go nie znajdzie, użyje pliku *loop.php*.

Wcześniej, aby dostać się do pętli, musielibyśmy utworzyć nowy plik *category.php*. Teraz natomiast wystarczy, że w motywie potomnym umieścimy plik *loop-category.php*. Dzięki temu, że pierwszeństwo mają pliki motywu potomnego, plik `category.php` motywu

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

nadrzędnego pobierze zawartość pliku *loop-category.php* motywu potomnego. Innymi słowy, możemy dołączyć pętlę do pliku *category.php*, nie tworząc nowego pliku *category.php* w motywie potomnym. Wystarczy sama pętla.

To samo oczywiście dotyczy pozostałych plików szablonowych, które są dołączane w ten sposób. Powiedzmy, że nasze menu różnie wygląda w zależności od miejsca w witrynie. Dołączamy je za pomocą funkcji `get_template_part()`, która szuka najpierw pliku *nav-X.php*, a jeśli go nie znajdzie, dołącza domyślnie plik *nav.php*. Poniższy kod spowoduje, że system najpierw będzie szukał pliku *nav-archives.php*, a jeśli go nie znajdzie, to dołączy *nav.php*:

```
<?php get_template_part( 'nav', 'archives' ); ?>
```

## Kilka słów o semantyce motywów

Motywy potomne mogą wiele rzeczy wywrócić do góry nogami. Powiedzmy, że bardzo nam się podoba jakiś motyw. W motywie tym używana jest klasa `column-right` definiująca pasek boczny znajdujący się po prawej stronie treści. Nic w tym niezwykłego. Problem pojawia się, gdy chcemy przenieść ten pasek boczny na lewą stronę, co można łatwo zrobić, stosując deklarację `float: left` w arkuszu stylów motywu potomnego. Mimo że można to wykonać, nie jest dobrze, gdy element przypisany do klasy `column-right` znajduje się po lewej stronie.

Może pomyślisz, że przesadzam, ale pisanie sensownego kodu jest bardzo ważne nie tylko wtedy, gdy nad projektem pracuje kilka osób, ale też i w ogóle. Elementom projektu nadaje się nazwy po to, aby było wiadomo, do czego służą, oraz aby ktoś czytający kod mógł rozszyfrować, o co w nim chodzi.

Dlatego właśnie klasa `column-right` powinna przenosić elementy na prawą stronę. Tam każdy będzie się ich spodziewał, patrząc na tę nazwę.

Inną ciekawą kwestią jest sama nazwa **pasek boczny**. Zdaniem wielu osób plik *sidebar.php*, a przynajmniej nazwa **sidebar**, powinien odejść na emeryturę. Jest to relikt przeszłości, z czasów gdy WordPress rzeczywiście służył tylko do blogowania. Dziś system ten jest CMS-em, a więc czymś więcej niż tylko platformą blogową. Dlaczego zatem używać nazwy **sidebar**, a nie **sidecolumn**? Można nawet pójść tym tokiem rozumowania jeszcze dalej. Kto powiedział, że ta część witryny musi znajdować się z boku czegokolwiek? W projektach jednokolumnowych zawartość pliku *sidebar.php* często znajduje się pod treścią główną i rozciąga się nad stopką na taką samą szerokość jak ta treść. Jest to poprawne zastosowanie dla tego elementu, ale nazwa pasek boczny tu oczywiście nie pasuje.

To pewnie trochę zbyt radykalne podejście i prawdopodobnie nazwa **sidebar** pozostanie jeszcze na długo w słowniku użytkowników WordPressa. To jednak nie znaczy, że my też mamy używać nazw typu `prawa-kolumna`. Warto o tym pamiętać podczas projektowania motywu, ponieważ podczas gdy w jednej chwili nazwa `prawa-kolumna` może wydawać się jak najbardziej trafiona, to w innej może się okazać, że `prawa-kolumna` znajduje się po lewej stronie. W szczególności może się tak zdarzyć, gdy ktoś użyje Twojego motywu jako nadrzędnego dla innego motywu.

Pamiętaj o semantyce. To ułatwi życie wszystkim.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W motywie potomnym można bez problemu utworzyć dostosowany plik *nav-archives.php*, podobnie jak można tworzyć różne pętle. W ten sposób można zmieniać także części witryny nie będące pętlą, co pozwala zaoszczędzić czas i pracę.

### MOTYWY POTOMNE DO ZASTOSOWAŃ SPECJALNYCH

Jednym z ciekawszych, choć może nie tak przełomowych zastosowań motywów potomnych jest tworzenie wersji motywów do specjalnych celów. Powiedzmy, że mamy motyw, który nam się podoba, ale ze względu na porę roku chcemy dodać płatki śniegu, renifery itp. elementy graficzne. Nic nie stoi na przeszkodzie, aby utworzyć motyw potomny z tymi wszystkimi dodatkami.

Albo wyobraź sobie, że postanowiłeś zarabiać na swojej witrynie, wynajmując firmom miejsce do celów promocyjnych. Zmiana teł, nagłówków i innych elementów dzięki motywom potomnym jest banalnie łatwa. Także reklamy wyświetlane na wszystkich stronach i inne podobne funkcje implementuje się w ten sposób z dużą łatwością.

Dzięki wykorzystaniu motywów potomnych do drobnych modyfikacji, dostosowywania wyglądu witryny do różnych okazji oraz promocji oryginalny motyw pozostaje wolny od zmian. Każdy dobry projektant powinien brać pod uwagę możliwość zastosowania tej techniki.



## INNE SPOJRZENIE NA KWESTIĘ DZIEDZICZENIA

Wiemy, że większość plików szablonowych motywu potomnego ma pierwszeństwo przed swoimi odpowiednikami w motywie nadrzędnym. Na przykład plik *style.css* motywu potomnego przesłania plik *style.css* motywu nadrzędnego, plik *index.php* przesłania plik *index.php* itd.

Motyw potomny dziedziczy po nadrzędnym tylko te pliki, które są mu potrzebne.

To stwarza pewne problemy, z których najbardziej oczywisty dotyczy stopnia, do jakiego należy przerabiać motyw nadrzędny. Ogólnie rzecz biorąc, w motywach potomnych chodzi o dostosowywanie wybranego motywu do indywidualnych potrzeb. Jeśli więc utworzysz motyw potomny na bazie jakiegoś motywu i dokonasz tylu zmian, że z tego starego projektu nic nie zostanie, będzie to oznaczało, że prawdopodobnie nie do końca rozumiesz ideę motywów potomnych. Taki sposób działania to po prostu utworzenie nowego motywu na bazie innego, przez co i tak nie odniesiesz potencjalnych korzyści z aktualizacji.

Utworzenie motywu potomnego jest zalecane, gdy większość zmian ma być dokonywana w pliku *style.css*, a trochę w plikach szablonowych i pliku *functions.php*. Pierwszy może być użyty do zmieniania wyglądu witryny, a pozostałe do modyfikowania sposobu jej działania.

W takim razie czy tworzenie motywu potomnego to dobry pomysł, czy zły? W większości przypadków jest to dobry pomysł. Jeśli w czasie pracy okaże się, że to, co tworzysz, jest praktycznie całkiem nowym produktem, lepiej zrezygnować z motywu potomnego i utworzyć nowy szablon, np. na bazie swojego motywu bazowego.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# CZĘSTO SPOTYKANE PROBLEMY

Z motywami potomnymi związane są dwie kwestie, które mogą powodować nieporozumienia. Pierwsza dotyczy przede wszystkim użytkowania tych motywów, ponieważ motyw potomny może działać tylko wówczas, gdy w folderze *wp-content/themes/* znajduje się jego motyw nadrzędny. Jest to oczywiste, ale większość użytkowników myśli, że aby używać wybranego motywu, wystarczy tylko go wysłać na serwer i włączyć, nie zaprzątając sobie głowy jeszcze jakimś innym motywem nadrzędnym. Jeśli jednak tego nadrzędnego motywu zabraknie, motyw potomny zdecydowanie nie będzie wyglądał ani działał tak, jak powinien.

Drugi problem jest natury technicznej i dotyczy ścieżki do motywu. Zazwyczaj jeśli chcemy się odwołać do folderu motywu, aby np. wyświetlić obraz, używamy funkcji `get_template_directory_uri()` w połączeniu z instrukcją `echo` wysyłającą na wyjście jej wynik. Ale w motywach potomnych ta funkcja nie działa tak, jak byśmy się tego spodziewali, ponieważ zwracana przez nią ścieżka prowadzi do folderu motywu nadrzędnego! Dlatego poniższy kod wyświetlający obraz wskazywałby folder motywu nadrzędnego, a nie potomnego:

```
<img src="<?php echo get_template_directory_uri(); ?>/images/
↪the-image.gif"alt="Mój obraz"/>
```



Na szczęście da się ten problem rozwiązać. Wystarczy zamiast `get_template_directory_uri()` użyć funkcji `get_stylesheet_directory_uri()`, która zwraca ścieżkę do folderu zawierającego arkusz stylów motywu. I nie zgadniesz... ta funkcja użyta w motywie potomnym zwraca ścieżkę do arkusza stylów motywu potomnego! W związku z tym powyższy kod można poprawić w następujący sposób, aby działał zgodnie z oczekiwaniami:

```
<img src="<?php echo get_template_directory_uri(); ?>/images/the-image.gif"
↪alt="Mój obraz"/>
```

Jest to częsta przyczyna problemów związanych z niewyświetlaniem się obrazów albo wyświetlaniem nie tych obrazów, co trzeba, gdy przypadkiem obrazy o takich samych nazwach znajdują się zarówno w motywie potomnym, jak i nadrzędnym.

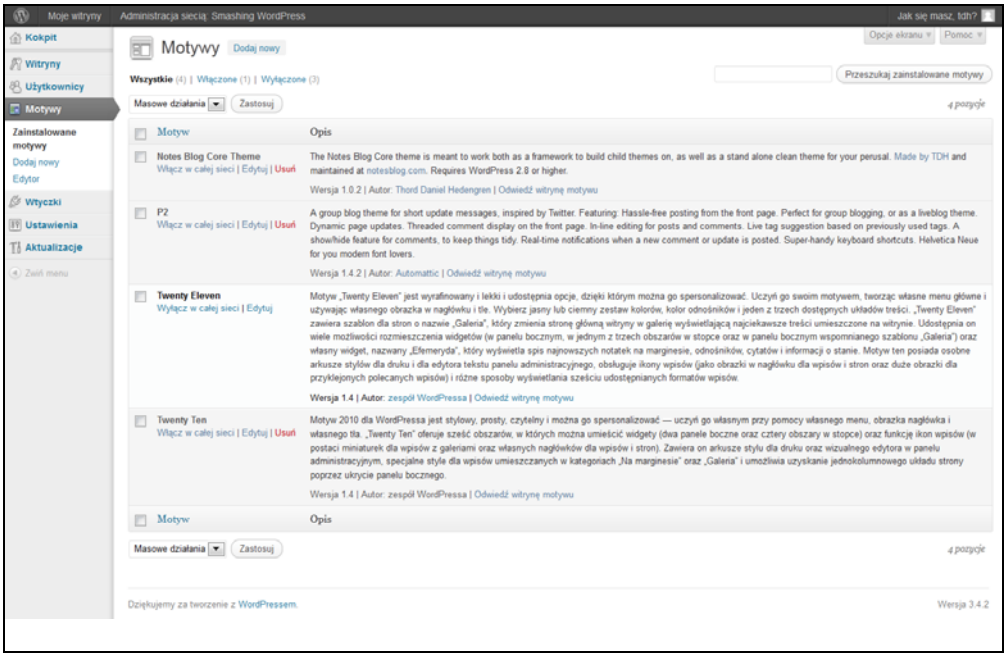
Pracując nad motywem potomnym, należy pamiętać, że funkcje odnoszące się do szablonu dotyczą motywu nadrzędnego, jak zilustrowałem to w powyższym przykładzie. W istocie jest to logiczne, ponieważ motyw potomny wykorzystuje motyw nadrzędny jako szablon, podczas gdy motyw nadrzędny jest — jak by to powiedzieć — swoim własnym szablonem.

# MOTYWY POTOMNE A ZARZĄDZANIE WIELOMA WITRYNAMI

Jeśli prowadzisz sieć blogów albo kilka witryn zbudowanych na bazie jednego motywu, to motywy potomne są wręcz stworzone dla Ciebie. Pomyśl: możesz więcej czasu poświęcić na wymyślanie i implementowanie nowych funkcji w motywie nadrzędnym, a wszystkie przeróbki polegające na dostosowaniu poszczególnych witryn możesz umieścić w ich motywach potomnych. W ten sposób przyspieszysz swoją pracę i ułatwisz sobie dokonywanie aktualizacji.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jeśli Twoi użytkownicy będą mieli do wyboru kilka motywów, pamiętaj, aby za pomocą opcji w panelu administracji siecią usunąć z listy możliwości do wyboru motyw nadrzędny (rysunek 5.2). Włączając motywy potomne, dajesz użytkownikom możliwość korzystania z funkcji motywu nadrzędnego, ale tylko poprzez jeden z motywów potomnych. W ten sposób uniemożliwiasz im użycie motywu, który może nie być przeznaczony do ogólnego użytku.




Rysunek 5.2. Gdy włączona jest obsługa wielu witryn, w panelu administracyjnym można wybrać, które motywy mają być dostępne do wyboru dla użytkowników



## ZARZĄDZANIE PROJEKTAMI WIELU WITRYN

Po pierwsze: znajdź elementy projektu, które będą wspólne dla wszystkich witryn. Ponieważ większość ludzi nie uruchamia wielu witryn, bazując na tym samym wyglądzie, lecz raczej pożycza motywy i projekty z różnych źródeł, nawet lepszym pomysłem może być utworzenie bazowego motywu od podstaw i dostosowywanie go do konkretnych potrzeb. Jest kilka bardzo znanych sieci witryn tego typu, wystarczy tylko trochę się rozejrzeć.

Po drugie: gdy już znajdziesz wszystkie elementy wspólne, które będą znajdować się w Twoich witrynach, utwórz szkielet głównego motywu nadrzędnego. Zagospodaruj miejsce na wszystko, pomyśl, co gdzie mogłoby być, i ogólnie planuj wszystko z wyprzedzeniem.

Po trzecie: utwórz motyw nadrzędny. Powinien on być tak prosty, jak to tylko możliwe, tzn. powinien zawierać tylko te elementy, które będą potrzebne we **wszystkich** Twoich witrynach. Często popełnianym błędem jest przesadzanie ze stylizacją motywu nadrzędnego tylko dlatego,

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

że wygląda trochę nieatrakcyjnie. Nie rób tego, bo wtedy w motywach potomnych będziesz zmuszony do przesłaniania własnego kodu, który już raz został wczytany. Nie ma sensu spowalniać szybkości ładowania swoich motywów, prawda?

Po tych trzech krokach będzie już z górki. Możesz tworzyć motywy potomne dla różnych witryn sieci i rozwijać je w indywidualny sposób. Gdy sieć jest w całości oparta na motywach potomnych, to aby dodać nową funkcję, wystarczy zaimplementować ją w motywie nadrzędnym. Później dokonujesz aktualizacji motywu we wszystkich witrynach i nowa funkcja staje się dostępna w każdej z nich. Porównaj to z koniecznością implementacji każdej funkcji w każdym motywie osobno. W najlepszym przypadku będzie to dla Ciebie tylko uciążliwe. Wkrótce zrozumiesz też, że dzięki motywom potomnym można zaoszczędzić sobie zarówno pracy, jak i kosztów.

## NIE ZAPOMINAJ O PLIKU FUNCTIONS.PHP

Korzyści z utworzenia jednego motywu głównego i serii motywów potomnych są widoczne nie tylko w warstwie wizualnej. Można je dostrzec także w kwestii funkcjonalności. Przypomnę, że w każdym motywie, zarówno nadrzędnym, jak i potomnym, można mieć plik *functions.php*. To oznacza, że jeśli znasz język PHP, możesz dodawać do motywów funkcje podobne do wtyczek.



Kolejnym problemem, z jakim borykają się właściciele sieci witryn, jest zarządzanie wtyczkami. Wprawdzie dzięki funkcji aktualizacji dostępnej w panelu administracyjnym jest to już o wiele prostsze niż kiedyś, ale mimo wszystko niektóre z funkcji realizowanych przez wtyczki można równie dobrze zaimplementować w plikach szablonowych. Jednak umieszczanie funkcji bezpośrednio w szablonach nie jest wcale takie wygodne. Poza tym cała idea motywów dotyczy oddzielenia kodu projektowego od implementacyjnego, a zatem wpisywanie dodatkowych fragmentów kodu PHP do szablonów jest z nią sprzeczne.

Dobrym rozwiązaniem w takim przypadku może być użycie pliku *functions.php*. Jest to plik szablonowy, ale znajdujący się poza warstwą projektową motywu, a jego jedynym zadaniem jest rozszerzanie funkcjonalności motywu przy użyciu kodu PHP. Zatem czasami napisanie funkcji w pliku *functions.php* głównego motywu nadrzędnego może być lepszym rozwiązaniem niż zarządzanie takimi samymi wtyczkami w wielu witrynach. To podejście pozwala również zredukować ilość pracy potrzebnej do zarządzania witrynami. Nie zapomnij tylko o przenośności danych. Witryna powinna działać prawidłowo także po zmianie motywu.

## NIE NALEŻY PRZESADZAĆ

Motywy potomne są świetnym wynalazkiem i zawsze warto rozważyć możliwość ich wykorzystania. To powinno być już do tej pory jasne. Jednak nie należy popadać w przesadę. Na przykład mimo że plik *functions.php* motywu potomnego może być przydatny, to jeśli za bardzo się on rozrasta, należy zastanowić się, czy nie lepiej byłoby utworzyć nowy motyw. To samo dotyczy plików szablonowych. Jeśli w motywie potomnym utworzysz ich dużą liczbę, to pewnie lepiej będzie, jeśli utworzysz całkiem nowy motyw na bazie istniejącego.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Ważna jest też kwestia aktualizacji motywu potomnego. Jeśli utworzysz motyw potomny zawierający wiele nowych funkcji, którymi trzeba się zajmować, to prawdopodobnie źle zrozumiałeś ideę motywów potomnych. W idealnej sytuacji jedynym miejscem, w którym trzeba wykonywać jakiekolwiek prace konserwacyjne, jest motyw nadrzędny, a w motywie potomnym dokonuje się tylko drobnych poprawek. Nie zapominaj o tym. Jeśli motyw potomny robi się skomplikowany i się rozrasta, to rozważ, czy nie lepiej zamienić go w normalny motyw.

# SZKIELETY MOTYWÓW

W społeczności WordPressa toczy się wiele dyskusji na tematy związane ze szkieletami motywów. Możesz się zastanawiać, co te dyskusje mają wspólnego z motywami potomnymi, i powiem Ci, że bardzo dużo. Większość tzw. szkieletów motywów (ang. *theme framework*) to na wpół puste motywy przygotowane do tego, aby je zmodyfikować techniką motywów potomnych lub w inny sposób. W niektórych przypadkach na przykład własny kod należy umieszczać w folderach znajdujących się w obrębie motywu, a więc wszystko zależy od wizji samego twórcy szkieletu.

To jednak nie oznacza, że musisz się ograniczać tylko do tego, co wymyślił projektant. W tym sensie każdy motyw może być szkieletem do budowy innych motywów i każdy motyw może być użyty jako nadrzędny dla motywów potomnych. Zawsze można wziąć motyw i ustawić go jako nadrzędny, a potem modyfikować.



Działanie niektórych funkcji dostępnych w motywach szkieletowych zależy od **haków** (ang. *hooks*). W zasadzie na nich opiera się przekazywanie przez motyw implementacji i danych do innego motywu przy użyciu pliku *functions.php*. Następnie motyw potomny (lub plik pseudofunkcji w motywie szkieletowym, jeśli takie rozwiązanie wybrano) może używać tych haków do różnych celów, a także usuwać je w razie potrzeby. Więcej o hakach dowiesz się w następnym rozdziale.

Zatem każdy motyw nie będący potomkiem innego motywu może być użyty jako motyw nadrzędny, a motywy szkieletowe to po prostu motywy specjalnie przystosowane do tego, aby łatwo można ich było używać jako podbudowy do tworzenia innych motywów. Warto o tym pamiętać.

## DLA ODMIANY KILKA SŁÓW O MOTYWACH NADRZĘDNYCH

Wszystko to może wydawać się niezrozumiałe, do czego z pewnością przyczynia się bardzo luźna definicja pojęcia **motyw szkieletowy**. Projektanci motywów często swoje produkty nazywają **szkieletami**, do czego oczywiście mają prawo. Ja jednak twierdzę, że większość z tych szkieletów to zwykłe motywy, tylko trochę lepiej dostosowane do tworzenia nowych motywów na ich bazie. Te nowe motywy mogłyby być ich motywami potomnymi. Wszystko zależy od tego, co chce się osiągnąć. Jeśli jeszcze do tego wszystkiego dodamy pojęcie **motywów bazowych**, to już kompletnie możemy stracić orientację.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Najważniejsze do zapamiętania jest to, aby nie brać zbyt poważnie nazwy nadanej motywowi przez projektanta. Każdy samodzielny motyw może być motywem nadrzędnym i jeśli chodzi o motywy potomne, tylko to jest ważne.

## WYŻSZY POZIOM WTAJEMNICZENIA

Zrozumienie idei motywów potomnych jest pierwszym krokiem do przejścia na wyższy poziom wtajemniczenia w budowę motywów do WordPressa, a przynajmniej pozwala poszerzyć horyzonty. Wszystkie najważniejsze funkcje można umieścić w jednym motywie, a następnie na jego bazie tworzyć różne motywy potomne.

Jestem zwolennikiem oszczędzania czasu i upraszczania procesu aktualizacji oprogramowania i dlatego moim zdaniem motywy potomne są znakomitym wynalazkiem, chociaż można znaleźć sytuacje, w których nie sprawdzą się najlepiej. Na przykład w blogach mających bardzo dużą liczbę odwiedzin należy usuwać wszystko, co powoduje zwiększenie zużycia transferu, a więc motyw powinien być jak najbardziej oszczędny.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 6 MOTYWY DLA ZAAWANSOWANYCH

**ABY ZACZĄĆ TWORZYĆ COŚ WIĘCEJ** niż zwykłe witryny oparte na WordPressie, musisz poznać bardziej zaawansowane funkcje systemu dostępne dla projektantów motywów. Większość z nich opiera się na tagach szablonowych i warunkowych, które już znasz, ale sposób ich użycia może być inny. Istnieją też techniki pozwalające rozszerzyć lub zmodyfikować funkcjonalność WordPressa z poziomu motywu, co może być nie najlepszym pomysłem, jeśli motyw ma być udostępniany publicznie, lecz dobrym rozwiązaniem w projekcie, nad którym akurat pracujesz.

Wszystko tak naprawdę sprowadza się do tego, aby wykorzystać WordPressa we właściwy sposób do wykonania zadania. Zbudować i opublikować, a nawet sprzedać motyw to jedno, ale stworzyć w pełni funkcjonalną witrynę w oparciu o WordPressa to coś całkiem innego. Rozdział ten jest w całości poświęcony tematowi wykorzystania WordPressa we właściwy sposób. Aby móc to zrobić, należy dobrze znać budowę motywów, biegle posługiwać się arkuszami stylów, umieć dodawać własne nagłówki i tła oraz umieć użyć własnych taksonomii, gdy kategorie i tagi okażą się niewystarczające. Ponadto dzięki lekturze nauczysz się dodawać haki do motywu, aby móc z łatwością dodawać nowe funkcje. Tak, z motywem do WordPressa można naprawdę wiele zdziałać, a więc czas wziąć się do pracy.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PLANOWANIE MOTYWU

Każdy nowy projekt oparty na WordPressie należy rozpocząć od zastanowienia się, jakie funkcje będą potrzebne w motywie. Budowa zwykłego bloga jest prosta — zaczynamy od nagłówka i kontynuujemy prace nad kolejnymi częściami. Jeśli jednak zechcesz zbudować witrynę w stylu gazetowym, musisz wziąć pod uwagę wiele różnych czynników. Jedną z najważniejszych kwestii, jakie należy rozwiązać, jest wygląd planowanej witryny. Bo przecież mimo iż wszyscy kochamy WordPressa, to jednak własny motyw tworzymy po to, aby nasza witryna wyglądała niepowtarzalnie i nie miała domyślnej oprawy wizualnej.

Przed rozpoczęciem projektowania i pisania kodu należy przemyśleć kilka spraw, które są wymienione poniżej:

- Główny układ witryny — dla jakich sekcji, stron i większych elementów należy przygotować miejsce i w jaki sposób zostaną one zapełnione treścią z WordPressa? Zazwyczaj trzeba zaplanować budowę kilku pętli i utworzenie paru plików szablonowych.
- Sortowanie treści — należy zdecydować się na określony zestaw kategorii i wybrać, które części witryny są na tyle statyczne, że mogą być stronami. Ponadto czy tagi przypisywane wpisom będą ujawnione? Jeśli nie, to można je wykorzystać do dostosowywania wyglądu wpisów i innych tego typu zadań.
- Drobiazgi — czy w witrynie będą potrzebne jakieś dynamiczne elementy, w których będzie można umieszczać ankiety i specjalne oferty promocyjne? Obszary te powinny najczęściej być przystosowane do wyświetlania widżetów.
- Komentarze — większość nowoczesnych witryn publikujących artykuły umożliwia komentowanie publikacji, a więc też musisz zastanowić się, czy włączyć tę funkcję w swoim serwisie.
- Specjalne funkcje — czy potrzebujesz jakichś funkcji, których WordPress standardowo nie udostępnia? Jeśli tak, to sprawdź, czy nie ma odpowiedniej wtyczki (albo napisz własną), lub znajdź zewnętrzną usługę, którą możesz wcielić w witrynę.



Gdy dokładnie określisz, co chcesz zrobić, będzie Ci łatwiej zaprojektować motyw idealnie dostosowany do Twoich potrzeb. Po przemyśleniu wymienionych kwestii możesz rozpocząć budowanie koncepcji, rysując szkice na papierze, kombinując z różnymi fragmentami kodu i robiąc inne rzeczy, które masz w zwyczaju robić podczas pracy nad każdą swoją niesamowitą stroną.

Tworzeniem witryn opartych na WordPressie zajmuję się od wersji 1.5 tego systemu, a użytkownikiem takich witryn jestem jeszcze dłużej. Wśród zrealizowanych przeze mnie projektów znajdują się zarówno proste blogi, witryny w stylu gazetowym, jak i całkiem inne realizacje. W dalszej części książki dowiesz się, jak używać WordPressa do bardzo różnych celów. Udowodnię Ci, że system ten może służyć zarówno jako CMS, jak i szkielet, ale na razie zapamiętaj tylko, że ciągle szukam nowych sposobów jego wykorzystania.

Gdy rozpoczynam nowy projekt, zawsze najpierw zastanawiam się, jaką funkcję ma pełnić i jak mogę to zrealizować. Jakie rodzaje treści będą prezentowane i jak ją wyświetlić, aby było to jak najbardziej wygodne dla użytkownika? W trzech kolejnych podrozdziałach znajdują się

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

moje najważniejsze porady dotyczące tych kwestii. Z pewnością ta metoda pracy nie jest doskonała w każdej sytuacji, ale pomoże Ci dobrze przemyśleć projekt przed rozpoczęciem jego realizacji.

## ZASADA 1. STYLIZUJ WEDŁUG KATEGORII, SORTUJ WEDŁUG TAGÓW, A DOSTOSOWUJ WEDŁUG FORMATÓW WPISÓW

Kategorie są doskonałym sposobem wstępnej segregacji wpisów, np. można utworzyć kategorie *Muzyka* i *Książki*, ale nie powinno się tworzyć zbyt szczegółowych kategorii. Natomiast tagi mogą być bardzo precyzyjne, np. recenzja książki może należeć do kategorii *Książki* oraz mieć przypisane tagi określające jej autora, gatunek literacki, tytuł, wydawcę itd. I nie jest to tylko kwestią nazewnictwa, ale ma podstawy techniczne. Listy wpisów należących do określonych kategorii można formatować w dowolny sposób przy użyciu pliku *category.php*, a nawet plików *category-X.php* (w miejsce *X* należy wstawić identyfikator). Dzięki temu każdemu rodzajowi treści można nadać indywidualny styl.

Natomiast tagi są bardzo szczegółowe i można je traktować częściowo jak słowa kluczowe, a częściowo jak opisy treści. Można je wykorzystać w obu tych rolach np. do zebrania kolekcji recenzji wszystkich książek J.K. Rowling bez używania standardowej (i niezbyt dokładnej) wyszukiwarki. Starannie planując kategorie i tagi oraz ich wzajemne relacje, można wiele osiągnąć.



Potrzebujesz jeszcze większej kontroli? Czasami same kategorie i tagi są niewystarczające. Wówczas można posiłkować się własnymi taksonomiami. Jedną z ich zalet jest to, że mimo iż służą jako dodatkowy sposób organizacji treści, użytkownik nie musi wiedzieć o ich istnieniu. Własne taksonomie świetnie nadają się np. do zwiększenia kontroli nad stylizowaniem wpisów.

Jeszcze mało? Dodatkową władzę nad wpisami i ich wyglądem otrzymasz dzięki formatom wpisów. W typowym blogu formaty są zazwyczaj lepszym rozwiązaniem niż używanie kategorii, ale każdy przypadek należy rozpatrywać indywidualnie.

## ZASADA 2. STARANNIE PRZEMYŚL WŁASNE POLA

Własne pola są bardzo przydatnym narzędziem. Można ich używać do przechowywania zarówno danych, jak i obrazów oraz mogą zapełnić lukę, gdy potrzebujemy czegoś więcej niż tylko tytułu, uproszczonej nazwy, treści i wypisu lub gdy chcemy uniknąć używania kategorii albo tagów. To świetnie. Niestety pola te są niezbyt przyjazne dla użytkownika, o czym już pisałem, i dlatego trzeba z nimi uważać. Jest wiele świetnych witryn opartych na WordPressie, w których własne pola są wykorzystywane na dużą skalę. Jest to jak najbardziej możliwe, tylko trzeba nauczyć ludzi ich obsługi. Dlatego czasami lepszym rozwiązaniem jest użycie wtyczki, która zrobi to samo, ale w prostszy dla użytkownika sposób.

---

*Pamiętaj, że używanie własnych pól do robienia prostych rzeczy, takich jak pisanie notatek na marginesie, jest całkowicie zbędne, ponieważ istnieją też formaty wpisów. Pól własnych należy używać do przechowywania danych.*

---

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### ZASADA 3. UŻYWAJ STRON I WŁASNYCH TYPÓW WPISÓW

Największą zaletą stron jest to, że można utworzyć dowolną ich liczbę i dla każdej z nich stworzyć osobny szablon. Dzięki temu możemy w nich realizować dowolne zadania. Na przykład ja na stronach rzadko kiedy umieszczam typową treść, jak pętla WordPressa itp. Zamiast tego wykorzystuję je do różnych innych celów i mimo że zarządzanie nimi może nie jest bardzo wygodne — trzeba modyfikować bezpośrednio kod szablonu, ponieważ w panelu administracyjnym można zmienić tylko tytuł i uproszczoną nazwę — to za ich pomocą można wykonać wiele niestandardowych czynności.

Powiedzmy, że chcemy pochwalić się na stronie jakąś swoją pracą, która jest przechowywana w innym serwisie. Pracę tę możemy dodać przy użyciu JavaScriptu udostępnianego przez ten serwis. Niestety WordPress nie przepuściłby takiego kodu. Rozwiązaniem jest utworzenie szablonu strony i wpisanie kodu w tym szablonie. To samo dotyczy wyświetlania wyników wyszukiwania własnej przeglądarki Google, wyświetlania kanałów RSS itp.

Szablony stron to potężne narzędzie. Używaj ich z rozwagą.

Chcesz mieć jeszcze większą kontrolę? Zawsze możesz dodać własne typy wpisów dostosowane do Twoich potrzeb. Chcesz mieć całkiem osobną hierarchię stron? Nie ma problemu. Utwórz po prostu własny typ wpisów o podobnych właściwościach co strony. Własne typy wpisów są doskonałym rozwiązaniem, gdy potrzebujesz mieć większą kontrolę nad treścią i jej organizacją od strony administracyjnej.



### CZY TO WSZYSTKO?

Oczywiście te trzy zasady nie opisują wszystkich możliwych sytuacji, ale są to najważniejsze kwestie, którymi zajmuję się, rozpoczynając każdy kolejny projekt oparty na WordPressie. Poza wymienionymi sprawami warto także przemyśleć, czy umożliwimy subskrypcję kanału RSS, jakie rozdzielczości ekranu będą obsługiwane oraz czy dodatkowo będą obsługiwane urządzenia przenośne. Wiele zależy też od rodzaju witryny, jaką się tworzy, warto np. przemyśleć sposób działania wyszukiwarki oraz wagę plików, ponieważ nie każdy użytkownik ma łącze szerokopasmowe. Krótko mówiąc, miej otwarty umysł i próbuj wcielić się w potencjalnego użytkownika swojej witryny.

## INDYWIDUALNE TECHNIKI STYLIZACJI

Indywidualne formatowanie może pomóc sprawić, że wpisy i strony będą bardziej atrakcyjne wizualnie. W pierwszej chwili może Ci się wydawać to trudne do osiągnięcia, zwłaszcza jeśli chodzi o wpisy, które są wyświetlane w oparciu o jeden szablon *single.php*. Na szczęście są pewne sposoby, dzięki którym można trochę ożywić wpisy. Wśród nich znajduje się tag szablonowy `post_class()` w tandemie z CSS.

Jednak wcześniej musisz poznać inną, bardziej oczywistą technikę stylizacji. Mam na myśli strony, które o wiele łatwiej jest dostosować wyglądem do własnych potrzeb, ponieważ wystarczy utworzyć nowy szablon i umieścić w nim, co tylko się chce. Można nawet zmieniać

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

paski boczne, nagłówki, stopki i inne ważne elementy. Największą zaletą szablonów stron jest to, że można je dowolnie konfigurować, a żeby użyć utworzonego szablonu strony, wystarczy go wybrać z listy rozwijanej w panelu administracyjnym.

To samo dotyczy list kategorii i tagów. Jeśli np. chcesz dla wybranej kategorii ustawić grafikę w nagłówku, to wystarczy, że odpowiednio zmodyfikujesz plik szablonowy tej kategorii. Przypomnę, że plik o nazwie *category-X.php* (*X* oznacza identyfikator lub uproszczoną nazwę kategorii) ma pierwszeństwo przed plikiem *category.php*, który z kolei jest przed plikami *archive.php* i *index.php*. Zatem plik o nazwie *category-37.php* byłby używany do wyświetlania wpisów należących do kategorii *37*, a plik *category-krowy.php* — do wyświetlania wpisów z kategorii *krowy*. A zatem wystarczy tylko ustawić odpowiedni nagłówek w pliku *category-37.php* lub *category-cows.php*.

Mówiąc krótko, wszystkie części witryny, które są wyświetlane przy użyciu osobnych plików szablonowych, można łatwo dostosować do indywidualnych potrzeb. W rozdziale 4. napisałem, kiedy ładowane są poszczególne pliki. Weź te informacje pod uwagę, projektując swoją witrynę.

## STYLIZOWANIE WPISÓW

Stylizowanie indywidualnych wpisów może być trochę trudniejsze. Sposób prezentacji wszystkich wpisów jest określony w jednym pliku *single.php*, co oznacza, że trzeba poszukać innych rozwiązań niż w przypadku stron.



Do tego doskonale nadadzą się nam tagi `the_ID()` i `post_class()`, których należy użyć w elemencie `div.container` w następujący sposób:

```
<div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
  <!-- Treść wpisu -->
</div>
```

Taki lub podobny kod znajdziesz w większości motywów do WordPressa. Tag `the_ID()` zwraca identyfikator wpisu, co w tym kodzie sprawia, że elementowi `div` zostaje nadany identyfikator w postaci `post-X`, gdzie `X` to zwrócony przez tag identyfikator. W ten sposób można utworzyć uchwyt do każdego wpisu, aby go dowolnie sformatować. Na przykład dla wpisu o identyfikatorze `674` element `div` miałby identyfikator `id="post-674"`. Możesz go użyć do napisania arkuszy stylów.

Najczęściej jednak indywidualne formatowanie nadaje się nie pojedynczym wpisom, lecz ich kategoriom. Do tego celu można oczywiście użyć tagu warunkowego i prostego skryptu PHP, ale lepiej skorzystać z efektu działania tagu `post_class()`. Tag ten dodaje do elementu zestaw klas, za pomocą których można ten element formatować na wiele różnych sposobów, np. w zależności od miejsca publikacji wpisu itd.

Najpierw dodawana jest klasa o takiej samej nazwie jak identyfikator, tzn. wpis o identyfikatorze `id="post-674"` będzie miał też klasę o nazwie `post-674`. Ponadto zostaną utworzone klasy dla każdej kategorii, do jakiej dany wpis należy. Każda z tych klas będzie składała się ze słowa

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

`category-` i uproszczonej nazwy wpisu. Zatem dla kategorii *Najświeższe wiadomości*, której uproszczona nazwa brzmiałaby `najswiezsze-wiadomosci`, zostałaby utworzona klasa `category-najswiezsze-wiadomosci`. To samo dotyczy tagów. Jeśli jakiemuś wpisowi przypiszesz tag `zabawne`, to `post_class()` utworzy klasę `tag-zabawne`.

Dodatkowo zostaną utworzone klasy `post` i `hentry`. Pierwsza ma za zadanie zapewnić zgodność tagu `post_class()` ze starszymi motywami do WordPressa, w których element `div` zawierający wpisy jest najczęściej oznaczany właśnie tą klasą. Druga natomiast informuje, że element `div` należy do wpisu. Jest ona wciąż używana.

Jakie korzyści są z tych klas? Przede wszystkim dzięki nim można zmieniać formatowanie kategorii wpisów i chyba do tego te klasy są najczęściej używane. Powiedzmy, że mamy kategorię *Wiadomości* o uproszczonej nazwie `wiadomosci`. Dla tej kategorii tag `post_class()` utworzyłby klasę `class="category-wiadomosci"`. A teraz wyobraź sobie, że chcesz, aby odnośniki w tej kategorii miały kolor zielony oraz aby lewa krawędź kontenera tych wpisów też była zielona. Opisany efekt uzyskasz dzięki poniższemu arkuszowi stylów:

```
div.category-news {
    padding-left: 20px;
    border: 5px solid green;
    border-width: 0 0 0 5px; }

div.category-news a { color: green; }
```



Teraz każdy wpis należący do kategorii *Wiadomości* będzie miał zielone łącza i zieloną krawędź z lewej strony. To świetnie, bo jak widać, nazwy kategorii można wykorzystywać do kontrolowania oprawy wizualnej wpisów. Tagów natomiast powinno się używać do dodatkowego sortowania wpisów. Na przykład mogę mieć tag `Ulubione`. Wówczas każdemu wpisowi należącemu do ulubionych mogę dodać specjalną ikonę w prawym górnym rogu. Nie będzie to żadne łącze ani nic szczególnego, po prostu znak, że jest to jeden z moich ulubionych wpisów.

```
div.tag-ulubione { background: url(ulubione.gif) top right no-repeat; }
```

Ta reguła spowodowałaby wyświetlenie obrazu *ulubione.gif* w tle, w prawym górnym rogu każdego wpisu, dla którego tag `post_class()` zwróciłby klasę `tag-ulubione`.

W końcu do tagu `post_class()` można przekazać konkretną klasę, co również czasami bywa przydatne. Może chcesz nadać pojedynczym wpisom klasę `stars`? Jeśli tak, to zmień kod wywołania `post_class()` w pliku *single.php* w następujący sposób:

```
<div id="post-<?php the_ID(); ?>" <?php post_class( 'stars' ); ?>>
```

Dodanie klasy do tagu `post_class()` polega na przekazaniu parametru wywołania. Podobne rozwiązanie jest stosowane, gdy trzeba zachować specyficzne formatowanie poprzez użycie tagu `post_class()`, ale poza pętlą. Wówczas należy tagowi `post_class()` nakazać, aby zwrócił identyfikator, na przykład:

```
<?php post_class( '', $post_id ); ?>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## KLASY DLA ELEMENTU BODY

Innym sposobem na zastosowanie arkuszy stylów do różnych części witryny opartej na WordPressie jest użycie wprowadzonego w WordPressie 2.8 tagu szablonowego `body_class()`. Najprościej mówiąc, jest on tym samym dla elementu `body` co `post_class()` dla kontenera `div` zawierającego pojedynczy wpis. Używa się go następująco:

```
<body <?php body_class(); ?>>
```

W zależności od miejsca w witrynie elementowi `body` są przypisywane różne klasy. Jeśli na przykład czytamy wpis o identyfikatorze 245 i jesteśmy zalogowani, to element `body` wygląda następująco:

```
<body class="class="single single-post postid-245 single-format-standard logged-in">
```

Dla listy kategorii zostałyby wygenerowane inne klasy, a dla list tagów jeszcze inne. Strony, strona główna, strona wyników wyszukiwania i praktycznie każde miejsce w witrynie, jakie można sobie wyobrazić, ma własny mniej lub bardziej podobny do innych zestaw stylów dla elementu `body`.

Jak można to wykorzystać? Powiedzmy, że chcemy, aby nagłówki `h2` miały różne wielkości, zależnie od tego, czy znajdują się na stronie zawierającej listę kategorii, czy pojedynczy wpis. Pomysł ten można zrealizować, dodając klasy do różnych plików szablonowych albo wykorzystując klasy generowane przez tag `body_class()`.



Najpierw trzeba sprawdzić, jakie klasy są zwracane i kiedy są one zwracane. W dalszej części rozdziału zamieściłem całą listę tych klas, abyś mógł zobaczyć, z czym masz do czynienia. Na razie jednak wystarczy nam wiedzieć, że na stronach pojedynczych wpisów elementowi `body` jest przypisywana m.in. klasa `single`, a na większości stron zawierających jakieś listy wpisów (podobnie jak szablon *archive.php* jest używany zarówno dla list kategorii, jak i tagów, jeśli w motywie nie ma plików *category.php* i *tag.php*) — klasa `archive`. Wiedząc o istnieniu tych klas, możemy ich użyć do swoich celów, na przykład:

```
body.single h2 { font-size: 48px; }
body.archive h2 { font-size: 36px; }
```

Dzięki temu gdy funkcja `body_class()` zwróci klasę `single`, czyli w widokach pojedynczych wpisów, nagłówki h2 będę miały rozmiar 48 pikseli. Natomiast na stronach archiwów (klasa `archive`) nagłówki te będą miały rozmiar 36 pikseli.

Za pomocą klas dla elementu `body` można wiele zdziałać, ponieważ klasy tego elementu są dodawane w zależności od miejsca, w którym znajduje się użytkownik, i znajdują się na samej górze hierarchii elementów w dokumencie. W większości motywów, które zostały zaprojektowane przed pojawieniem się tagu `body_class()`, nie jest to wykorzystywane, ale jeśli dokładniej przemyślisz strukturę swoich arkuszy stylów, to odkryjesz, że przy użyciu klas dla elementu `body` łatwiej jest pracować, niż dodając klasy osobno dla każdego elementu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Poniżej znajduje się lista klas (od najważniejszej), które mogą być dostępne w zależności od strony przeglądanej w witrynie. Rzadko się zdarza, aby była potrzeba użycia ich wszystkich:

- `rtl`,
- `home`,
- `blog`,
- `archive`,
- `date`,
- `search`,
- `paged`,
- `attachment`,
- `error404`,
- `single-postid-`*`X`* (*X* jest identyfikatorem wpisu),
- `page-id-`*`X`* (*X* identyfikatorem strony),
- `attachmentid-`*`X`* (*X* jest identyfikatorem załącznika),
- `attachment-`*`MIME`* (*MIME* jest typem MIME),
- `author`,
- `author-`*`UŻYTKOWNIK`* (*UŻYTKOWNIK* jest uproszczoną nazwą użytkownika),
- `category`,
- `category-`*`X`* (*X* jest uproszczoną nazwą kategorii),
- `tag`,
- `tag-`*`X`* (*X* jest uproszczoną nazwą tagu),
- `page-parent`,
- `page-child parent-pageid-`*`X`* (*X* jest identyfikatorem strony),
- `page-template page-template-`*`PLIK`* (*PLIK* jest nazwą pliku szablonowego),
- `search-results`,
- `search-no-results`,
- `logged-in`,
- `paged-`*`X`* (*X* jest numerem strony listy),
- `single-paged-`*`X`* (*X* jest numerem strony listy),
- `page-paged-`*`X`* (*X* jest numerem strony listy),
- `category-paged-`*`X`* (*X* jest numerem strony listy),
- `tag-paged-`*`X`* (*X* jest numerem strony listy),
- `date-paged-`*`X`* (*X* jest numerem strony listy),
- `author-paged-`*`X`* (*X* jest numerem strony listy),
- `search-paged-`*`X`* (*X* jest numerem strony listy),
- `tax-`*`X`* (*X* jest nazwą taksonomii),



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- `term-X` (*X* jest nazwą terminu),
- `admin-bar`,
- `custom-background`.

Dzięki klasom generowanym przez tag `body_class()` można precyzyjniej kontrolować wygląd poszczególnych części witryny. W wielu starszych motywach znajdują się niepotrzebne pliki szablonowe, które łatwo można zastąpić arkuszami stylów podłączonymi do klas zwracanych przez tag `body_class()`. Ta metoda jest o wiele lepsza niż tworzenie specjalnych klas w kolejnych szablonach.

## PRZYKLEJONE WPISY

Kiedy w WordPressie 2.7 wprowadzono funkcję przyklejania wpisów, trzeba było do kontenerów `div` zawierających wpisy dodawać tag `sticky_class()`. Powodował on dodanie klasy `sticky`, przy użyciu której można było potem wpisowi nadać dowolne formatowanie. Dzięki tagowi `post_class()` tag ten nie jest jednak już potrzebny. Tag `post_class()` oprócz wymienionych wcześniej klas dodaje także klasę `sticky` do wpisów, które w panelu administracyjnym zostały oznaczone jako przyklejone.

Wyróżnienie przyklejonego wpisu jest bardzo łatwe. Wystarczy w arkuszu stylów użyć selektora `div.sticky`:

```
div.sticky {
    padding: 20px 20px 8px 20px;
    background: #f8f8f8; }
```



Jak widać, jest to bardzo proste (więcej na temat przyklejania wpisów piszę w rozdziale 3.). Jedyny problem dotyczący przyklejonych wpisów, jaki przychodzi mi do głowy, to że nie wiadomo, ile takich wpisów będzie. Jeśli będą dwa, to na samej górze zostanie wyświetlony nowszy z nich, a pod nim od razu drugi. To oznacza, że należy zadbać, aby wpisy te wyglądały dobrze także wtedy, gdy znajdują się w swoim bezpośrednim sąsiedztwie.

Do czego można wykorzystać możliwość przyklejania wpisów? Najbardziej oczywistym zastosowaniem w typowych blogowych witrynach jest przyklejanie na początku listy najważniejszych albo najdłuższych wpisów, które można następnie odkleić, gdy pojawi się coś nowszego i równie ważnego. Jest to coś w rodzaju dodatkowego obszaru nagłówkowego.

Innym oczywistym zastosowaniem dla przyklejonych wpisów jest publikowanie ogłoszeń. Jeśli np. sprzedajesz e-booki, możesz na stronie głównej umieścić następujące ogłoszenie: „Ej, Ty! Powinieneś kupić moją najnowszą książkę. Muszę przecież z czegoś opłacić czynsz” (rysunek 6.1). Można także publikować bardziej tradycyjne ogłoszenia, jak np. o uruchomieniu nowej witryny albo wezwanie do jakiegoś działania.

W końcu możesz też użyć tagu warunkowego `is_sticky()`. Przy jego użyciu można pobrać z WordPressa przyklejone wpisy (przy użyciu pętli i funkcji `query_posts()`), a następnie zrobić z nimi, co się chce. Jednym z pomysłów może być utworzenie obszaru nagłówkowego

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Rysunek 6.1. Po ukazaniu się tej książki w sprzedaży na stronie *http://notesblog.com/* znalazło się takie ogłoszenie



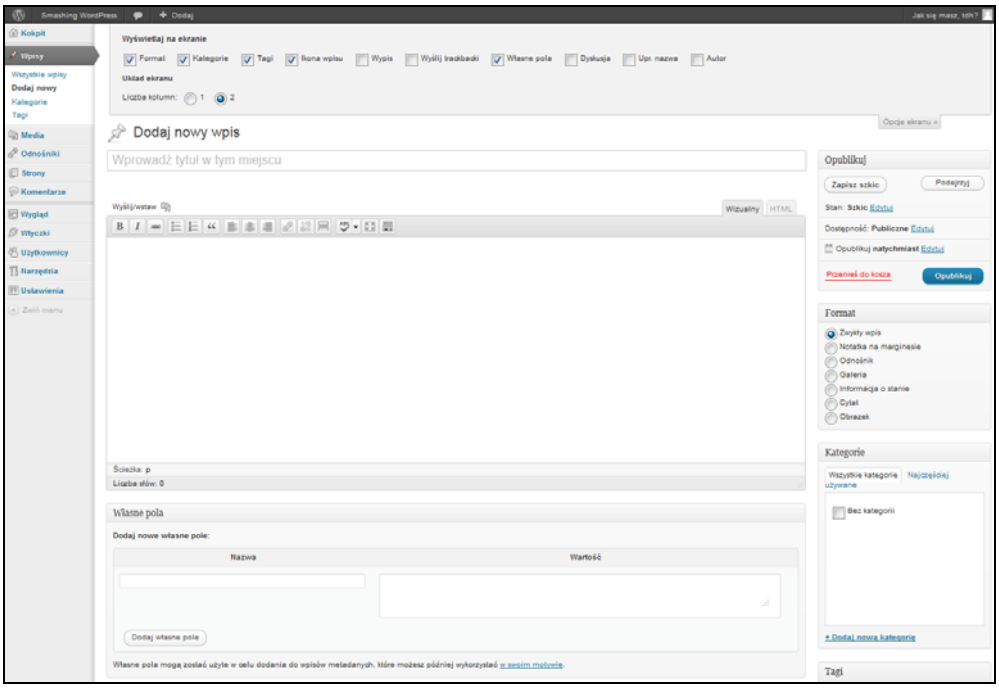
poza normalną pętlą i umieszczenie w nim określonej liczby przyklejonych wpisów. W ogóle w ten sposób można kontrolować zawartość nagłówka, ale ten sam efekt uzyska się także przy użyciu pól własnych lub tagów, a biorąc pod uwagę, że pole wyboru służące do przyklejania wpisów jest trochę ukryte, użycie tagu może być lepszym rozwiązaniem.

# UŻYWANIE WŁASNYCH PÓL

**Pola własne** to pola bazy danych, których zawartość jest w pełni kontrolowana przez użytkownika. Treści tej zawsze używa się w pętli. Na stronie dodawania nowego wpisu w panelu administracyjnym powinna znajdować się sekcja o nazwie *Własne pola*. Jeśli jej nie ma, trzeba ją włączyć w *Opcjach ekranu* na samej górze strony. W sekcji *Własne pola* można tworzyć nowe pola i definiować ich wartości (rysunek 6.2).

Własne pola to w istocie metadane, nazwane tak na potrzeby interfejsu WordPressa. W większości przypadków dane te należą do wpisów i składają się z dwóch elementów: klucza i wartości. Nowy klucz można dodać w dowolnym momencie i kiedy raz się utworzy klucz, jest on później dostępny w liście rozwijanej, dzięki czemu nie trzeba go więcej wpisywać, co pozwala uniknąć błędów przy wpisywaniu. Metadane mogą też należeć do komentarzy i użytkowników.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 6.2. Sekcja Własne pola jest dostępna w panelu administracyjnym na stronie dodawania nowego wpisu i znajduje się pod formularzem dodawania wpisu



## PODSTAWY WŁASNYCH PÓL

Rozpoczęcie pracy z własnymi polami jest łatwe. Wystarczy rozpocząć tworzenie nowego wpisu i wpisać cokolwiek w polu wartości, a następnie kliknąć przycisk *Dodaj własne pole*. W ten sposób zapiszesz dane we własnym polu, ale nigdzie nie będą one wyświetlane, dopóki się ich nie wywoła wewnątrz pętli. Ogólnie pola takiego używa się w wielu różnych wpisach i zmienia się tylko jego wartość w każdym wpisie. Jest to coś w rodzaju sposobu kategoryzacji przechowywanych danych.

Tag szablonowy służący do pobierania zawartości własnych pól nazywa się `the_meta()`. Domyślnie zwraca on nieuporządkowaną listę (czyli element `ul` z elementami `li`) zawierającą własne pola przypisane do wpisu. Jest to doskonały sposób na wyświetlanie informacji typu jaki jesteś w danej chwili szczęśliwy albo co teraz czytasz lub ocen recenzowanych publikacji. Element `ul` listy ma przypisaną klasę `post-meta`, a wartość każdego pola znajduje się w osobnym elemencie `li`. W elemencie `li` znajduje się element `span` z klasą `post-meta-key` zawierający klucz pola. Za nim znajduje się wartość pola w postaci tekstu.

Przypuśćmy, że mamy trzy własna pola: z kluczem `Aktor` i wartością `Harrison Ford`, z kluczem `Reżyser` i wartością `Steven Spielberg` oraz z kluczem `Tytuł filmu` i wartością `Indiana Jones. Poszukiwacze zaginionej arki`. Wszystkie te pola wraz z wartościami należą do konkretnego wpisu. Jeśli gdzieś w pętli umieścisz tag szablonowy `the_meta()`, to na stronie zawierającej ten wpis zostanie dodatkowo wyświetlony następujący kod:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<ul class='post-meta'>
    <li>
        <span class='post-meta-key'>Aktor</span> Harrison Ford
    </li>
    <li>
        <span class='post-meta-key'>Reżyser</span> Steven Spielberg
    </li>
    <li>
        <span class='post-meta-key'>Tytuł filmu</span> Indiana Jones. Poszukiwacze
        ↪zaginionej arki
    </li>
</ul>
```

## TWORZENIE MODUŁÓW META

Jednym ze sposobów na ułatwienie pracy z metadanymi jest pominięcie standardowego interfejsu własnych pól i utworzenie własnych modułów meta (ang. *meta box*). Do tego służy funkcja `add_meta_box()`. Utworzymy taki przykładowy moduł, który będzie zawierał pole tekstowe formularza i instrukcję.

Moduł utworzymy w funkcji `smashing_add_meta_boxes()`, którą powiążemy z plikami *load-post.php* i *load-post-new.php*:



```
// Powiązanie z load-post.php i load-post-new.php
add_action( 'load-post.php', 'smashing_add_meta_boxes' );
add_action( 'load-post-new.php', 'smashing_add_meta_boxes' );
```

Teraz napiszemy kod samej funkcji `smashing_add_meta_boxes()`:

```
// Tworzenie modułów meta
function smashing_add_meta_boxes() {
    // Tworzenie przykładowego modułu meta
    add_meta_box(
        'smashing-post-demo',
        'Przykładowy moduł',
        'smashing_post_demo_meta_box',
        'post',
        'side',
        'default'
    );
}
```

W funkcji tej można utworzyć kilka takich modułów, ale w przykładzie ograniczymy się do jednego. Moduł, który tu tworzymy, ma identyfikator `smashing-post-demo`, tytuł *Przykładowy moduł* oraz funkcję zwrotną (wkrótce opiszę dokładniej, co to jest) `smashing_post_demo_meta_box`. Ponadto moduł ten będzie wyświetlany tylko na stronach tworzenia nowych wpisów (dzięki opcji `post`), po prawej stronie (dzięki opcji `side`) oraz będzie miał domyślny priorytet. Więcej informacji na temat argumentów, jakie można przekazywać do funkcji `add_meta_box()`, znajduje się w dokumentacji na stronie *http://codex.wordpress.org/Function_Reference/add_meta_box*.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Musimy utworzyć jeszcze jedną funkcję, o nazwie `smashing_post_demo_meta_box()`, która będzie używana jako wywołanie zwrotne w funkcji `add_meta_box()` do tworzenia modułu:

```
// Wyświetla przykładowy moduł meta
function smashing_post_demo_meta_box( $object, $box ) { ?>
    <?php wp_nonce_field( basename( __FILE__ ), 'smashing_post_demo_nonce' ); ?>
    <p>
        <label for="smashing-post-class">
            To jest przykładowy moduł meta, w którym na razie nic nie działa.
        </label>
    </p>
    <p>
        <input class="widefat" type="text" name="smashing-post-demo"
            id="smashing-post-demo" value="<?php echo esc_attr( get_post_meta(
            ↪$object->ID,
             'smashing_post_demo', true ) ); ?>" size="30" />
    </p>
<?php
// Koniec funkcji
}
```

Funkcja ta zawiera pole `nonce`, dzięki czemu mamy pewność, że wszystkie znajdujące się wewnątrz pola są weryfikowane przez WordPressa. Jeśli nie wiesz, czym są pola `nonce`, możesz o nich przeczytać na stronie *http://codex.wordpress.org/Function_Reference/wp_nonce_field*.



Reszta to prosty kod HTML, który łatwiej jest pisać poza blokiem PHP i dlatego właśnie tak zrobiłem. Wartość pola wejściowego jest pobierana za pomocą funkcji `get_post_meta()`, a jego identyfikator ustawiono przy użyciu konstrukcji `$object->ID`. Wywołanie `esc_attr()` służy do oczyszczenia wartości.

Teraz na stronie tworzenia wpisu pojawi się nowy moduł, jak widać na rysunku 6.3.

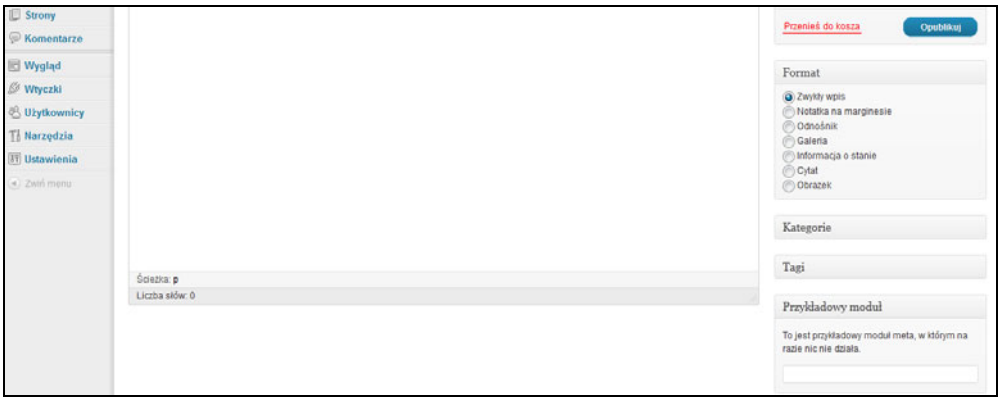
Na razie ten moduł nie spełnia żadnej funkcji, tzn. nie da się przy jego użyciu zapisywać danych itp. Aby cokolwiek zrobić w tym module, musimy użyć funkcji `add_post_meta()`, `update_post_meta()` lub `delete_post_meta()`. Opis tych funkcji wraz z przykładami można znaleźć w dokumentacji:

- `add_post_meta()`: *http://codex.wordpress.org/Function_Reference/add_post_meta*,
- `update_post_meta()`: *http://codex.wordpress.org/Function_Reference/update_post_meta*,
- `delete_post_meta()`: *http://codex.wordpress.org/Function_Reference/delete_post_meta*.

# CIEKAWE FUNKCJE WŁASNE

Określenie „funkcje własne” może być mylące, ponieważ obsługa własnych menu, nagłówków i teł jest wbudowana w WordPressa, chociaż nie w każdym motywie włączona. Jednak włączenie tych elementów w motywie jest bardzo łatwe, a dodatkowo przydatne dla użytkownika. W kilku poniższych podrozdziałach znajduje się opis tych funkcji.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 6.3. Nowy moduł meta

## POPRAWNE DODAWANIE FUNKCJI W PLIKU FUNCTIONS.PHP

Zanim przejdę do wspomnianych funkcji, najpierw wyjaśnię, jak je prawidłowo dodawać do pliku *functions.php*. Przypomnę, że w rozdziale 4. była mowa o haku `after_setup_theme` użytym w pliku *functions.php* motywu Simple Blog. Wszystkie funkcje motywu powinny być podłączone do tego haka. W tym celu każdą nową funkcjonalność należy zdefiniować w postaci funkcji i tę funkcję dodać do haka w pliku *functions.php*, na przykład:



```
// Konfiguracja motywu
add_action( 'after_setup_theme', 'smashing_theme_setup' );
function smashing_theme_setup() {
    // Tu należy wpisać funkcje
}
```

Jeśli więc chcesz dodać obsługę ikon wpisów (zobacz następny podrozdział), możesz napisać taki kod:

```
// Konfiguracja motywu
add_action( 'after_setup_theme', 'smashing_theme_setup' );
function smashing_theme_setup() {
    // Włączenie obsługi ikon wpisów
    add_theme_support( 'post-thumbnails' );
}
```

Oczywiście funkcja ta może być o wiele bardziej rozbudowana. Jeśli potrzebujesz odświeżenia wiadomości, jeszcze raz przeczytaj rozdział 4. i przejrzyj plik *functions.php* z motywu Simple Blog.

Funkcjonalność motywu w pliku *functions.php* należy rozszerzać za pomocą funkcji i haków, a nie wpisując kod jak popadnie. Oczywiście można nie stosować się do tej zasady, ale wówczas nie ma się kontroli nad tym, kiedy kod zostanie zainicjowany.

Dalej w przykładach kodu hak i definicja funkcji będą pomijane, aby nie zajmowały miejsca. Prezentowane przykłady kodu należy wpisywać w powyższej funkcji i wszystko będzie dobrze.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## IKONY WPISÓW

**Ikony wpisów** to obrazy graficzne reprezentujące wpisy, które dodaje się przy użyciu specjalnego okna dialogowego dostępnego na stronie edycji wpisu (rysunek 6.4). Włączenie tej funkcjonalności jest proste i wymaga tylko wywołania funkcji `add_theme_support()` z parametrem `post-thumbnails` w pliku *functions.php*, na przykład:

```
add_theme_support( 'post-thumbnails' );
```

Rysunek 6.4. Moduł Ikona wpisu znajduje się po prawej stronie na dole ekranu

To spowoduje pojawienie się modułu *Ikona wpisu* w panelu administracyjnym. Funkcjonalność tę można włączyć także np. dla własnych typów wpisów. Na rysunku 6.4 widać moduł *Ikona wpisu*, za pomocą którego użytkownik może wybrać obraz mający reprezentować dodawany wpis.



Można określić dopuszczalny rozmiar ikony wpisu. Za pomocą funkcji `add_image_size()` rejestruje się nowe rozmiary obrazów, które zostaną utworzone dla każdego pliku graficznego wysyłanego do biblioteki mediów, podobnie jak pliki o średnim i dużym rozmiarze. Poniższy kod, który powinien być używany z hakiem `after_setup_theme`, doda rozmiar obrazów o nazwie `top-feature` i wymiarach 500×225 pikseli. Jeśli chcesz skalować tylko według szerokości, to zamień wartość `225` na `9999`.

```
add_image_size( 'top-feature', 500, 225 );
```

Jednak samo włączenie możliwości dodawania ikon wpisów nie spowoduje ich pojawienia się w motywie. Aby wyświetlić ikonę wpisu, należy w pętli użyć funkcji `the_post_thumbnail()`. Jeśli potrzebujesz dodatkowej kontroli, możesz dodatkowo użyć tagu warunkowego `has_post_thumbnail()`.

```
<?php the_post_thumbnail(); ?>
```

Powyższy kod spowoduje wyświetlenie zminiaturyzowanej wersji wysłanego do systemu obrazu. Jeśli w wywołaniu funkcji przekażesz parametr `thumbnail`, `medium` lub `large`, to otrzymasz odpowiednio miniaturę oraz średni lub duży obraz. Jeśli dodałeś dodatkowy rozmiar obrazu, jak wcześniej pokazany `top-feature`, to jego nazwę również możesz przekazać jako parametr. Można też przekazać tablicę określającą wybrane wymiary, na przykład:

```
the_post_thumbnail( array( 320, 130) );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Ikony wpisów są często wyświetlane na początku wpisów i w nagłówkach, aby uatrakcyjnić oprawę wizualną tych wpisów. Aby mieć nad nimi lepszą kontrolę, można przekazać klasę CSS, za pomocą której będzie można podłączyć arkusze stylów:

```
the_post_thumbnail( 'medium', array( 'class' => 'headline' ) );
```

To spowoduje dodanie klasy `headline` do elementu `img` zwróconego przez funkcję `the_post_thumbnail()`. Domyślnie dodawana jest klasa `wp-post-image` i klasa `attachment-X`, gdzie `X` oznacza wybraną wersję obrazu (np. `attachment-large`).

## WŁASNE MENU

Własne menu możesz pamiętać z rozdziału 4., ale żeby nie pozostawiać żadnych wątpliwości, napiszę o nich jeszcze raz tutaj. Włączanie obsługi menu jest bardzo proste. Wystarczy dodać obszar menu do motywu, np. o nazwie Smashing Menu. W pliku *functions.php* należy wpisać poniższy kod:

```
register_nav_menus( array(
    'smashing-menu' => 'Smashing Menu',
) );
```



To wszystko. Teraz wystarczy w wybranym miejscu w motywie wstawić tag szablonowy `wp_nav_menu()`:

```
<?php wp_nav_menu( array( 'theme_location' => 'smashing-menu' ) ); ?>
```

To menu można skonfigurować za pomocą narzędzi dostępnych w panelu administracyjnym, w sekcji *Wygląd-Menu*. Proste, prawda?

## WŁASNE NAGŁÓWKI

Dodanie obsługi własnego nagłówka nie jest trudne, ale wymaga napisania kilku wierszy kodu w pliku *functions.php*. Najpierw należy zdefiniować kolor tekstu nagłówka i jego domyślny obraz. Następnie określa się wysokość i szerokość obrazu (należy użyć haka `after_setup_theme`):

```
define( 'HEADER_TEXTCOLOR', 'ffffff' );
define( 'HEADER_IMAGE', get_template_directory_uri() . '/img/header-default.jpg' );
define( 'HEADER_IMAGE_WIDTH', 940 );
define( 'HEADER_IMAGE_HEIGHT', 130 );
```

Należy też określić sposób wyświetlania nagłówka:

```
// Funkcja nagłówka witryny
function site_header_style() { ?>
    <style type="text/css">
        div#header {
            background: url(<?php header_image(); ?>);
        }
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    </style>
<?php }
```

Jest to prosty kod CSS, w którym tag `header_image()` został użyty jako tło. Tag ten ma tylko jedno zadanie: zwrócenie adresu URL obrazu nagłówka. Można go wykorzystać także w innych celach. Powyższy kod spowodowałby ustawienie wybranego obrazu nagłówka w tle elementu `div#header`. Możesz zmienić ten kod, aby dostosować go do swojego motywu.

Ponieważ nagłówkami zarządza się w panelu administracyjnym, trzeba zadbać, aby w panelu wszystko też dobrze wyglądało:

```
// Funkcja nagłówka w panelu administracyjnym
function admin_header_style() { ?>
    <style type="text/css">
        #headimg {
            width: <?php echo HEADER_IMAGE_WIDTH; ?>px;
            height: <?php echo HEADER_IMAGE_HEIGHT; ?>px;
        }
    </style><?php
}
```

Identyfikator `#headimg` jest używany w panelu administracyjnym, ale nagłówek musi mieć ustawione szerokość i wysokość i stąd te dodatkowe style.



Na koniec trzeba całość włączyć. Bez tego nie będzie możliwości ustawiania własnego nagłówka, a więc nie można zapomnieć o tym drobnym fragmencie kodu:

```
// Włączenie możliwości wybierania własnego obrazu nagłówka
add_custom_image_header( 'site_header_style', 'admin_header_style' );
```

### WŁASNE OBRAZY TŁA

Jak w takim razie włączyć możliwość wybierania własnego obrazu tła? To proste: trzeba tylko dodać do pliku *functions.php* poniższy kod (i najlepiej jak zwykle użyć go w połączeniu z hakiem `after_setup_theme`):

```
// Aktywacja paneli administracyjnych
add_custom_background();
```

To wszystko. Wszelkie ustawienia obrazu czy koloru tła zostaną dodane do `wp_head()` w arkuszu stylów. Włączanie własnych teł musi być proste, prawda?

## HAKI AKCJI

Haki akcji podobnie jak szkielety motywów robią dziś wielką karierę. Myślę, że to właśnie **dzięki** szkieletom są tak popularne, ponieważ szkielety są nimi wypełnione. Ale za bardzo wybiegam do przodu. Najpierw musisz dowiedzieć się, czym są haki akcji.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Motyw, którego używasz, prawie na pewno zawiera jakieś haki akcji. Są nimi znane Ci już funkcje `wp_head` i `wp_footer` używane w plikach *header.php* i *footer.php*. Haki te mają za zadanie umożliwić programiście dodanie kodu wykonywanego podczas ładowania WordPressa lub po jego zakończeniu. Bez przerwy używają ich programiści wtyczek i jestem pewien, że użyto ich też do budowy np. wtyczek do śledzenia statystyk Google Analytics, Woopra i innych. Ale to nie wszystko: wiele haków jest używanych także wtedy, gdy używane są standardowe tagi szablonowe, takie jak np. `the_content()`. Prawdopodobnie możesz użyć haka do większości tagów szablonowych użytych w Twoim motywie.

Można także tworzyć własne haki. Jest to najczęstszy sposób ich użycia przez projektantów szkieletów — dodają haki do różnych części motywu, pozostawiając możliwość wykorzystania ich przez użytkownika bezpośrednio lub w motywie potomnym do zmiany wyglądu witryny i modyfikacji jej funkcjonowania. Haków można używać do wielu różnych celów, od konstruowania menu lub przygotowania miejsca na reklamę po usuwanie elementów z projektu. W pewnym sensie haki pozwalają utworzyć bogato wyposażony projekt, a użytkownicy mogą później zdecydować, których jego części będą używać. Jeśli doda się stronę obsługi tych ustawień w panelu administracyjnym, to później można je zmieniać bez grzebania w kodzie pliku *functions.php*.



Ja jednak zazwyczaj stosuję nieco inne podejście. Podczas gdy perspektywa wyposażenia motywu we wszystkie możliwe funkcje i pozwolenia użytkownikowi zdecydować, które z nich włączyć, jest kusząca, można w ten sposób stworzyć rozdęty i trudny w użytkowaniu produkt. Moim zdaniem dobry szkielet to taki, na którym można coś budować, a nie z którego można coś wyciąć.

Niemniej jednak za pomocą haków można dodawać funkcje poprzez tworzenie własnych haków (wkrótce pokażę, jak to robić) lub wykorzystanie standardowych haków WordPressa. W WordPressie dostępnych jest wiele przydatnych funkcji, jak np. hak `wp_head` (który najczęściej można znaleźć w pliku *header.php*) i `wp_footer`. Jest ich o wiele za dużo, aby wszystkie tu wymienić, dlatego podaję adres, pod którym znajduje się (mam nadzieję aktualna) lista haków: *http://codex.wordpress.org/Plugin_API/Action_Reference*.

## UŻYWANIE HAKÓW

Jeśli chcesz podłączyć się do jednego z haków WordPressa, np. aby dodać statystyki witryny do haka `wp_footer`, musisz to zrobić w pliku *functions.php*. Należy napisać funkcję PHP zawierającą to, co chce się dodać, a następnie funkcję tę dodać do haka.

Zaczniemy od definicji funkcji. Użyty w niej kod śledzenia ruchu na stronie jest oczywiście fikcyjny i jeśli chcesz rzeczywiście użyć tego haka, musisz w jego miejsce wstawić własny kod:

```
<?php function my_webstats() { ?>
    <script Kod śledzenia statystyk />
<?php } ?>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jest to definicja funkcji `my_webstats()` zawierającej błędnie skonstruowany element `script`. Funkcję tę chcemy podłączyć do haka `wp_footer`. Zwróć uwagę na to, co zrobiłem ze znacznikami PHP, aby móc wygodniej pisać kod HTML. Do podłączania funkcji do haków służy funkcja `add_action()`:

```
add_action( 'wp_footer', 'my_webstats' );
```

Pierwszy parametr określa, do którego haka chcemy coś dodać (`wp_footer`), a drugi — co chcemy dodać (`my_webstats`). Wywołanie funkcji `add_action()` powinno znajdować się przed definicją funkcji dodawanej. Poniżej znajduje się kompletny kod tego przykładu:

```
<?php
    add_action( 'wp_footer', 'my_webstats' );
    function my_webstats() {
?>
    <script Kod śledzenia statystyk />
<?php } ?>
```

To spowoduje dodanie kodu statystyk na końcu motywu. Jest w tym trochę czarów WordPressa.

## TWORZENIE WŁASNYCH HAKÓW

Tworzenie własnych haków jest bardzo łatwe. Najpierw należy utworzyć funkcję PHP, podobnie jak poprzednio. Funkcja ta może zawierać wszystko, od instrukcji `echo` po zaawansowane skrypty. Jedyna różnica między własnymi a standardowymi hakami jest taka, że własne haki trzeba dodać do motywu. Haki `wp_head` i `wp_footer` już w nim są, a więc nie trzeba ich dodawać. Jeśli natomiast utworzymy własny hak, to oczywiście musimy go dodać.



Kod do tworzenia nowego haka jest bardzo prosty i wygląda następująco:

```
<?php do_action( 'nazwa-haka' ); ?>
```

Wystarczy wpisać nazwę haka i umieścić ten kod w dowolnym miejscu w motywie. Jeśli np. utworzysz hak o nazwie `welcome-front`, to umieść poniższy kod w dowolnym miejscu, w którym chcesz mieć możliwość podłączania się do haka `welcome_front()`:

```
<?php do_action( 'welcome_front' ); ?>
```

Aby podłączyć się do utworzonego przez siebie haka, należy użyć funkcji `add_action()` w podobny sposób jak wcześniej. W istocie także funkcje `wp_head()` i `wp_footer()` to skrócone formy zapisów `do_action( 'wp_head' )` i `do_action( 'wp_footer' )`.

Podsumowując, funkcja `add_action()` służy do dodawania akcji do haków i najczęściej używa się jej w pliku *functions.php*. Natomiast funkcja `do_action()` służy do tworzenia własnych haków i jest używana w plikach szablonowych motywu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### ODŁĄCZANIE AKCJI OD HAKÓW

Jeśli utworzysz motyw potomny na bazie motywu zawierającego dużą ilość haków, możesz chcieć pozbyć się części z nich. W zależności od sytuacji możesz to zrobić na kilka sposobów, chociaż wszystkie z nich opierają się na tej samej zasadzie. Załóżmy, że chcemy usunąć akcję związaną z hakiem `the_content()` za pomocą funkcji `add_action()`. Do tego celu użylibyśmy funkcji `remove_action()`. Przyjmijmy, że motyw nadrzędny zawiera poniższy fragment kodu dodającego funkcję `superfunction()` do haka `the_content()`:

```
add_action( 'the_content', 'superfunction' );
```

My nie chcemy tej funkcji, a więc ją usuniemy za pomocą funkcji `remove_action()`:

```
remove_action( 'the_content', 'superfunction' );
```

To wszystko! Pamiętaj, żeby konkretnie określić, co chcesz usunąć, i jeśli w funkcji `add_action()` były przekazane jakieś dodatkowe informacje, to w funkcji `remove_action()` również muszą one się znaleźć.

Funkcja `remove_filter()` działa podobnie, ale dotyczy filtrów.



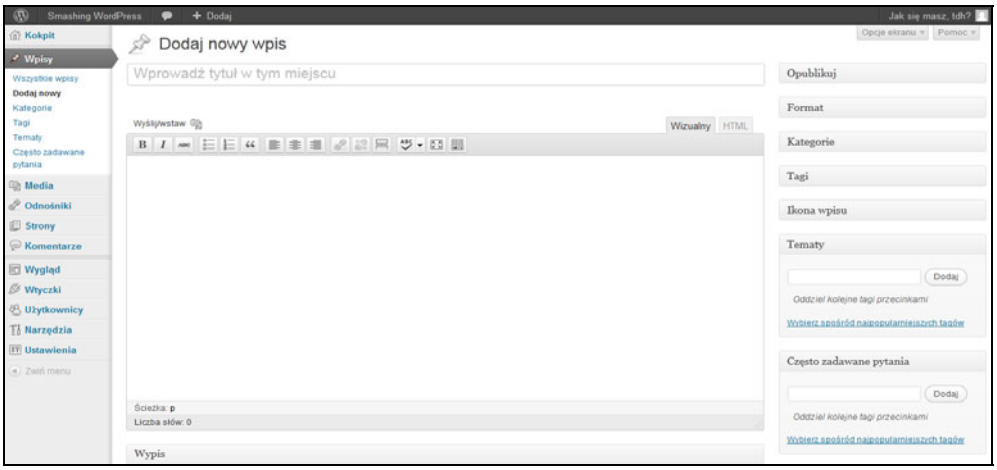
## TAKSONOMIE

Taksonomie to bardzo przydatna funkcja WordPressa. Mówiąc najprościej, są techniką umożliwiającą tworzenie własnych wersji kategorii i tagów. To oznacza, że można mieć np. kilka różnych zestawów tagów. Z każdą taksonomią związane są **terminy** (ang. *term*). Terminami są np. tagi w standardowej taksonomii *Tagi*, podobnie jak kategorie w standardowej taksonomii *Kategorie*.

Własne taksonomie stanowią dodatkowy sposób na kategoryzowanie i oznaczanie wpisów. Można powiedzieć, że tagi i kategorie to dwie standardowe taksonomie WordPressa o nazwach odpowiednio *Tagi* i *Kategorie*. Przypuśćmy, że to nam nie wystarczy i chcielibyśmy mieć dodatkową możliwość oznaczania wpisów tagami o nazwie *Tematy*. Możesz dodać taką taksonomię i wówczas będziesz mógł używać dwóch różnych taksonomii, z których każda będzie miała własny zestaw tagów. Jeśli chcesz, możesz też dodać kolejną taksonomię, np. o nazwie *Często zadawane pytania*. Wówczas będziesz mieć trzy taksonomie do oznaczania wpisów tagami: domyślną i dwie własne (rysunek 6.5).

Oczywiście istnieje też specjalny plik dla taksonomii o nazwie *taxonomy.php*. Można go traktować jak każdy inny plik archiwalny, tzn. powinna znajdować się w nim pętla i można go używać do wyświetlania list wpisów generowanych poprzez kliknięcie odpowiedniego łącza. Jak się zapewne spodziewasz, można też utworzyć pliki *taxonomy-X.php* dla wybranych typów taksonomii oraz pliki *taxonomy-X-Y.php* (*Y* oznacza konkretną wartość taksonomii) dla konkretnych terminów taksonomii, podobnie jak w przypadku tagów i kategorii.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 6.5. Taksonomie tagów

## ZASTOSOWANIA TAKSONOMII

Taksonomie można wykorzystać do wielu celów. Ich zaletą jest to, że niezależnie od tego, czy są hierarchiczne (jak kategorie), czy pozbawione hierarchii (jak tagi), mogą stanowić dodatkową metodę organizacji treści. Oto kilka pomysłów na ich wykorzystanie:



- Grupowanie wpisów bez określonej hierarchii i stylizowanie archiwów tej taksonomii w specyficzny sposób.
- Utworzenie dodatkowej warstwy nawigacyjnej. Powiedzmy, że prowadzimy witrynę internetową o grach wideo i chcemy oznaczać gry według platform, na jakie są przeznaczone, np. Xbox 360, iOS i Wii, ale nie chcemy tego mieszać ze zwykłymi kategoriami lub tagami. Można do tego celu utworzyć własną taksonomię, np. o nazwie *Platforma*.
- Oddzielenie różnych rodzajów terminów. Mając oddzielne taksonomie dla osób, firm i marek, można tworzyć chmury tagów zawierające np. tylko osoby i każde archiwum sformatować w inny sposób.
- Tworzenie własnych formatów wpisów. Formaty wpisów to w istocie funkcja własnych taksonomii, której poświęcono trochę więcej miłości w plikach rdzenia WordPressa. Jeśli chcesz, możesz zapomnieć o standardowych formatach wpisów i utworzyć własne taksonomie dostosowane do Twoich potrzeb.

Możliwości jest bez liku. Dzięki własnym taksonomiom masz znacznie większe możliwości, jeśli chodzi o organizację treści.

## MYŚL

Nie ma wątpliwości, że umiejętność dodawania nowych taksonomii, a w szczególności tworzenia osobnych zbiorów tagów jest bardzo przydatna. Z pewnością przyda się to w witrynach, w których zarządzanie treścią odbywa się prawie tak jak przy bezpośrednim korzystaniu

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

z bazy danych, ale można to wykorzystać także jako alternatywny sposób organizacji treści. W szczególności dodanie nowych taksonomii może umożliwić wykorzystanie kategorii jako funkcji projektowej witryny. Tagi są dużo bardziej precyzyjną techniką znakowania i przy tworzeniu nowych taksonomii wszystkie ich zalety pozostają do naszej dyspozycji. Można na przykład utworzyć taksonomię do tagowania tylko na potrzeby projektu. Tagi z tej taksonomii byłyby używane wyłącznie przez motyw, np. poprzez dodawanie klas do elementu `div` albo `body`. Możliwości takie istnieją, ale nie są dostępne standardowo. Na szczęście bardzo łatwo można je włączyć za pomocą kilku wierszy kodu.

Łatwo też sobie wyobrazić pokrywanie się taksonomii z własnymi polami, ponieważ funkcja ta służy przede wszystkim do sortowania wpisów. Taksonomie z definicji lepiej się do tego nadają. Jednak ich pełen potencjał można wykorzystać dopiero wtedy, gdy do zarządzania mamy duże ilości treści. Bądź co bądź taksonomie są metodą sortowania treści, nic więc dziwnego, że najlepiej sprawdzają się, gdy trzeba coś posortować.

Początkowo możesz mieć problemy z ogarnięciem wszystkich możliwości stwarzanych przez własne taksonomie. Można z nich zrobić naprawdę dobry użytek i jest wiele witryn opartych na WordPressie, w których taksonomie grają ważną rolę. Użytkownik może nie zdawać sobie z tego sprawy, ale programista do osiągnięcia widocznych efektów niekoniecznie musiał napisać mnóstwo wtyczek i męczyć się z niezbyt przyjaznymi własnymi polami. Dlatego nie zapominaj o taksonomiach. W niektórych sytuacjach mogą one być Twoim najlepszym wyborem.



### TAKSONOMIE A PRZENOŚNOŚĆ

Własne taksonomie to doskonałe narzędzie i jestem pewien, że masz już tysiące pomysłów, jak je wykorzystać. Musisz jednak pamiętać, że mimo iż można je tworzyć w pliku *functions.php*, kod ten lepiej zapisać w postaci wtyczki, aby móc ich bezproblemowo używać także po zmianie motywu. Zatem jeśli to tylko możliwe, kod tworzący własne taksonomie implementuj w postaci wtyczek, a ich prezentacją wizualną niech się martwi sam motyw i jego pliki szablonowe.

Więcej na temat przenośności danych piszę w rozdziale 8.

## WŁASNE TYPY WPISÓW

Własne typy wpisów to niezwykle przydatne narzędzie. Dzięki nim nie musisz już ograniczać się tylko do wpisów i stron i możesz tworzyć własne typy dokładnie dostosowane do Twoich potrzeb. Jeśli potrzebujesz typu wpisów o cechach bardzo podobnych do standardowych stron (które również są typem wpisów), ale z możliwością dodawania tagów, to możesz go utworzyć.

Możliwość tworzenia i nazywania typów wpisów istniejących obok zwykłych wpisów i stron jest korzystna jeszcze z jednego powodu: prostoty użytkowania. Od razu wiadomo, co z nimi robić, ponieważ podobnie jak zwykłe wpisy i strony znajdują się w lewej kolumnie w panelu administracyjnym, a ich obsługa jest taka sama jak zwykłych wpisów i stron.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### UŻYWANIE WŁASNYCH TYPÓW WPISÓW

Własne typy wpisów są przydatne w wielu sytuacjach, ale im bardziej chcesz podzielić swoją treść, tym większy będziesz mieć z nich pożytek. Pod wieloma względami to właśnie dzięki nim WordPress stał się dobrym systemem CMS do tworzenia dużych tradycyjnych witryn internetowych. Podczas gdy wcześniej też można było rozwiązać wszystkie problemy, korzystając tylko z wpisów i stron oraz stosując różne sprytne rozwiązania, własne typy wpisów znacznie ułatwiły budowanie witryn zawierających wiele różnych rodzajów treści.

Jednak własne typy wpisów służą nie tylko do tego. Są także wykorzystywane przez wtyczki do przechowywania danych. Mają wiele zastosowań, od wtyczek do wysyłania uwag po wtyczki do tworzenia forów internetowych. Dzięki systemowi szablonów WordPressa tworzenie interesującej treści i ciekawych funkcji w motywach jest bardzo łatwe, ponieważ praca wygląda tak samo jak nad każdą inną częścią witryny. Należy jednak pamiętać, że treść własnych typów wpisów nie jest wyświetlana w pętli na stronie głównej ani w wynikach wyszukiwania, chyba że zmodyfikuje się pętlę.

W rozdziale 14. opisałem wiele ciekawych zastosowań własnych typów wpisów. Wcześniej jednak w następnym podrozdziale, a także kolejnym rozdziale przeczytasz, jak korzystać z tej funkcji we własnych motywach.

### UŻYWANIE WŁASNYCH TYPÓW WPISÓW W MOTYWACH



Własne typy wpisów działają tak samo jak standardowe wpisy, a więc tak samo się z nimi pracuje i w ten sam sposób tworzy się ich warstwę wizualną. Dla archiwów własnych typów wpisów tworzy się pliki o nazwie *archive-X.php*, gdzie *X* jest uproszczoną nazwą typu wpisów (jeśli np. masz typ wpisów o uproszczonej nazwie `produkt`, to plik szablonowy dla tych wpisów powinien mieć nazwę *archive-product.php*). Jest to taki sam sposób działania jak w przypadku archiwów tagów i kategorii. Pojedyncze własne wpisy są obsługiwane przez szablony o nazwie *single-X.php*, gdzie *X* jest uproszczoną nazwą. Jeśli chcesz przypomnieć sobie, które pliki szablonowe są używane w różnych sytuacjach, zajrzyj do rozdziału 4.

Więcej praktycznych przykładów wykorzystania własnych typów wpisów znajdziesz w rozdziale 14. Natomiast w następnym rozdziale nauczysz się je tworzyć.

Są też wtyczki tworzące własne typy wpisów, np. wtyczka bbPress (*http://bbpress.org*) do tworzenia forów internetowych. Dzięki użyciu własnych typów wpisów można łatwo dostosowywać wygląd treści generowanej przez te wtyczki za pomocą plików szablonowych. To bardzo przydatne.

## STRONY OPCJI MOTYWU

Czasami chcemy, aby wszystkie składniki potrzebne do sterowania zachowaniem witryny były zebrane w jednym miejscu. Oczywiście można to zakodować w plikach szablonowych motywu, ale innym popularnym rozwiązaniem jest tworzenie stron opcji motywu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

**Strona opcji motywu** to strona w panelu administracyjnym zawierająca informacje dotyczące używania motywu i opcje pozwalające sterować jego działaniem. Tworzy się ją w całości w pliku *functions.php*. Możliwość tworzenia tych stron jest dostępna już od dłuższego czasu i nawet w standardowym motywie istnieje strona z opcjami pozwalającymi np. wybrać kolor nagłówka.

To, jakie opcje powinny znajdować się na stronie opcji, zależy od rodzaju motywu. W niektórych motywach użytkownik może zmieniać wygląd witryny poprzez zmianę rozmiarów pisma i innych elementów, podczas gdy w innych można zapisywać kod, który zostanie wyświetlony w określonych miejscach witryny dzięki użyciu haków. Zanim utworzysz stronę opcji, zastanów się, jakie opcje na niej umieścisz i czy na pewno jest Ci ona w ogóle potrzebna. Dlatego najpierw usiądź i przemyśl, jakie opcje mogą być potrzebne w tym motywie, a dopiero potem przystąp do realizacji pomysłów. Oczywiście opcje w każdym motywie mogą być inne. W niektórych przydatna może być możliwość zmieniania własności pisma, a w innych — wyświetlania i ukrywania paska bocznego.

Strony opcji motywu tworzy się przy użyciu API ustawień (*http://codex.wordpress.org/Settings_API*), którego opis znajduje się w następnym rozdziale.

## PROBLEMY DOTYCZĄCE OPCJI MOTYWÓW



Oprócz oczywistych problemów, jakie może wywołać błędnie napisany kod, są jeszcze dwie ważne kwestie dotyczące stron opcji motywu i ich obsługi w pliku *functions.php*: walory użytkowe i szybkość.

Zaczniemy od walorów użytkowych. Istnieją motywy zawierające mnóstwo najróżniejszych opcji, których dodanie może się wydawać dobrym pomysłem. Przecież dzięki nim można zrobić wszystko, co się chce, prawda? Może i tak, ale implementacja tego wszystkiego wymaga dużej ilości kodu, a poza tym sam użytkownik może się po prostu pogubić w gąszczu przełączników. Jeśli jest za dużo możliwości do wyboru i niektóre kombinacje źle współdziałają (dotyczy to często różnych ustawień krojów pisma i kolorów), to jakie korzyści można mieć z ich istnienia? Moim zdaniem ilość opcji należy minimalizować. Użytkownicy będą nam za to wdzięczni.

Podsumowując, pliku *functions.php* najlepiej jest użyć, gdy chce się utworzyć niewielką stronę opcji. Jeśli natomiast strona ta ma być rozbudowana, lepiej jest utworzyć osobny plik albo wtyczkę. Wszystko zależy od tego, co chce się osiągnąć. Dlatego zawsze dokładnie przemyśl, czy wszystko robisz jak należy i jeśli plik *functions.php* robi się zbyt długi, to w miarę możliwości przenieś kod (a przynajmniej jego część) w inne miejsce.

Jest jedna rzecz, na której nie należy oszczędzać — lokalizacja, przynajmniej jeżeli planujesz udostępnić swój motyw do użytku innym. Lokalizacja jest tematem następnego podrozdziału.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# OBSŁUGA RÓŻNYCH JĘZYKÓW

Podobnie jak dzięki plikom językowym możesz mieć WordPressa w swoim języku, to samo możesz zrobić ze swoimi motywami i wtyczkami. Tak samo jak jest w przypadku WordPressa, musisz dostarczyć odpowiedni plik *.mo*, który generuje się z pliku POT (ang. *portable object template*) poprzez plik o rozszerzeniu *.po*. Proces przystosowywania motywu do lokalizacji jest dość skomplikowany, ponieważ wymaga użycia kilku niezgranych narzędzi, ale warto się przemęczyć.

Przede wszystkim musisz wiedzieć, jak to się dzieje, że WordPress wczytuje określony plik językowy, gdy plik ten jest dostępny. W menu nie ma żadnego ustawienia pozwalającego wybrać język motywu ani wtyczki, ponieważ system pobiera to ustawienie z opcji `WPLANG` znajdującej się w pliku *wp-config.php* (zobacz rozdział 1.). Jeśli więc masz WordPressa po niemiecku, to w motywie zostaną użyte niemieckie pliki językowe, jeżeli będą dostępne.

Proces tłumaczenia składa się z trzech etapów. Najpierw tworzy się plik POT. Do tego celu używa się specjalnego programu (odnośniki do takich narzędzi znajdziesz w podrozdziale „Praca z plikami językowymi"), który przegląda wszystkie pliki szablonu, pobiera z nich wszystkie łańcuchy oznaczone jako przeznaczone do tłumaczenia i zapisuje je w pliku POT.

Drugi krok polega na utworzeniu pliku *.po* (ang. *portable object*) z pliku POT. W procesie tym następuje zapisanie tekstu oryginalnego z tłumaczeniem w jednym pliku. Etap ten to właściwe tłumaczenie tekstu z jednego języka na inny.



Trzeci etap to utworzenie pliku *.mo* (ang. *machine object*) przy użyciu odpowiedniego programu. Jest to plik w formacie czytelnym dla maszyny i tworzy się go z pliku *.po*. Format ten jest dużo szybciej przetwarzany przez komputer niż zwykły format tekstowy. Plików w formacie *.mo* używa zarówno sam WordPress, jak i Twój motyw.

Ale skąd program generujący pliki POT „wie", co jest przeznaczone do tłumaczenia? To jest zadanie dla programisty motywu lub wtyczki. Musisz ręcznie oznaczyć fragmenty tekstu do tłumaczenia przy użyciu specjalnych konstrukcji języka PHP oraz określić ich dziedzinę. Dziedzina to po prostu zbiór tłumaczeń, który definiuje się w pliku *functions.php*.

Gdybyśmy np. chcieli umożliwić przetłumaczenie tekstu „Hello you!", napisalibyśmy następujący wiersz kodu:

```
<?php _e( 'Hello you!', 'mydomain' ); ?>
```

To spowodowałoby wyświetlenie tekstu „Hello you!" w normalny sposób, chyba że dostępne byłoby tłumaczenie, a w pliku *functions.php* znajdowałaby się definicja dziedziny `mydomain`. Wówczas zostałoby wyświetlone tłumaczenie zamiast angielskiego oryginału.

Zamiast litery `e` można też wpisać drugi znak podkreślenia, tak żeby przed nawiasem znajdowały się dwa takie znaki:

```
<?php __( 'Hello you!', 'mydomain' ); ?>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jako przykładem posłużę się starym dobrym motywem Twenty Eleven. W pliku *index.php* tego motywu znajduje się poniższy kod wyświetlający wiadomość o nieznalezieniu strony:

```
<h1 class="entry-title"><?php _e( 'Nothing Found', 'twentyeleven' ); ?></h1>
```

Jeśli będzie zdefiniowana domena `twentyeleven`, kod ten spowoduje wyświetlenie tekstu *Nothing Found* („Nic nie znaleziono").

Czasami teksty przeznaczone do tłumaczenia są używane jako parametry tagów. Poniżej znajduje się przykład przygotowania do tłumaczenia łącza *Read more* (czytaj dalej) wyświetlanego poprzez umieszczenie tagu `<!-more->` w treści wpisów:

```
<?php the_content( __( 'Read more', 'twentyeleven' ) ); ?>
```

W tym przypadku użyte zostały dwa znaki podkreślenia. Ale dlaczego akurat ta metoda? Funkcja `_e()` jest odpowiednikiem instrukcji `echo`, natomiast `__()` to odpowiednik instrukcji `return`. Wykonywanie instrukcji `echo` w funkcjach PHP może mieć nieprzyjemne skutki, ponieważ instrukcja ta drukuje treść, natomiast `return` tylko ją zwraca do dalszego używania.



Jak w takim razie zmusić motyw do używania pliku z tłumaczeniem? Samo umieszczenie tego pliku wśród innych plików motywu nie wystarczy. Trzeba jeszcze zadeklarować, że ma być używana określona dziedzina (w przedstawionych przykładach `twentyeleven`). W tym celu wystarczy dodać poniższy wiersz kodu do pliku *functions.php* (oczywiście jak zwykle między znacznikami PHP):

```
load_theme_textdomain( 'twentyeleven' );
```

Ten wiersz kodu informuje WordPressa, że:

- motyw jest zinternacjonalizowany i nie ma sensu używać funkcji `load_theme_textdomain()`;
- pod uwagę mają być brane wszystkie przeznaczone do tłumaczenia łańcuchy, które należą do dziedziny `twentyeleven`;
- wciąż potrzebne jest tłumaczenie.

Tłumaczenie tekstu powinno znajdować się w pliku *.mo* wygenerowanym z pliku *.po*, który z kolei utworzony został przy użyciu pliku POT. Plik POT tworzy się przy użyciu specjalnego programu (zobacz następny podrozdział), który sprawdził zawartość wszystkich plików motywu i wydobył z nich łańcuchy oznaczone do tłumaczenia za pomocą funkcji `__()` i `_e()`.

## PRACA Z PLIKAMI JĘZYKOWYMI

Z plikami językowymi można pracować na kilka sposobów. Jednym z najpopularniejszych przeznaczonych do tego programów jest Poedit (*http://www.poedit.net/*), którego można używać w różnych systemach operacyjnych. Program ten umożliwia otwieranie i edytowanie plików *.po* oraz zapisywanie ich w formacie *.mo*, aby mogły być używane w motywie lub wtyczce.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Inne programy tego typu to gettext GNU (*www.gnu.org/software/gettext*) i Launchpad (*https://translations.launchpad.net*). Jednak Poedit jest najłatwiejszy w użyciu z wszystkich, dzięki czemu jest też najczęściej polecany. Jest też ciekawy projekt o nazwie GlotPress (*http://glotpress.org*), ale znajduje się on dopiero we wczesnej fazie rozwoju.

Jeśli opublikujesz wtyczkę lub motyw z plikiem językowym, dodaj do niego także oryginalny plik *.po*, aby użytkownicy mogli sobie go sami przetłumaczyć na swój własny język. Warto do tego zachęcać, ponieważ dzięki temu mogą skorzystać także inni.

### PROBLEM Z NAZWAMI

W niektórych motywach i wtyczkach przystosowanych do tłumaczenia może wystąpić dość niezwykły problem polegający na tym, że system nie może znaleźć motywu lub wtyczki, ponieważ nazwa obiektu została przetłumaczona. Dlatego należy uważać, co się tłumaczy — nie powinno się tłumaczyć żadnych nazw, które są używane w innych miejscach kodu, albo powinno się tłumaczeniem zastąpić wszystkie wystąpienia danej nazwy.

## KANAŁY RSS



Kanały RSS są doskonałym sposobem docierania do czytelników i subskrybowania najświeższych wiadomości w witrynach internetowych. WordPress obsługuje wersje 1.0 i 2.0 standardu RSS. Jest też obsługa starej wersji 0.91, ale polega tylko na przekierowaniu 301 do wersji domyślnej. Ponadto WordPress obsługuje też protokół Atom, ale trzeba go włączyć na stronie ustawień.

Aby włączyć kanały wiadomości dla całej witryny, należy dodać obsługę automatycznego tworzenia łączy kanałów. Poniższy kod należy wpisać w pliku *functions.php*, najlepiej w funkcji podłączonej do haka `after_setup_theme`, podobnie jak to było robione w przykładach dotyczących motywu Simple Blog w rozdziale 4.:

```
add_theme_support( 'automatic-feed-links' );
```

Większość motywów ma wbudowane łącza kanałów i mimo że przeglądarka dyskretnie poinformuje każdego odwiedzającego, że dostępny jest kanał RSS, to jednak warto taki kanał trochę wyróżnić. Przyjrzyj się dowolnej witrynie, która ma dużą liczbę subskrybentów. Od razu zauważysz rzucającą się w oczy ikonę kanału RSS (jeśli potrzebujesz takiej ikony, możesz ją pobrać ze strony *www.feedicons.com*).

Jeśli chcesz mieć wielu subskrybentów, powinieneś pójść tą samą drogą. Lekcja nr 1: ikonę kanału RSS umieść w eksponowanym miejscu, ponieważ inaczej użytkownicy jej nie znajdą i nie użyją.

Lekcja 2. dotyczy zastanowienia się nad ilością treści prezentowanej w kanałach. Możesz w kanale publikować całe wpisy albo tylko ich fragmenty. Kanały z fragmentami wpisów są zazwyczaj mniej popularne, ponieważ czytelnicy wolą otrzymywać całe teksty.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Prawdopodobnie zauważysz, że wielu czytelników subskrybujących fragmenty będzie klikać łącza znajdujące się w kanałach, aby przeczytać cały artykuł, ale po pewnym czasie mogą zacząć rezygnować z subskrypcji. Mówiąc krótko: jeśli naprawdę koniecznie chcesz ściągnąć użytkowników na swoją witrynę, a publikowanie treści w czytnikach RSS temu nie sprzyja, to w porządku. Tylko upewnij się, że wiesz, co robisz, i dlaczego tak postępujesz.

## KANAŁY WORDPRESSA

System WordPress może udostępniać kilka kanałów subskrypcji. Można je łatwo włączyć za pomocą funkcji `bloginfo()`:

```
<?php bloginfo( 'rdf_url' ); ?>
<?php bloginfo( 'rss_url' ); ?>
<?php bloginfo( 'rss2_url' ); ?>
<?php bloginfo( 'atom_url' ); ?>
```

W kodzie tym włączamy po kolei: przestarzały kanał RSS 0.91, który jest tylko przekierowaniem do domyślnego kanału, kanał RSS 1.0, kanał RSS 2.0 oraz kanał Atom.

Można także włączyć kanał wszystkich komentarzy, jak również kanał komentarzy do wybranych wpisów. Służą do tego poniższe funkcje, których należy użyć w motywie:



```
<?php bloginfo( 'comments_rss2_url' ); ?>
<?php comments_rss_link( 'RSS 2.0' ); ?>
```

Drugi z tych wierszy kodu włącza kanał RSS komentarzy do konkretnego wpisu.

Zazwyczaj jednak staram się ograniczać liczbę wywołań PHP do minimum. Wszystkie te kanały można znaleźć także poprzez bezpośredni adres URL. Ponieważ wiesz, jak ważne są bezpośrednie odnośniki (albo zrozumiesz to po lekturze tego rozdziału), na pewno używasz ich w swojej witrynie. Jeśli używasz bezpośrednich odnośników, w Twojej witrynie działają poniższe adresy:

- *http://mojadomena.pl/feed*
- *http://mojadomena.pl/feed/rss*
- *http://mojadomena.pl/feed/rss2*
- *http://mojadomena.pl/feed/rdf*
- *http://mojadomena.pl/feed/atom*

Pierwszy adres dotyczy kanału domyślnego. Można go znaleźć w niektórych motywach, ale jego używanie nie jest najlepszym pomysłem, ponieważ wymaga bezpośrednich odnośników, a nie wszystkie hosty je obsługują. Poniższe adresy natomiast zadziałają zawsze, chociaż nie wyglądają już tak ładnie jak poprzednie:

- *http://mojadomena.pl/?feed=rss*
- *http://mojadomena.pl/?feed=rss2*

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- *http://mojadomena.pl/?feed=rdf*
- *http://mojadomena.pl/?feed=atom*

Ale to nie wszystko! To są tylko adresy głównych kanałów subskrypcji, a w WordPressie dostępne są kanały praktycznie wszystkiego. Są więc np. kanały autorów, kategorii i tagów. Jest nawet kanał dla komentarzy do poszczególnych wpisów. Jeśli włączone są bezpośrednie odnośniki, adresy tych kanałów mają następującą strukturę:

- *http://mojadomena.pl/author/NAZWAUZYTKOWNIKA/feed*
- *http://mojadomena.pl/category/UPROSZCZONANAZWA/feed*
- *http://mojadomena.pl/tag/UPROSZCZONANAZWA/feed*
- *http://mojadomena.pl/BEZPOSREDNIELACZDOWPISU/comments/feed*

Gdybym zatem posługiwał się nazwą użytkownika **superman** w witrynie *http://tdh.me/*, to kanał RSS wszystkich moich wpisów miałby adres *http://tdh.me/author/superman/feed*.

### TWORZENIE WŁASNEGO KANAŁU RSS

Czasami chcemy wykluczyć z kanału RSS jakąś kategorię albo utworzyć kanał uwzględniający tylko kilka tagów. Można tego dokonać, modyfikując adres URL kanału, który może zawierać parametry. Adresy takie są zazwyczaj długie i brzydkie, więc można je następnie przepuścić przez jakąś usługę upiększania adresów.



Powiedzmy na przykład, że z naszego kanału chcemy wykluczyć kategorię o identyfikatorze 47. Adres takiego kanału wyglądałby tak: *http://mojadomena.pl/feed/?cat=-47*.

Ten adres otwiera kanał RSS zawierający wpisy z wszystkich kategorii oprócz tej, która ma identyfikator 47. Kategoria została wykluczona dzięki użyciu znaku `-` przed jej identyfikatorem.

A gdybyśmy chcieli utworzyć kanał RSS dla zapytania do wyszukiwarki? Nic prostszego: *http:// mojadomena.pl/feed/?s=słowokluczowe1+słowokluczowe2*.

No to może jeszcze kanał z wpisami z kategorii 39 i oznaczonymi tagiem o uproszczonej nazwie `ninja`: *http://mydomain.com/feed/?cat=39&tag=ninja*.

Krótko mówiąc, cała praca polega na dodawaniu parametrów do adresu URL. W tabeli 6.1 przedstawiona jest lista parametrów, których można używać.

## PODSTAWOWE KWESTIE SEO

Wyciskanie wszystkiego co się da z wyszukiwarek internetowych to temat na osobną książkę. Okrutna prawda jest taka, że optymalizację witryny pod kątem wyszukiwarek (ang. *search engine optimization* — SEO) najlepiej jest wykonywać dla każdego wpisu i dla każdej strony osobno i automaty raczej się tu nie sprawdzają. Jest jednak kilka rzeczy, jakie możesz zrobić już podczas prac nad motywem, aby był przyjazny dla wyszukiwarek.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Tabela 6.1. Dostępne parametry

| Parametr | Opis |
|---|---|
| `author` | Identyfikator autora |
| `cat` | Numer kategorii |
| `tag` | Uproszczona nazwa tagu |
| `keyword` | Słowa kluczowe do wyszukiwania |
| `year` | Rok, np. 2012 |
| `day` | Dzień, np. 15 |
| `monthnum` | Numer miesiąca, np. 3 (marzec) |
| `hour` | Godzina, np. 19 |
| `minute` | Minuta, np. 45 |
| `second` | Sekunda, np. 13 |
| `p` | Identyfikator wpisu |
| `paged` | Konkretna strona listy wpisów na stronie głównej |

Oto kilka podpowiedzi na temat SEO, które warto wziąć pod uwagę podczas budowy motywu:



- Sprawdzaj poprawność kodu. Poprawny kod jest lepszy. Możesz sprawdzić swoją stronę w serwisie W3C Markup Validation pod adresem *http://validator.w3.org*.
- Stosuj ładne adresy URL. Ładne adresy URL są lepsze dla użytkowników (i wyszukiwarek), ponieważ nie zawierają parametrów, identyfikatorów itp. Najlepiej jest ustawić, aby bezpośrednie łącza zawierały tytuły wpisów i ewentualnie ich kategorie. Niektórzy twierdzą, że także dodanie daty do adresów URL jest korzystne, i może to być prawda, zwłaszcza w często aktualizowanych witrynach. Gdy jednak zmienisz starą strukturę adresów na upiększone adresy, może nastąpić tąpnięcie witryny w wynikach wyszukiwania, dlatego lepiej od razu je ustawić.
- Wykorzystuj tagi i słowa kluczowe: tagi służą nie tylko do opisu treści użytkownikom, lecz również są dobrym sposobem na podpowiedzenie wyszukiwarkom dodatkowych informacji o wpisach. Można użyć wtyczki wstawiającej tagi do elementu `meta keywords` i innych podobnych rozwiązań.
- Utwórz mapę witryny: jednym z najpopularniejszych narzędzi do tworzenia map witryn jest wtyczka Google XML Sitemaps Generator dostępna pod adresem *www.arnebrachhold.de/projects/wordpress-plugins/google-xml-sitemaps-generator*.
- Umieść odpowiednią treść w elemencie `title`. Element `title` powinien zawierać zarówno nazwę wpisu (na pierwszym miejscu), jak i nazwę witryny (na końcu). Można te dwie części czymś rozdzielić.
- Nagłówki: na stronie głównej nazwa witryny powinna znajdować się w elemencie `h1`. Natomiast na stronach i w pojedynczych wpisach w elemencie `h1` powinien znajdować się tytuł strony lub wpisu. Na stronach list archiwalnych (wyszukiwania, kategorii, tagów itd.) w elemencie `h1` zazwyczaj umieszczam tytuł listy, a tytuły poszczególnych publikacji są w elemencie `h2`.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Odnośniki: odnośniki są dobre tylko wtedy, gdy są tematyczne, a więc umieszczanie długiej listy odnośników w blogrollu w pasku bocznym może nie być najlepszym pomysłem. Nie jest to jednak takie proste i trzeba po prostu postępować w sposób przemyślany. Łącza tematyczne zawsze są dobre.
- Podobne wpisy również są tematycznymi łączami: istnieje wiele wtyczek do wyświetlania podobnych wpisów i jeśli rzeczywiście wyświetlają podobne propozycje, to warto z nich korzystać.
- Okruszki: okruszki to łącza, zazwyczaj umieszczone w górnej części strony i pokazujące, w którym miejscu witryny aktualnie znajduje się użytkownik, np. *Home/Reviews/Smashing WordPress*. Odnośniki te są przydatne nie tylko dla użytkownika, ale również dla wyszukiwarek. Okruszki nie są obsługiwane standardowo, dlatego aby ich używać, trzeba zainstalować wtyczkę, np. popularną WordPress SEO ze strony *http://yoast.com/wordpress/seo/*.
- Czas ładowania: nie tylko użytkownicy lubią, gdy strony szybko się otwierają. Dla wyszukiwarek też jest to ważne, a więc postaraj się oczyścić kod swoich stron. Warto też rozważyć możliwość użycia wtyczki buforującej, chociaż tego typu rozwiązania mogą powodować powstanie zduplikowanej treści, a więc najpierw trochę poczytaj, zanim się na to zdecydujesz.

Przejrzyj też listę wtyczek znajdującą się w dodatku A. Znajdziesz tam adresy i porady dotyczące dobrych wtyczek SEO, które pomogą Ci uatrakcyjnić witrynę dla wyszukiwarek. To oczywiste, że motyw powinien być jak najbardziej atrakcyjny dla wyszukiwarek, ponieważ przeważająca część ruchu na stronach pochodzi właśnie z nich.



# POZBYWANIE SIĘ NIEPOTRZEBNYCH RZECZY Z MOTYWU

Każda strona powinna ładować się najszybciej, jak to możliwe. Dużo w tej kwestii można zdziałać w samym motywie, ale trzeba zastanowić się, jak daleko chcemy się posunąć. W ekstremalnym przypadku otrzymasz strony zawierające tylko jeden praktycznie niemożliwy do odczytania wiersz kodu, przez co będziesz musiał utrzymywać drugi plik roboczy, którego będziesz używać do edycji. Dotyczy to w szczególności plików CSS, które można łatwo mocno skompresować i dzięki temu zaoszczędzić kilka bajtów, ale takie pliki bardzo trudno się czyta.

Oto kilka porad, jak przyspieszyć działanie motywu:

- Usuń niepotrzebny kod. Większość motywów warto przejrzeć raz lub dwa razy przed ostatecznym przekazaniem ich do użytku.
- Zminimalizuj wykorzystanie PHP. Nie używaj dynamicznych funkcji i ogólnie unikaj PHP, jeśli spodziewasz się dużego ruchu w witrynie. Co tylko możesz, koduj bezpośrednio w motywie. Czasami spowoduje to ścisłe powiązanie motywu z konkretną witryną, ale nie zawsze to przeszkadza.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Uważaj na wtyczki. Wtyczki są wspaniałym dodatkiem, ale nie każda wtyczka jest dobrze napisana. Każda z nich w najlepszym przypadku dodaje jedno lub kilka wywołań PHP i jakieś zapytania do bazy danych. Dobrze się zastanów, zanim jakąś zainstalujesz.
- Miej na uwadze zewnętrzne usługi. Nie jest to ściśle zależne od samego motywu, ale wszelkie zewnętrzne dodatki, takie jak widżet czatu czy plakietki Twittera, spowalniają ładowanie stron i zmniejszają wydajność Twojej witryny.
- Zoptymalizuj ustawienia serwera. W Twoim hostingu mogą być dostępne akceleratory PHP i funkcje buforowania często używanych części witryny. Mogą działać programy typu Varnish oraz czasami można pogrzebać w ustawieniach serwera, aby przyspieszyć jego działanie. Wszystko zależy od konfiguracji konta hostingowego. Poproś administratora o informacje na ten temat, a potem sprawdź, czy dostępne narzędzia optymalizacji są zgodne z WordPressem.
- Rozważ możliwość użycia wtyczki buforującej. Istnieją wtyczki do WordPressa, które zamiast pobierać dane za każdym razem z bazy danych, serwują użytkownikom statyczne strony. Może to pozytywnie wpłynąć na szybkość działania witryny, a więc warto się nad tym zastanowić.
- Skompresuj pliki. W Google bez trudu znajdziesz narzędzia do kompresji kodu HTML i CSS, które zmniejszą objętość plików, ale też spowodują, że o wiele trudniej będzie się je czytać. Zawsze też dokładnie sprawdź kod zwrócony przez narzędzie kompresujące, aby się upewnić, że kompresor nie dodał do niego żadnego niepożądanego kodu. Nigdy nic nie wiadomo.



W istocie optymalizacja WordPressa wygląda tak samo jak optymalizacja każdej innej witryny. Dobrze napisany, zwięzły i najlepiej poprawny kod zawsze będzie wczytywany szybciej niż niechlujne i rozdęte pliki.

Jeśli mimo wszystkich Twoich starań Twoja witryna nadal działa powoli, zastanów się, czy nie przerosła ona możliwości Twojego hostingu. Bo jeżeli wywołania PHP są ograniczone do minimum, na serwerze działa tylko to, co musi, kod jest zwięzły i poprawny, włączone są narzędzia buforujące i przyspieszające, to pozostaje już tylko zwiększyć moc samego serwera. Choć nie trzeba od razu szukać nowego hostingu, gdy witryna działa powoli, w niektórych przypadkach jest to jedyne rozwiązanie, zwłaszcza jeśli witryna ta szybko się rozrasta.

# MOTYWY A WTYCZKI

Obecnie motyw nie tylko określa wygląd i ogólnie styl witryny, ale zawiera również proste wtyczki. Za pomocą motywu wybiera się, gdzie mają być wyświetlone różne treści, oraz zmienia się normalny układ wpisów w pętli. Za pomocą tagów szablonowych i warunkowych można naprawdę dużo zdziałać. A jeśli doda się do tego plik *functions.php*, haki akcji oraz ogólny spryt programisty, to można osiągnąć praktycznie każdy cel.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jak wspomniałem na początku tego rozdziału, pakowanie wszystkiego do pliku *functions.php* nie zawsze jest dobrym pomysłem. Wiele rzeczy z tego pliku można zaimplementować w postaci wtyczki i podczas gdy czasami wtyczka jest zbędna, to w niektórych sytuacjach może też być najlepszym możliwym wyjściem. Cała idea tworzenia różnego rodzaju skórek w systemach CMS polega na oddzieleniu warstwy projektowej od kodu implementacyjnego. W WordPressie trzeba trochę tę zasadę złamać, jeśli chce się wykonywać niektóre sprytne sztuczki.

Wiele można zdziałać za pomocą wtyczek, które lepiej sprawdzają się w kwestii rozdziału projektu i implementacji niż wpisywanie kodu bezpośrednio w plikach szablonowych. O wiele więcej na temat wtyczek dowiesz się z następnych rozdziałów.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

CZĘŚĆ III

# III

# WTYCZKI DO WORDPRESSA

Rozdział 7. Anatomia wtyczki do WordPressa

Rozdział 8. Wtyczka czy plik functions.php?

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 7

# ANATOMIA WTYCZKI DO WORDPRESSA

**NIE MA WĄTPLIWOŚCI**, że wtyczki nie są tym samym co motywy, choć można znaleźć wiele łączących je podobieństw. Można powiedzieć, że gdy implementuje się jakąkolwiek funkcję w pliku *functions.php*, to w istocie pisze się wtyczkę.

Dzieli je jednak ogromna różnica. Motywy służą do prezentowania treści witryny przy użyciu dostępnych narzędzi. Natomiast wtyczki służą do rozszerzania funkcjonalności WordPressa o dodatkowe funkcje. Należy o tym pamiętać, ponieważ rozbudowywanie pliku *functions.php* w nieskończoność wcale nie jest najlepszym rozwiązaniem.

W tym rozdziale spojrzymy na wtyczki z nieco innej perspektywy niż do tej pory. Za pomocą wtyczki można zrobić wszystko. Ogólnie rzecz biorąc, wtyczki są metodą pozwalającą dodać do WordPressa dowolną funkcję bez żadnych ograniczeń. Porównaj to z kombinowaniem przy użyciu kilku tagów w plikach szablonowych motywu. W przypadku wtyczek kwestią nie jest, **co** można zrobić, lecz **po co** miałoby się coś robić.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# RODZAJE WTYCZEK

Wyróżnia się trzy główne rodzaje wtyczek: zwykłe wtyczki, których na pewno nieraz używałeś, wtyczki do rdzenia (ang. *drop-in*), które zastępują rdzenne funkcje, oraz wtyczki obowiązkowe.

## ZWYKŁE WTYCZKI

Pod pojęciem **zwykłych wtyczek** rozumiem wtyczki, do których używania jesteś przyzwyczajony. Są to wtyczki, które się pobiera z internetu oraz włącza, aby rozpocząć ich używanie. Taką zwykłą wtyczką jest np. Akismet (*http://wordpress.org/extend/plugins/akismet*). Aby taka wtyczka działała, wystarczy ją po prostu włączyć, ewentualnie skonfigurować jakieś drobne ustawienia. Zwykłe wtyczki są przechowywane w folderze *wp-content/plugins/*.

W zasadzie o tym rodzaju wtyczek wszystko już wiesz, a więc przejdźmy do omówienia następnych typów.

## WTYCZKI DO RDZENIA

**Wtyczki do rdzenia** przesłaniają rdzenne funkcje systemu. Umieszcza się je bezpośrednio w folderze *wp-content* pod nazwą odpowiadającą plikowi, który mają zastąpić, np. *advanced-cache.php* albo *db.php*.



Poniżej podano dostępne wtyczki tego typu. Pamiętaj, że jeśli zdecydujesz się na ich użycie, musisz rzeczywiście napisać własny kod zastępujący standardowe skrypty. Jeśli tego nie zrobisz, prawie na pewno będziesz mieć problemy.

- *advanced-cache.php* — własne skrypty zaawansowanego buforowania;
- *db.php* — własna klasa bazy danych;
- *db-error.php* — własne powiadomienia o błędach bazy danych;
- *install.php* — własne skrypty instalacyjne;
- *maintenance.php* — własne wiadomości dotyczące spraw utrzymania serwisu;
- *object-cache.php* — zewnętrzne buforowanie;
- *sunrise.php* — skrypty, które mają zostać wykonane przed załadowaniem sieci witryn;
- *blog-deleted.php* — usuwanie blogów w sieci witryn;
- *blog-inactive.php* — wiadomości o nieaktywnych blogach w sieci witryn;
- *blog-suspended.php* — wiadomości o zawieszonych blogach w sieci witryn.

Wtyczek tych należy używać bardzo ostrożnie, chociaż dają one naprawdę bardzo duże możliwości. Możliwe, że niektórych z nich zdarzało Ci się używać, zapewne jako części innych wtyczek. W szczególności dotyczy to wtyczki *advanced-cache.php*. Zabawa z niektórymi jest mniej ryzykowna, np. wtyczka *maintenance.php* umożliwia wyświetlenie własnej wiadomości na temat uaktualniania instalacji systemu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### WTYCZKI OBOWIĄZKOWE

Wtyczki obowiązkowe różnią się od zwykłych wtyczek. Ich pliki przechowywane są w folderze *wp-content/mu-plugins/* i nie da się ich wyłączyć w panelu administracyjnym WordPressa. Jedyny sposób na ich dezaktywację to usunięcie ich z folderu na serwerze. Wtyczki obowiązkowe nie muszą zawierać specjalnego nagłówka. Bez niego również zostaną wykonane.

W folderze *wp-content/mu-plugins/* można umieścić dowolną wtyczkę, ale z niektórymi wtyczkami mogą być problemy, zwłaszcza w sieciach witryn. Dlatego postępuj ostrożnie. Najlepiej wtyczek używać w sposób zgodny z ich przeznaczeniem.

Wtyczki obowiązkowe są najlepszym rozwiązaniem, gdy chcemy mieć pewność, że jakieś funkcje nie zostaną przypadkowo wyłączone.

## TWORZENIE WTYCZEK DO UŻYTKU W SIECIACH WITRYN

Od WordPressa 3.0 wersja systemu dla wielu użytkowników, zwana **WordPress MU**, stała się częścią podstawowej wersji. Teraz jest to tzw. funkcja tworzenia wielu witryn (ang. *multisite*), dzięki której można tworzyć sieci witryn. Większość wtyczek i motywów dobrze działa w tych sieciach. Problemy mogą się pojawić jedynie wtedy, gdy wtyczka będzie dodawać tabele do bazy danych albo modyfikować istniejące tabele rdzenia. Funkcję sieci witryn włącza się poprzez wpisanie kilku wierszy kodu w pliku *wp-config.php*. Na początek należy dodać poniższy wiersz kodu nad komentarzem `/* To wszystko, zakończ edycję w tym miejscu! Miłego blogowania! */`:



```
define( 'WP_ALLOW_MULTISITE', true );
```

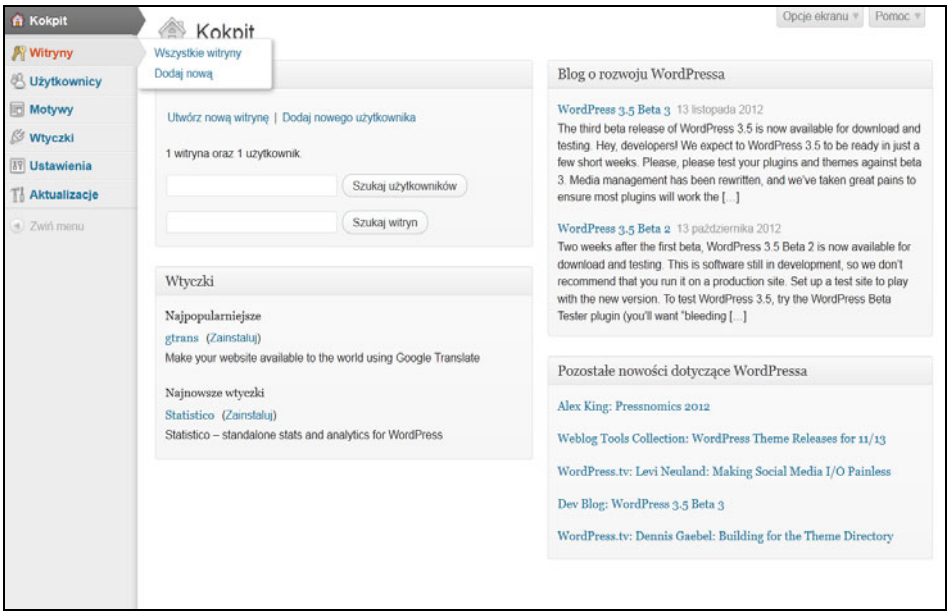
Dodanie tego wiersza kodu do pliku *wp-config.php* spowoduje pojawienie się nowego odnośnika w panelu administracyjnym (rysunek 7.1). Gdy go klikniesz, zostaną wyświetlone proste instrukcje, według których należy postępować, aby uruchomić sieć witryn. Bardziej szczegółowe informacje na ten temat znajdują się na stronie *http://codex.wordpress.org/Create_A_Network*.

Sieć witryn pozwala uruchomić coś w rodzaju nadrzędnego panelu administracyjnego, w którym administrator może zarządzać wieloma witrynami utworzonymi w jego sieci. W serwisie WordPress.com każdy może założyć własnego bloga, ale sieć witryn niekoniecznie musi działać właśnie w ten sposób. Równie dobrze można ją wykorzystać do uruchomienia wielu witryn, nie pozwalając użytkownikom tworzyć własnych blogów.

Tworzenie wtyczek przeznaczonych do użytku w sieciach witryn niewiele różni się od tworzenia zwykłych wtyczek. Największa różnica dotyczy bazy danych i w mniejszym stopniu struktury katalogów.

Jeśli chodzi o strukturę katalogów, to prawie wszystko jest tak samo jak w standardowym WordPressie. Różnica polega na dodaniu katalogu *blogs.dir* do folderu *wp-content*, w którym przechowywane są wszystkie dane utworzonych witryn, takie jak obrazy i inne pliki. Z folderu

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 7.1. Panel administracji siecią witryn



tego nie będziesz często korzystać, ponieważ motywy i wtyczki należą do folderu *wp-content*, tak jak zawsze.

Jeśli chcesz, aby wybrane wtyczki były włączone w całej sieci, możesz je aktywować w panelu administracyjnym sieci. Możesz też skorzystać z wtyczek obowiązkowych, czyli po prostu umieścić wybrane wtyczki w folderze *wp-content/mu-plugins/*, chociaż to może przysporzyć Ci problemów, jeśli któraś z tych wtyczek nie będzie przystosowana do takiego sposobu użycia. Zwykle najlepszym rozwiązaniem jest aktywowanie wtyczki dla całej sieci.

Sam proces powstawania wtyczki wygląda tak samo, jak dla normalnego WordPressa. Trzeba tylko bardziej uważać podczas tworzenia nowych tabel w bazie danych oraz przy pobieraniu treści z tabel rdzenia. W większości przypadków czynności te nie sprawiają kłopotów, ale baza danych dla sieci witryn ma trochę inną strukturę, więc należy uważać.

Kolejną kwestią, jaką należy wziąć pod uwagę podczas pisania wtyczki dla sieci witryn, jest jej przewidywany sposób użycia. Sieć witryn może być prowadzona na wiele różnych sposobów, może być otwarta lub zamknięta, użytkownicy mogą mieć możliwość używania wtyczek lub nie itd. Trzeba to wszystko wziąć pod uwagę, gdy tworzy się wtyczkę.

## WTYCZKI DLA CAŁEJ SIECI

Wtyczki można też włączać dla całej sieci witryn w panelu administracyjnym sieci. W ten sposób można włączać wtyczki znajdujące się w folderze *wp-content/plugins/* dla wszystkich witryn. To jest oczywiście bardzo wygodne rozwiązanie i należy z niego korzystać, zamiast używać wtyczek obowiązkowych. Skoro WordPress może być automatycznie aktualizowany, to im więcej ustawień można zdefiniować kliknięciem w panelu administracyjnym, tym lepiej.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PODSTAWY BUDOWY WTYCZEK

Podstawy budowy wtyczek są podobne do podstaw budowy motywów:

- Główny plik wtyczki musi być w formacie PHP i mieć niepowtarzalną nazwę albo znajdować się w folderze o niepowtarzalnej nazwie, jeśli wtyczka składa się z wielu plików.
- Główny plik PHP wtyczki musi mieć specjalny identyfikujący go nagłówek, podobny do pliku *style.css* w motywach.

Wtyczka może składać się z pliku głównego i wielu plików zawierających różne funkcje, podobnie jak motyw może składać się z wielu plików szablonowych i pliku *style.css*.

---

*Zanim przejdziemy do szczegółowego omawiania podstaw budowy wtyczek, muszę Cię ostrzec, że tworzenie wtyczek jest znacznie bardziej wymagającym zadaniem niż tworzenie motywów. Do tego potrzebna jest solidna znajomość języka PHP i jeśli masz w tej kategorii braki, lepiej trochę się podszkol, zanim zaczniesz pisać jakąś poważniejszą wtyczkę.*

---

Bardzo ważną kwestią jest wybór nazwy pliku lub folderu wtyczki, ponieważ wtyczka ta będzie zapisana w folderze *wp-content/plugins/* z innymi wtyczkami, a więc może dojść do konfliktów nazw. Nazwij zatem swoją wtyczkę w taki sposób, aby ktoś, kto będzie jej szukał na FTP, bez problemu mógł ją znaleźć, znając jej nazwę tylko z panelu administracyjnego WordPressa.



Blok identyfikacyjny wtyczki wygląda znajomo. Poniżej znajduje się przykład takiego bloku:

```
<?php
/*
Plugin Name: Moja wtyczka
Plugin URI: http://url-mojej-wtyczki.com/
Description: Opis mojej wtyczki.
Version: 1.0
Author: Thord Daniel Hedengren
Author URI: http://tdh.me/
*/
?>
```

Tak naprawdę obowiązkowo nagłówek musi zawierać tylko nazwę wtyczki, która w powyższym przykładzie została zapisana w pierwszym wierszu komentarza. Jednak pozostałe informacje również należy dodawać, aby użytkownik mógł przeczytać, co to za wtyczka, skąd pobierać aktualizacje, jaki jest numer wersji itd.

Powinno się także dodać informację o licencji. W dokumentacji WordPressa zalecane jest używanie poniższego standardowego wyciągu z tekstu licencji GPL:

```
<?php
/* Copyright ROK AUTOR (e-mail : E-MAIL AUTORA WTYCZKI)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU
General Public License as published by the Free Software Foundation; either version 2 of the License,
or (at your option) any later version.
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.*

*You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA*
*/*
?>

Oczywiście napisy `ROK`, `AUTOR` i `E-MAIL AUTORA WTYCZKI` należy zastąpić prawdziwymi informacjami. Można także dołączyć cały tekst licencji GPL w pliku o nazwie *license.txt*. Tekst ten jest dostępny pod adresem *www.gnu.org/copyleft/gpl.html*.

To wystarczy, aby WordPress znalazł i poprawnie zidentyfikował wtyczkę. Jeśli umieścisz ją w folderze *wp-content/plugins/*, zostanie wyświetlona na liście wtyczek sekcji *Wtyczki* w panelu administracyjnym systemu. Można ją tam włączyć, aby była dostępna w motywie i akcjach WordPressa.

Od tego momentu zaczyna się prawdziwa zabawa, ponieważ teraz musisz zastanowić się, co wtyczka ma robić i jak sprawić, żeby to robiła.



Niezależnie od tego, czy planowana wtyczka ma zmienić sposób działania WordPressa, czy tylko dodać do niego jakąś funkcję, przed rozpoczęciem pracy powinieneś przejrzeć poniższą listę punktów do sprawdzenia. Dzięki temu możesz zaoszczędzić sobie sporo czasu i problemów.

- Czy taka wtyczka już istnieje? Jeśli tak, zastanów się, czy nie lepiej jej użyć, zamiast tworzyć nową o prawie identycznym działaniu.
- Upewnij się, że nazwa Twojej wtyczki nie jest już zajęta. Sprawdź nie tylko w zasobach WordPress.org, ale również w Google. Dobrym sposobem na utworzenie niepowtarzalnej nazwy jest wstawienie na początku nazwy firmy, np. `acme_nazwawtyczki`.
- Nazwy wszystkich funkcji konsekwentnie zaczynaj od jakiegoś przedrostka. W ten sposób unikniesz konfliktów nazw z innymi funkcjami. To ważne: przedrostki stosuj przed wszystkimi nazwami!
- Czy chcesz umożliwić przetłumaczenie wtyczki na różne języki? Powinieneś. Przygotowywanie wtyczki do internacjonalizacji wygląda tak samo jak w przypadku motywów, a więc jest bardzo łatwe.
- Czy wtyczka będzie tworzyła widżety? Jeśli tak, to jakie ustawienia powinna udostępniać?
- Czy potrzebna będzie strona ustawień w panelu administracyjnym? Staraj się, aby ustawień było jak najmniej, ponieważ użytkownicy wolą proste rozwiązania.
- Na jakiej licencji będzie udostępniana wtyczka? Pamiętaj, że musi być to licencja zgodna z GPL, jeśli chcesz umieścić wtyczkę w zbiorach WordPress.org.
- Na koniec sprawdź, czy nagłówek jest aktualny, czy numer wersji jest poprawny, czy wszystkie łącza działają, czy w paczce są wszystkie pliki oraz czy nie ma błędów w tekście.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# METODY INKORPOROWANIA WTYCZEK

Do budowy wtyczek o wiele częściej używa się zwykłego kodu PHP niż przy budowie motywów. Podczas gdy tagów szablonowych i warunkowych także można używać, to jednak w zdecydowanej większości przypadków będziesz pisać własne funkcje. Oczywiście dużo zależy też od rodzaju tworzonej wtyczki, jednak ogólnie rzecz biorąc, kod wtyczki jest w większości dziełem jej autora, a nie autorów WordPressa.

Działanie wtyczek zwykle opiera się na własnych funkcjach, które programista podłącza do haków i filtrów WordPressa. Umieszczając funkcję w odpowiednim miejscu, np. podłączając do `wp_head` lub komentarzy, możesz ją uruchomić w odpowiednim czasie.

W kolejnych trzech podrozdziałach znajduje się opis trzech sposobów pisania i używania wtyczek. Zaczniemy od jak zawsze ważnych haków.

## UŻYWANIE HAKÓW

Pamiętasz opis haków z poprzedniego rozdziału? **Haki akcji** są wyzwalane przez określone zdarzenia podczas działania WordPressa, np. dla publikacji wpisu jest hak `publish_post`. Drugi rodzaj haków to **haki filtrów**. Są to funkcje, przez które WordPress przekazuje dane, a więc można ich używać do obróbki danych. Do przydatnych haków filtrów zaliczają się np. `the_excerpt` i `the_title`. Trzeba je odróżniać od tagów szablonowych.



Haki są przydatne także przy budowie wtyczek, nie tylko motywów. Dzięki nim można podłączyć wtyczki do odpowiednich części WordPressa, np. `wp_head` albo `wp_footer`. Dobrym tego przykładem jest dodanie funkcji w pliku *functions.php*, a następnie podpięcie jej do haka `wp_footer` za pomocą funkcji `add_action()`, opisane w rozdziale 6.

Podczas budowy wtyczek często pisze się funkcje, które następnie podłącza się do wybranych haków za pomocą funkcji `add_action()`:

```
add_action ( $nazwa_haka, $nazwa_funkcji, $priorytet, $parametry );
```

W miejsce `nazwa_haka` należy wpisać nazwę haka, do którego chcemy dodać naszą akcję. Napotykając ten hak podczas przetwarzania kodu, WordPress sprawdza, czy nie ma dla niego zarejestrowanych żadnych funkcji. Jeśli są, wykonuje je. Funkcja, która zostanie wykonana, jest zdefiniowana w miejscu `nazwa_funkcji`. Na przykład w poniższym fragmencie kodu zostanie wywołana funkcja `smashingshortcode` podczas wykonywania funkcji `wp_head()`:

```
add_action ( 'wp_head', 'smashingshortcode' );
```

Argumenty `$priority` i `$parameters` są nieobowiązkowe. Argument `$priority` jest liczbą całkowitą określającą priorytet (domyślnie ma wartość 10), według którego akcje są sortowane podczas dodawania do haka. Im mniejsza liczba, tym wcześniej dana funkcja zostanie wykonana. Jeśli więc chcesz, aby jedna funkcja została wykonana przed inną funkcją, za pomocą tego atrybutu możesz to ustawić. Natomiast argument `$parameters` określa liczbę argumentów

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

przyjmowanych przez naszą funkcję (domyślna wartość to 1). Jeśli chcesz, aby funkcja przyjmowała więcej niż jeden argument, możesz wpisać w tym miejscu dowolną liczbę całkowitą.

Argumenty priorytetu i liczby argumentów nie są często używane, ale w pewnych przypadkach są naprawdę bardzo przydatne. Ponieważ są opcjonalne, to jeśli się ich nie potrzebuje, można je po prostu opuścić. Jak już jednak napisałem, są sytuacje, w których argument priorytetu jest bardzo przydatny, ponieważ umożliwia uniknięcie konfliktów między wtyczkami. Jeśli masz taki problem, możesz ustawić priorytet swojej wtyczki tak, aby była ładowana jako pierwsza lub ostatnia.

Filtry działają mniej więcej tak samo, tylko zamiast `add_action()` używa się funkcji `add_filter()`. Parametry są takie same i tak samo się je przekazuje. Jedyna różnica polega na tym, że z funkcją `add_filter()` nie używa się haków akcji, tylko filtrów.

Wiemy już, że dodawanie akcji do haków każdego rodzaju jest łatwe, ale co z ich **usuwaniem**? Czasami nie chcemy, aby jakiś hak był uruchamiany, więc musimy go usunąć. Dla haków akcji istnieje funkcja `remove_action()`, a dla haków filtrów — `remove_filter()`. Składnia tych funkcji jest prosta:

```
remove_action( $hook_name, $function_name )
remove_filter( $hook_name, $function_name )
```



Funkcje te służą nie tylko do usuwania własnych funkcji, ale również funkcji samego systemu, a więc za ich pomocą można usunąć praktycznie każdy filtr i każdą akcję, od pingowania po usuwanie załączników. Dlatego niektóre wtyczki mogą tylko usuwać funkcje WordPressa, a nie rozszerzać jego funkcjonalność.

## TWORZENIE WŁASNYCH TAGÓW SZABLONOWYCH

Innym sposobem na uzyskanie dostępu do funkcjonalności wtyczki jest utworzenie własnych tagów szablonowych, jak `bloginfo()` czy `the_title()`. Nie jest to wcale trudne. Wystarczy po prostu utworzyć funkcję we wtyczce (lub w pliku *functions.php*), a następnie ją wywoływać:

```
<?php nazwa_funkcji(); ?>
```

Proste, prawda? Może proste, ale to nie oznacza, że jest to najlepszy sposób dodawania funkcjonalności wtyczki do systemu. W tej metodzie nie musisz tworzyć żadnych haków, funkcja zostanie wykonana w chwili załadowania tagu szablonowego wtyczki, a więc można ją umieścić w plikach szablonowych motywu, gdzie się chce. Jest to szczególnie przydatne, gdy nie ma takiego haka, jaki jest w danej chwili potrzebny. Potem można utworzyć tag szablonowy. Oczywiście lepiej jest jednak używać istniejących haków, kiedy to tylko możliwe.

Zanim zdecydujesz się na takie rozwiązanie, powinieneś zastanowić się nad kwestią użyteczności. Nie każdy lubi i potrafi modyfikować pliki szablonowe, dlatego zwłaszcza jeśli planujesz przekazać wtyczkę do ogólnego użytku albo klientowi, zmuszanie użytkowników do grzebania w szablonach nie będzie najlepszym pomysłem. Jeśli wtyczki używać będziesz

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

tylko Ty, nie ma to żadnego znaczenia. Jeśli jednak inni użytkownicy będą samodzielnie wybierać miejsce do umieszczenia wtyczki albo będą zmieniać jej parametry, to powinieneś poszukać innego rozwiązania.

Jednak w niektórych sytuacjach samo dodanie tagu szablonowego nie wystarczy i trzeba nadpisać część funkcjonalności systemu. Wówczas należy użyć specjalnych funkcji nadpisujących (ang. *pluggable functions*).

### FUNKCJE NADPISUJĄCE

Czasami trzeba nadpisać wybrane części rdzenia WordPressa, aby np. zastąpić je własnymi rozwiązaniami albo żeby po prostu się ich pozbyć, ponieważ chcemy używać WordPressa w niestandardowy sposób. Może nie chcesz, aby w panelu administracyjnym działały lokalizacje (wówczas pozbądź się funkcji `load_textdomain()`), albo chcesz zmienić stopkę panelu administracyjnego na własną. Tych rzeczy nie da się zrobić, usuwając tylko wybrany hak. Z tego typu problemami trzeba zwrócić się do pliku *pluggable.php* znajdującego się w folderze *wp-includes*. Oczywiście nie będziemy go modyfikować, bo mielibyśmy z nim same kłopoty przy każdej aktualizacji WordPressa. Zamiast tego napiszemy wtyczkę, która będzie go przesłaniać. Miej świadomość, że jest to niebezpieczna praca. Przede wszystkim każdą funkcję można nadpisać tylko raz, a więc jeśli dwie wtyczki nadpiszą tę samą funkcję w pliku *pluggable.php*, witryna w najlepszym wypadku będzie źle działać, a w najgorszym przestanie działać w ogóle. To oznacza, że nie można zainstalować dwóch wtyczek przesłaniających tę samą funkcję w pliku *pluggable.php*, co jest poważną wadą systemu. Z tego powodu funkcji nadpisujących najlepiej jest używać wyłącznie w witrynach, nad którymi ma się pełną kontrolę.



Aby zapobiec wyświetlaniu niepotrzebnych powiadomień o błędach, dodatkowo można kod wtyczki umieścić w instrukcji warunkowej sprawdzającej, czy dana funkcja istnieje:

```
<?php if ( ! function_exists( 'function_name' ) ); ?>
```

Oczywiście funkcje nadpisujące są czasami przydatne. Aktualna lista funkcji, które można przesłaniać, znajduje się w dokumentacji na stronie *http://codex.wordpress.org/Pluggable_Functions*.

Pamiętaj, że samo poprawne napisanie wtyczki, o czym jest mowa dalej, nie wystarczy do zapewnienia jej poprawnego działania, gdy wtyczka ta usuwa wybrane części rdzenia WordPressa. Nie dziw się, jeśli coś przestanie działać albo pojawią się jakieś konflikty, których nigdy byś się nie spodziewał. Bądź co bądź funkcje WordPressa, które próbujesz obejść, po coś przecież są.

## WŁASNE TAKSONOMIE I TYPY WPISÓW

Własne taksonomie i typy wpisów to bardzo przydatne narzędzia, zwłaszcza gdy planuje się używanie WordPressa do prowadzenia czegoś więcej niż prosty blog. Jeśli nie wiesz, do czego mogą Ci się one przydać, ciekawe przykłady ich zastosowań znajdziesz w rozdziale 14.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Dla przypomnienia: własne taksonomie są dodatkowym sposobem organizacji treści. Domyślne tagi i kategorie są przykładami taksonomii odpowiednio niehierarchicznej i hierarchicznej, natomiast konkretne tagi i kategorie to terminy tych taksonomii. Natomiast własne typy wpisów są dodatkowym rodzajem publikacji, podobnie jak wpisy i strony, które też są rodzajem wpisów.

## POWODY, ABY UŻYĆ WTYCZKI

Powodem, dla którego do tworzenia własnych taksonomii i typów wpisów najlepiej używać wtyczek, jest przenośność. Jeśli kod umieścisz w pliku *functions.php* (co jest możliwe), to po przenosinach do innego motywu zostanie on utracony, chociaż oczywiście można go skopiować do tego nowego motywu. We wtyczkach powinno się implementować raczej funkcje dotyczące treści niż projektu, dzięki czemu można ich używać w różnych motywach. Wystarczy włączyć wtyczkę i gotowe.

Więcej na temat przenośności danych piszę w rozdziale 8.

## TWORZENIE WŁASNEJ TAKSONOMII



Aby utworzyć własną taksonomię, należy napisać funkcję zawierającą wywołanie funkcji `register_taxonomy()` i związać ją z hakiem `init`. Ustawienia funkcji `register_taxonomy()` są proste i dotyczą sposobu prezentacji taksonomii w panelu administracyjnym oraz tego, czy taksonomia powinna mieć własny bezpośredni odnośnik (argument `rewrite`), czy ma mieć strukturę hierarchiczną itd.

Poniżej znajduje się przykład tworzenia taksonomii *Moje tagi*, która jest niehierarchiczna, a więc bardzo podobna do domyślnej taksonomii tagów (rysunek 7.2):

```
// Powiązanie funkcji z hakiem init
add_action( 'init', 'smashing_tax', 0 );

// Funkcja taksonomii
function smashing_tax() {

    // Rejestracja taksonomii
    register_taxonomy( 'smashing_taxonomy', 'post',
        array(
            'hierarchical' => false,
            'labels' => array(
                'name' => 'Moje tagi',
                'singular_name' => 'Moje tagi',
                'search_items' => 'Przeszukuj Moje tagi',
                'popular_items' => 'Popularne Moje tagi',
                ↪'add_new_item' => 'Dodaj nowy Mój tag'
            ),
            'query_var' => true,
            'rewrite' => true
        )
    );

}
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Rysunek 7.2. Taksonomia Moje tagi

Niezbyt skomplikowane, prawda? Kod ten możesz zapisać w pliku *functions.php*, ale zapewne Twoja nowa taksonomia nie jest związana z konkretnym motywem, tylko z treścią, a więc lepiej byłoby utworzyć wtyczkę. W tym celu wystarczy przenieść ten kod do nowego pliku PHP zawierającego na początku odpowiedni nagłówek. Poniżej znajduje się treść takiego pliku:

```
<?php
/*
Plugin Name: Moje tagi
Plugin URI: http://tdh.me/wordpress/moje-tagi/
Description: Dodaje taksonomię Moje tagi.
Version: 1.0
Author: Thord Daniel Hedengren
Author URI: http://tdh.me/
*/
// Powiązanie funkcji z hakiem init
add_action( 'init', 'smashing_tax', 0 );

// Funkcja taksonomii
function smashing_tax() {

    // Rejestracja taksonomii
    register_taxonomy( 'smashing_taxonomy', 'post',
        array(
            'hierarchical' => false,
            'labels' => array(
                'name' => 'Moje tagi',
                'singular_name' => 'Moje tagi',
                'search_items' => 'Przeszukuj Moje tagi',
                'popular_items' => 'Popularne Moje tagi',
                 ↪_new_item' => 'Dodaj nowy Mój tag'
            ),
            'query_var' => true,
            'rewrite' => true
        )
    );

}
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## TWORZENIE WŁASNEGO TYPU WPISÓW

Własne typy wpisów również tworzy się w prosty sposób. Mimo że je również można tworzyć we wtyczkach (istnieją nawet wtyczki dodające interfejs pozwalający tworzyć własne typy wpisów), tutaj skoncentruję się na robieniu tego bezpośrednio w motywie. Kluczowe znaczenie w tym przypadku ma funkcja `register_post_type()`:

```
// Rejestracja nowego typu wpisów
register_post_type( 'moj_typ_wpisow',
    array(
         'labels' => array(
             'name' => 'Mój typ wpisów'
        ),
         'singular_label' => 'Mój typ wpisów',
          'public' => true,
          'show_ui' => true,
          'capability_type' => 'post',
          'has_archive' => true,
          'hierarchical' => false,
          'show_in_menu' => true,
          'supports' => array( 'title', 'editor', 'author', 'revisions', 'comments'
)
    )
);
```



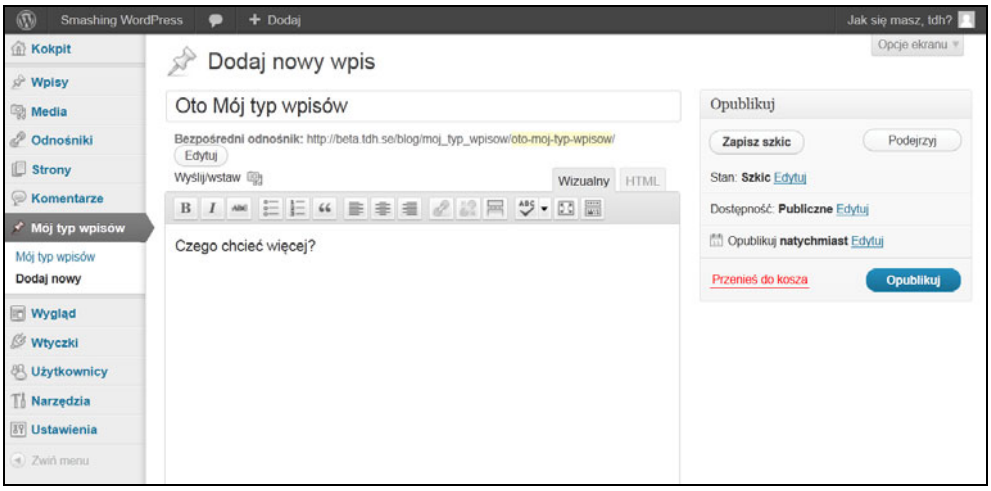
To spowoduje zarejestrowanie nowego typu wpisów o nazwie `moj_typ_wpisow` określonej w pierwszym parametrze. Drugi parametr to tablica zawierająca ustawienia wyglądu i właściwości tego typu wpisów, np. treść etykiet w panelu administracyjnym, czy ten typ jest publiczny, czy ma on być wyświetlany jako opcja w menu i które role użytkowników mogą z niego korzystać. Większości tych ustawień nie trzeba objaśniać, ale warto zwrócić uwagę na tablicę `supports` znajdującą się w tablicy ustawień. W niej definiuje się, co dany typ wpisów obsługuje, tu: tytuł, edytora tekstu, możliwość wyboru autora, wersje wpisu oraz komentarze. Żadnych taksonomii, wypisów, własnych pól — tylko to, co zostało wpisane.

Aby nowy typ wpisów pojawił się w panelu administracyjnym, wystarczy powyższy kod wkleić do pliku *functions.php* (rysunek 7.3). Prawdopodobnie będzie trzeba ponownie wygenerować wszystkie bezpośrednie odnośniki, aby adresy wpisów nowego typu zaczęły działać.

Treść własnych typów wpisów nie jest uwzględniana przez standardową pętlę. Aby je wyświetlić, trzeba tę pętlę zmodyfikować za pomocą funkcji `query_posts()` lub stosując jakąś inną metodę — więcej informacji o pętli znajduje się w rozdziale 3. Ale jeśli masz nowy typ wpisów, możesz bez przeszkód umieścić w menu łącze do strony ich archiwum.

Istnieje wiele opcji tworzenia wpisów. Ich kompletną listę można znaleźć w dokumentacji WordPressa na stronie *http://codex.wordpress.org/Function_Reference/register_post_type*. Własnych typów wpisów będziemy używać w projektach opisanych w rozdziale 14., a więc jeśli chcesz dowiedzieć się więcej na temat ich możliwości, zajrzyj do tego rozdziału.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 7.3. Nowy typ wpisów

Jak kod tworzący własny typ wpisów wyglądałby we wtyczce? Bardzo podobnie do własnej taksonomii. Oto wtyczka Smashing Post Type w całej okazałości:

```
<?php
/*
Plugin Name: Smashing Post Type
Plugin URI: http://tdh.me/wordpress/smashing-post-type/
Description: Adding the Smashing Post Type.
Version: 1.0
Author: Thord Daniel Hedengren
Author URI: http://tdh.me/
*/

// Dodanie do haka init
add_action( 'init', 'moj_typ_wpisow' );

// Dodanie własnych typów wpisów
function smashing_post_types() {

    // Rejestracja Mojego typu wpisów
    register_post_type( 'moj_typ_wpisow',
        array(
            'labels' => array(
                'name' => 'Mój typ wpisów',
                'menu_name' => 'Moje wpisy'
            ),
            'singular_label' => 'Mój typ wpisów',
            'public' => true,
            'show_ui' => true,
            'menu_position' => 5,
            'capability_type' => 'post',
            'has_archive' => true,
            'hierarchical' => false,
            'show_in_menu' => true,
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            'supports' => array( 'title', 'editor', 'author', 'revisions', 'comments' )
        )
    );

}

?>
```

# CO POWINNA MIEĆ KAŻDA WTYCZKA

Tak naprawdę jedyną obowiązkową rzeczą, jaką musi mieć każda wtyczka, jest plik PHP zawierający nagłówek identyfikacyjny oraz kod potrzebny do wykonania zadania. W rzeczywistości jednak powinno się troszkę więcej popracować. Przecież wtyczki może używać jeszcze ktoś inny, a więc powinna ona być jak najłatwiejsza w użyciu i jak najbardziej dostępna. A to oznacza, że należy dołożyć wszelkich starań, aby wtopić wtyczkę w panel administracyjny WordPressa.

To samo dotyczy wszystkich elementów widocznych dla użytkownika witryny. Niektóre wtyczki dodają elementy wizualne i jeśli wtyczka ma być udostępniana publicznie, elementy te powinny nadawać się do wyświetlenia w jak największej ilości motywów. Oczywiście kwestii tej nie ma w przypadku wtyczek pisanych tylko dla konkretnych projektów. Podobnie jest z lokalizacją. Jeśli nie planujesz dodawać obsługi innych języków, nie ma sensu przygotowywać wtyczki do lokalizacji.



Moim zdaniem każda wtyczka powinna mieć dołączoną licencję i instrukcję obsługi. Nie mam nic przeciwko plikom *readme.txt*, ale wielu użytkowników nie otwiera ich, bo im się nie chce albo są zbyt niecierpliwi. Dlatego dobrze jest, jeśli podstawowe instrukcje znajdują się w samej wtyczce. Można też dodać informacje na karcie Pomoc WordPressa za pomocą funkcji `add_help_tab()`. Więcej informacji na ten temat znajduje się na stronie *http://codex.wordpress.org/Function_Reference/add_help_tab*.

## USTAWIENIA WTYCZEK

Czasami trzeba zapisać jakieś informacje w bazie danych. Jeśli chodzi o bazę danych, to możesz z nią robić wszystko, co można zrobić przy użyciu skryptów PHP, a więc dodawać tabele itd. Nie będę się o tym rozpisywał.

Pokażę Ci natomiast, jak używać API ustawień (więcej informacji znajdziesz na stronie *http://codex.wordpress.org/Settings_API*), aby zlecić zapisywanie danych ustawień WordPressowi. To nie tylko pozwala zaoszczędzić mnóstwo czasu, ale również jest bezpieczne, ponieważ w WordPressie stosowane są różne zabezpieczenia, takie jak np. nonce (ang. *number used once* — liczba użyta tylko raz).

W celach testowych utworzymy prostą stronę ustawień, którą dodamy do menu *Ustawienia* na lewym pasku bocznym panelu administracyjnego. Strona ta będzie zawierała pole tekstowe i pole wyboru.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Najpierw musimy utworzyć wtyczkę. Wystarczy nam zwykły plik PHP o nazwie *smashing-settings.php* zaczynający się od poniższego nagłówka:

```
<?php
/*
Plugin Name: Ustawienia Smashing
Plugin URI: http://tdh.me/wordpress/ustawienia-settings/
Description: Prosta wtyczka ustawień.
Author: Thord Daniel Hedengren
Author URI: http://tdh.me/
*/
```

Teraz dodamy stronę ustawień do menu *Ustawienia*. W tym celu podłączymy funkcję do haka `admin_menu`. W tym przykładzie użyjemy funkcji `add_options_page()`, ale są też inne funkcje do wyboru:

```
// Dodaje funkcję strony ustawień do menu
add_action( 'admin_menu', 'smashings_settingsdemo_add_page' );

// Dodanie do menu
function smashings_settingsdemo_add_page() {
    add_options_page( 'Przykład ustawień Smashing',
         'Ustawienia Smashing',
         'manage_options',
         'smashings_settingsdemo',
         'smashings_settingsdemo_do_page' );
}
```



Za pomocą funkcji `add_options_page()` utworzyliśmy stronę o nazwie *Przykład ustawień Smashing*, która w menu po lewej stronie jest skrócona do *Ustawienia Smashing*. Strona ta jest dostępna tylko dla użytkowników mających rolę `manage_options`. Identyfikator tej strony to `smashings_settingsdemo`. Ponadto treść samej strony jest tworzona przez funkcję `smashings_settingsdemo_do_page()`, którą musimy jeszcze napisać:

```
//Rzeczywiste dodanie strony z ustawieniami
function smashings_settingsdemo_do_page() {
// Opuszczamy PHP na moment ?>

    <h2>Ustawienia Smashing</h2>
    <p>To jest nasza strona ustawień.</p>
    <form action="options.php" method="post">
    <?php settings_fields( 'smashings_settingsdemo' ); ?>
    <?php do_settings_sections( 'smashings_settingsdemo' ); ?>
    <?php submit_button(); ?>
    </form>

<?php
} // Powrót do PHP
```

Jest to cały kod HTML strony ustawień — na razie jest go niewiele, ale powoli go rozbudujemy. Zwróć uwagę na funkcję `settings_fields()` wskazującą pole ustawień o nazwie `smashings_settingsdemo` utworzone za pomocą funkcji `add_settings_field()`

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

oraz funkcję `do_settings_sections()`, która wyświetla sekcję o podanej nazwie. Nie trzeba też samodzielnie tworzyć pliku zatwierdzania formularza, ponieważ można go wstawić za pomocą wywołania `submit_button()`.

Następnie tworzymy funkcję dla naszych ustawień i dodajemy sekcję ustawień za pomocą funkcji `add_settings_section()` oraz pole wejściowe za pomocą funkcji `add_settings_field()`:

```
// Funkcja ustawień
function smashings_settingsdemo_init(){

    // Dodanie sekcji
    add_settings_section('smashing_settings_section',
         'Smashing Settings',
         'smashing_settings_section_callback',
         'smashings_settingsdemo');

    // Dodanie pola ustawień
    add_settings_field('smashing_sample_input',
         'Input sample',
         'smashing_sample_input_callback',
         'smashings_settingsdemo',
         'smashing_settings_section');

    // Rejestracja ustawień
    register_setting( 'smashings_settingsdemo', 'smashing_sample_input',
         'smashing_settingsdemo_validate' );
}

// Inicjacja smashings_settingsdemo_init() w panelu administracyjnym
add_action( 'admin_init', 'smashings_settingsdemo_init' );
```



Nie jest to wcale takie skomplikowane. Najpierw przy użyciu funkcji `add_settings_section()` tworzymy sekcję o nazwie `smashing_settings_section`. Zwróć uwagę na wartość `smashing_settingsdemo` będącą parametrem `$page`. Używaliśmy jej już w funkcji `do_settings_sections()` w kodzie strony. Do nowo utworzonej sekcji dodajemy pole ustawień `smashing_sample_input`, w ostatnim parametrze funkcji `add_settings_field()` wpisując `smashing_settings_section`. Listy wszystkich parametrów funkcji `add_settings_section()` i `add_settings_field()` znajdują się w dokumentacji na stronach *http://codex.wordpress.org/Function_Reference/add_settings_section* i *http://codex.wordpress.org/Function_Reference/add_settings_field*.

Następnie dodajemy funkcję `smashings_settingsdemo_init()` do haka `admin_init`.

Następnie dodamy coś do sekcji tylko po to, aby pokazać, że jest to możliwe:

```
// Funkcja wykonywana na początku sekcji
function smashing_settings_section_callback() {
    echo '<p>Informacja na początku sekcji.</p>';
}
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Nazwę funkcji `smashing_settings_section_callback()` widziałeś już w funkcji `add_settings_section()`. Jest to funkcja zwrotna, która będzie wywoływana na początku sekcji i będzie wyświetlała element `p` z tekstem.

W podobny sposób utworzymy funkcję zwrotną dla funkcji `add_settings_field()` o nazwie `smashing_sample_input_callback()`:

```
// Implementacja funkcji smashing_sample_input_callback()
function smashing_sample_input_callback() {
// Opuszczamy PHP na chwilę ?>
    <input type="text" name="smashing_sample_input" value="<?php echo
      get_option( 'smashing_sample_input' ); ?>" />
<?php }
// Powrót do PHP
```

Funkcja ta zawiera tylko pole wejściowe, którego wartość `smashing_sample_input` jest przekazywana przez funkcję `get_option()`. Wartość tę będziemy zapisywać i jest ona identyfikatorem pola, które utworzyliśmy.

Na koniec musimy trochę oczyścić dane wprowadzane do formularza. W tym przykładzie użyjemy tylko funkcji `esc_attr()`, ale jeśli chcesz, możesz skorzystać też z innych metod oczyszczających:



```
// Oczyszczanie
function smashing_settingsdemo_validate($input) {

    // Kodowanie
    $newinput = esc_attr($input);
    return $newinput;
}
```

Co robi ten kod? Utworzyliśmy stronę ustawień zawierającą pola ustawień tworzone za pomocą funkcji `settings_fields()` (w tym przykładzie tylko jedno, ale można dodać więcej) i sekcje utworzone za pomocą funkcji `do_settings_section()`. Poniżej znajduje się kompletny kod wtyczki.

```
<?php
/*
Plugin Name: Ustawienia Smashing
Plugin URI: http://tdh.me/wordpress/ustawienia-settings/
Description: Prosta wtyczka ustawień.
Author: Thord Daniel Hedengren
Author URI: http://tdh.me/
*/

// Dodaje stronę ustawień do menu
add_action( 'admin_menu', 'smashings_settingsdemo_add_page' );

// Dodanie do menu
function smashings_settingsdemo_add_page() {
    add_options_page( 'Przykład ustawień Smashing',
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

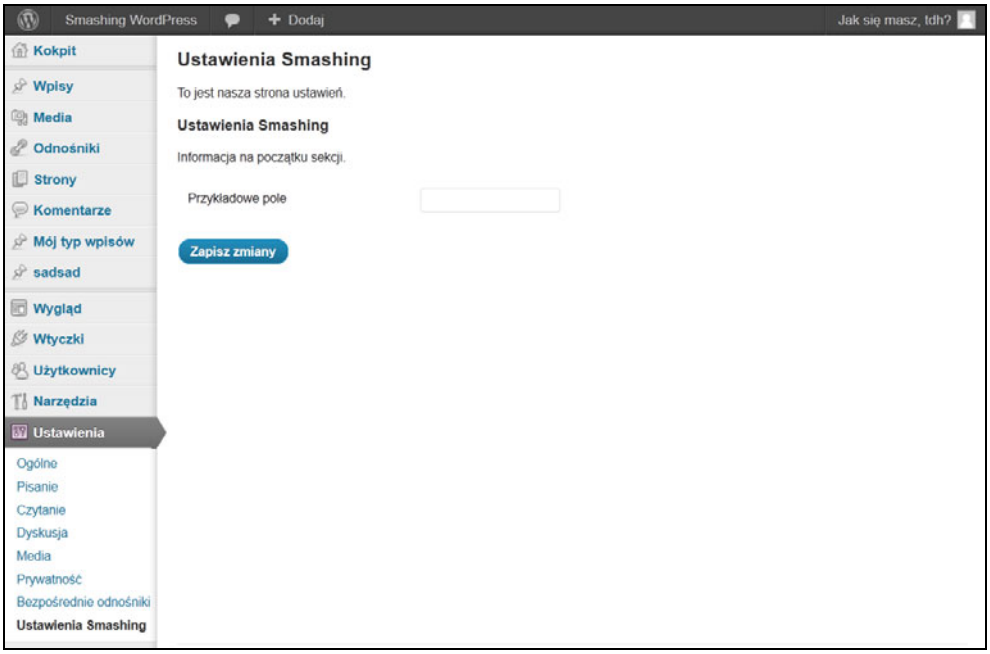
```
        'Ustawienia Smashing',
        'manage_options',
        'smashings_settingsdemo',
        'smashings_settingsdemo_do_page' );
}

// Rzeczywiste dodanie strony ustawień
function smashings_settingsdemo_do_page() {
// Opuszczamy PHP na moment ?>

    <h2>Ustawienia Smashing</h2>
    <p>To jest nasza strona ustawień.</p>
    <form action="options.php" method="post">
    <?php settings_fields( 'smashings_settingsdemo' ); ?>
    <?php do_settings_sections( 'smashings_settingsdemo' ); ?>
    <?php submit_button(); ?>
    </form>

<?php
} // Powrót do PHP

// Funkcja ustawień
function smashings_settingsdemo_init(){

    // Dodanie sekcji
    add_settings_section('smashing_settings_section',
        'Ustawienia Smashing',
        'smashing_settings_section_callback',
        'smashings_settingsdemo');

    // Dodanie pola ustawień
    add_settings_field('smashing_sample_input',
        'Przykładowe pole',
        'smashing_sample_input_callback',
        'smashings_settingsdemo',
        'smashing_settings_section');

    // Rejestracja ustawień
    register_setting( 'smashings_settingsdemo', 'smashing_sample_input',
        'smashing_settingsdemo_validate' );
}

// Inicjacja smashings_settingsdemo_init() w panelu administracyjnym
add_action( 'admin_init', 'smashings_settingsdemo_init' );

// Funkcja wykonywana na początku sekcji
function smashing_settings_section_callback() {
    echo '<p> Informacja na początku sekcji.</p>';
}

// Implementacja funkcji smashing_sample_input_callback()
function smashing_sample_input_callback() {
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
// Opuszczamy PHP na moment ?>
    <input type="text" name="smashing_sample_input" value="<?php echo
      get_option( 'smashing_sample_input' ); ?>" />
<?php }
// Powrót do PHP

// Oczyszczanie
function smashing_settingsdemo_validate($input) {

    // Kodowanie
    $newinput = esc_attr($input);
    return $newinput;

}

?>
```

W ten sposób utworzyliśmy bardzo prostą stronę ustawień widoczną na rysunku 7.4. Za jej pomocą można zapisywać ustawienia w bazie danych.




Rysunek 7.4. Prosta strona ustawień z tekstem i polem tekstowym

## BAZA DANYCH A ODINSTALOWYWANIE WTYCZKI

Tworząc wtyczkę przechowującą w bazie danych jakieś informacje, należy zastanowić się, co zrobić, gdy ktoś, kto ją zainstalował, zechce z niej po pewnym czasie zrezygnować. Czy wtyczka powinna po sobie sprzątać? W większości przypadków tak, zwłaszcza jeśli zapisała w bazie danych sporo informacji, które nie powinny tam zalegać bezużytecznie.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jest kilka sposobów na usunięcie niepotrzebnych danych. Jednym z nich jest utworzenie pliku *uninstall.php* zawierającego kod usuwający z bazy danych treść, która została dodana tam przez wtyczkę:

```
delete_option( 'my-data' );
```

Powyższa instrukcja spowoduje usunięcie pola `my-data` z tabeli `option` bazy danych. Jako że wiele wtyczek zapisuje różne opcje w bazie danych, tabela ta szybko może stać się bardzo zabałaganiona, a to niczemu dobremu nie służy. Oczywiście w swoim pliku *uninstall.php* powinieneś zapisać własne instrukcje usuwające niepotrzebne dane. To samo dotyczy deinstalacji wykonywanych poprzez panel administracyjny.

Oto przykładowa zawartość pliku *uninstall.php*:

```
<?php
    // Dla starych wersji
    if ( !defined( 'WP_UNINSTALL_PLUGIN' ) ) {
        exit();
    }
    // Usuwanie danych opcji
    delete_option( 'myplugindata_post' );
    delete_option( 'myplugindata_feed' );
?>
```



W pierwszej części skryptu sprawdzamy, czy funkcja deinstalacji jest dostępna. W starszych wersjach WordPressa nie było jej, a więc jeśli używasz systemu w wersji starszej niż 3.0 (choć nie powinieneś!), skrypt zakończy działanie, nic nie robiąc. Dzięki temu ten plik jest zgodny ze starszymi wersjami WordPressa. Pozostały kod służy do usuwania danych z bazy: `myplugindata_post` i `myplugindata_feed`. Instrukcje te są wykonywane podczas usuwania wtyczki z panelu administracyjnego, dzięki czemu po deinstalacji wtyczki baza danych jest od razu posprzątana.

Ważne jest, aby pamiętać o zaimplementowaniu funkcji deinstalacji, jeśli wtyczka zapisuje cokolwiek w bazie danych. Można też pozwolić użytkownikowi wybrać, czy chce usunąć dane, czy woli je pozostawić, aby móc ich użyć w przyszłości. Dobrym pomysłem jest też usuwanie danych przy wyłączaniu, ponieważ zaleca się zrobienie tego dla wszystkich wtyczek podczas ręcznego uaktualniania WordPressa.

## PO DEINSTALACJI

Łatwo jest zapomnieć, że po odinstalowaniu wtyczki w systemie mogą jeszcze być jakieś jej pozostałości. Dane w bazie danych to jedno (użytkownik powinien mieć przynajmniej możliwość wyboru, czy chce je usunąć), ale jest jeszcze jedna rzecz, która może być nawet bardziej kłopotliwa: tzw. skróty kodowe (ang. *shortcode*).

Co się dzieje, gdy zostanie odinstalowana wtyczka tworząca skróty kodowe? Skróty kodowe to specjalne ciągi znaków, które powodują wyświetlenie treści we wpisie w miejscu, w którym zostaną umieszczone. Jednym z najczęściej używanych w WordPressie skrótów tego typu jest

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

`[gallery]`. Można go zobaczyć, gdy doda się galerię do wpisu, a następnie przełączy się edytor w tryb HTML.

Co dzieje się ze skrótami wtyczki, gdy wtyczka ta zostanie odinstalowana? Nie zostaną przetworzone przez system jako skróty, tylko wyświetlone jak zwykły tekst. W tekście wówczas pojawią się napisy typu `[mojshortcode]`.

To nie będzie dobrze wyglądać.

Dlatego na wypadek gdyby wtyczka została wyłączona albo odinstalowana, powinniśmy udostępnić jakieś rozwiązanie kwestii skrótów kodowych. Rozwiązanie to musi umożliwiać użytkownikowi łatwe pozbycie się nieprzetwarzanych ciągów z tekstu wpisów. Jednym ze sposobów jest napisanie zapytania SQL po prostu usuwającego wszystkie wystąpienia danego skrótu, ale to jest dość drastyczne posunięcie i nie da się przewidzieć, co się stanie, gdy coś w trakcie procesu usuwania się nie powiedzie. W ten sposób można uszkodzić całą bazę danych. Poza tym nie wiadomo, czy z powodu błędu użytkownika wtyczka nie usunie za dużego fragmentu tekstu.

Sposób radzenia sobie z niepotrzebnymi skrótami kodowymi zależy od sposobu działania wtyczki. Bardziej szczegółowo kwestią tą zajmuję się w rozdziale 8.

Oczywiście nie z każdą wtyczką jest ten problem. Na przykład wtyczki tworzące tylko widżety nie są problematyczne, ponieważ po ich wyłączeniu widżety po prostu znikają.



# WTYCZKI TWORZĄCE WIDŻETY

Za pomocą widżetów można łatwo dostosować sposób wyświetlania treści w blogu lub witrynie. Widżety umieszcza się w specjalnie wyznaczonych do tego obszarach za pośrednictwem panelu administracyjnego. W WordPressie dostępnych jest kilka standardowych widżetów, np. wyświetlający kanały RSS, najnowsze wpisy, listę stron, listę kategorii itp. Funkcjonalność tych widżetów może być niewystarczająca dla użytkownika i dlatego tworząc wtyczkę, można dać użytkownikom możliwość skorzystania z niej także w formie widżetu. Jest to o wiele lepsze niż zmuszanie użytkownika do wpisywania tagów szablonowych w plikach PHP motywu. Jeżeli więc funkcjonalność Twojej wtyczki ku temu przemawia, warto umożliwić używanie jej jako widżetu.

Tworzenie widżetów dla wtyczek nie jest trudne, głównie dzięki API widżetów, którego szczegółowy opis można znaleźć na stronie *http://codex.wordpress.org/Widgets_API*. Polega to na rozszerzeniu wbudowanej klasy `WP_Widget`, przekazaniu kilku instrukcji i zarejestrowaniu widżetu, aby mógł być wyświetlany, na przykład:

```
class SmashingWidget extends WP_Widget {
    function SmashingWidget() {
        // Kod widżetu
    }
    function widget( $args, $instance ) {
        // Zwrócenie treści widżetu
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    }
    function update( $new_instance, $old_instance ) {
        // Przetworzenie i zapisanie opcji widżetu
    }
    function form( $instance ) {
        // Wyświetlenie formularza opcji w panelu administracyjnym
    }
}
register_widget( 'SmashingWidget' );
```

W tym przykładzie utworzyliśmy podklasę klasy `WP_Widget` o nazwie `SmashingWidget`. Pierwsza funkcja, `function SmashingWidget()`, zawiera rzeczywisty kod widżetu, a więc to ona jest odpowiedzialna za jego działanie. Funkcje `widget()`, `update()` i `form()` pozwalają sprawić, aby widżet zachowywał się tak, jak chcemy. Oczywiście widżet należy zarejestrować za pomocą funkcji `register_widget()`. Łącza *Anuluj* i *Zapisz* są wbudowane w API widżetów, a więc nie musimy implementować ich obsługi, aby użytkownik mógł zapisać lub anulować swoje ustawienia.

## TWORZENIE WIDŻETU

W tym podrozdziale przedstawiony jest krok po kroku proces tworzenia widżetu. Opisany tu widżet będzie wyświetlał tekst powitalny oraz będzie można zmienić jego tytuł w panelu administracyjnym:



1. Pamiętaj, że cały opisywany kod powinien znajdować się w pliku PHP wtyczki zawierającym nagłówek identyfikacyjny. Jeśli nie masz żadnej wtyczki, którą mógłbyś teraz rozbudować, utwórz nową.

   Zaczniemy od utworzenia klasy widżetu:

   ```
   class SmashingHello extends WP_Widget {
   ```

   Ten widżet będzie nazywał się `SmashingHello`, dzięki czemu od razu wiadomo, co prawdopodobnie będzie robił.

2. Następnie definiujemy funkcję widżetu:

   ```
   function SmashingHello() {
       parent::WP_Widget( false, $name = 'Witajcie, cześć i czołem' );
   }
   ```

3. Do wykonania wielu czynności potrzebne są też funkcje `widget()`, `update()` i `form()`. Zaczniemy od definicji funkcji `widget()`:

   ```
   function widget($args, $instance) {
       extract( $args );
       ?>
           <?php echo $before_widget; ?>
               <?php echo $before_title
                   . $instance['title']
                   . $after_title; ?>
               Cześć! Czy to nie jest wspaniałe?
   ```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
        <?php echo $after_widget; ?>
    <?php
}
```

W funkcji tej pobieramy argumenty. Zwróć uwagę na ustawienia `$before_widget`, `$after_widget`, `$before_title` oraz `$after_title`. Nie należy ich zmieniać, jeśli nie jest to konieczne. Są one kontrolowane przez API widżetów i domyślne funkcje motywów i dzięki nim widżety dobrze wyglądają.

Wartości zmiennych `$before_widget` i `$before_title` po prostu wysyłamy na wyjście, nie robiąc z nimi niczego szczególnego, a więc po prostu otrzymamy domyślny kod. Następna jest zmienna `$instance` reprezentująca tytuł widżetu, który użytkownik może wpisać w polu tekstowym dostępnym w panelu administracyjnym. Dalej znajduje się zmienna `$after_title`, a za nią tekst, który zostanie wyświetlony jako treść widżetu: *Cześć! Czy to nie jest wspaniałe?*. To jest oczywiście tylko przykład, więc nie ma tu żadnych fajerwerków, ale w tym miejscu możesz umieścić dowolny kod, choćby pętlę WordPressa. Na końcu widżet zamyka zmienna `$after_widget`.

Przypomnę, że zmienne ze słowem `before` i `after` w nazwie umożliwiają zmuszenie widżetu do zachowywania się zgodnie z konstrukcją motywu. Jest to kwestia zależna od projektanta motywu, a więc jeśli chcesz, aby Twój widżet był prawidłowo wyświetlany we wszystkich motywach, pozostaw ustawienia domyślne.

4. Następnie trzeba zadbać o to, aby widżet był poprawnie zapisywany w przypadku aktualizacji:

```
function update($new_instance, $old_instance) {
    return $new_instance;
}
```

Funkcja `update()` pobiera tylko argumenty `$new_instance` i `$old_instance`. Oczywiście zwraca `$new_instance`, ponieważ jest to reprezentacja zmian. Jeśli obawiasz się nieprzyjaznego kodu HTML, możesz zastosować filtrowanie znaczników przy użyciu funkcji `strip_tags()`. Użycie tej funkcji jest bardzo łatwe, poniżej znajduje się przykład dla pola wejściowego o nazwie `music`:

```
$instance['music'] = strip_tags( $new_instance['music'] );
```

W tym kodzie funkcja `strip_tags()` pilnuje, aby nie przedostał się żaden niepożądany kod HTML — jest to bardzo przydatne.

5. Teraz dodamy jeszcze jedno ustawienie pozwalające zmienić tytuł widżetu:

```
function form( $instance ) {
    $title = esc_attr( $instance['title'] );
    ?>
        <p>
            <label for="<?php echo $this->get_field_id( 'title' ); ?>">
                Tytuł: <input class="widefat" id="<?php echo $this->
                get_field_id( 'title' ); ?>" name="<?php echo $this->
                get_field_name( 'title' ); ?>" type="text" value="<?php
                  echo $title; ?>" />
            </label>
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            </p>
        <?php
    }
```

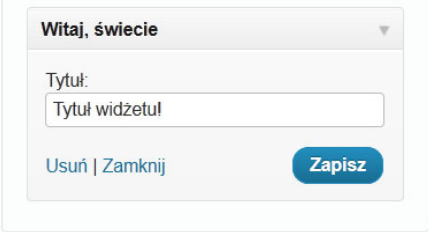
Kluczowe w tym przypadku są funkcje `get_field_name()` i `get_field_id()`. Pierwsza określa nazwę, a druga identyfikator elementu. W ten sposób został utworzony formularz ustawień widżetu, który można zapisać za pomocą przycisku *Zapisz* automatycznie tworzonego przez API widżetów.

6. Na koniec zamykamy klasę klamrą i rejestrujemy widżet:

```
}

function smashing_widget_init() {
    register_widget( 'SmashingHello' );
}

add_action( 'widgets_init', 'smashing_widget_init' );
```

Gotowy widżet jest przedstawiony na rysunku 7.5.






Rysunek 7.5. Utworzony widżet umieszczony na pasku bocznym w panelu administracyjnym

Masz już gotowy widżet, któremu możesz ustawiać tytuł i który wyświetla tekst. Oczywiście zamiast tekstu można wyświetlić cokolwiek, ponieważ wynik działania widżetu jest po prostu wynikiem działania skryptu PHP.

Należy też pamiętać, że nie wszystkie widżety muszą przyjmować opcje. Jeśli chcesz tylko umożliwić umieszczenie widżetu w obszarze widżetów, to nie twórz formularza ustawień. Nie ma sensu dodawać niepotrzebnych funkcji.

## WIDŻETY KOKPITU

Można tworzyć nie tylko zwykłe widżety, ale również widżety kokpitu, czyli takie, które umieszcza się w panelu administracyjnym WordPressa potocznie nazywanym **kokpitem**. Widżetami są wszystkie ramki, które widać po wejściu do panelu administracyjnego; można też tworzyć własne takie ramki.

Aby utworzyć widżet kokpitu, należy utworzyć wtyczkę, a więc też i nowy plik. Poniżej znajduje się krótkie przypomnienie dla użytkowników grupowego bloga, żeby weszli na wewnętrzną stronę. Składa się ono tylko z tekstu i odnośników. Najpierw należy utworzyć odpowiednią funkcję:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
function dashboard_reminder() {
    echo '
        Hej! Nie zapomnijcie przeczytać ważnych informacji na wewnętrznych
        stronach:<br />
        &larr; <a href="http://domain.com/internal/forum">Forum</a><br />
        &larr; <a href="http://domain.com/internal/docs">Dokumentacja</a><br />
        &larr; <a href="http://domain.com/internal/staff">Obsada</a><br />
        DZIĘKI!
     ';
}
```

Ta prosta funkcja o nazwie `dashboard_reminder()` wysyła na wyjście kod HTML, który stanowi treść widżetu. Następnym krokiem jest dodanie samego widżetu:

```
function dashboard_reminder_setup() {
    wp_add_dashboard_widget( 'dashboard_reminder_widget', 'Przypomnienie',
         'dashboard_reminder' );
}
```

Najważniejsza w tym kodzie jest funkcja `wp_add_dashboard_widget()`, której przekazaliśmy identyfikator widżetu (`dashboard_reminder_widget`), etykietę tekstową widżetu oraz nazwę funkcji zawierającej treść widżetu (`dashboard_reminder()`). Warto też wiedzieć, że identyfikator widżetu będący pierwszym parametrem funkcji `wp_add_dashboard_widget()` zostanie również użyty jako klasa tego widżetu, za pomocą której można go dowolnie sformatować przy użyciu CSS.



Zatrzymamy się na moment przy funkcji `wp_add_dashboard_widget()`. Ma ona jeszcze jeden parametr, o nazwie `$control_callback`, który jest opcjonalny i domyślnie przyjmuje wartość `null`. Nie został użyty w tym przykładzie, ale warto wiedzieć o istnieniu jeszcze jednego parametru, który również może być przydatny.

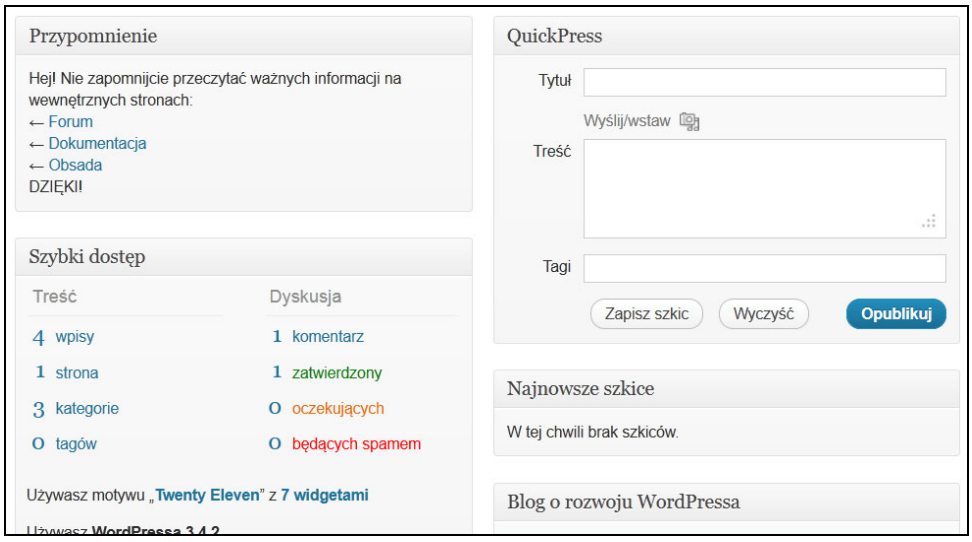
Wracając do przykładu, musimy jeszcze dodać akcję widżetu do haka `wp_dashboard_setup` za pomocą funkcji `add_action()`:

```
add_action( 'wp_dashboard_setup', 'dashboard_reminder_setup' );
```

To wszystko, widżet kokpitu jest już gotowy (rysunek 7.6)! Na razie nie istnieje żadne API pozwalające określać kolejność widżetów w panelu administracyjnym, a więc nasz widżet zostanie wyświetlony na dole. Użytkownik może go przesunąć w dowolne miejsce, ale są też sposoby na wyniesienie widżetu na górę automatycznie. Jeśli Cię to interesuje, zajrzyj np. do dokumentacji — na stronę *http://codex.wordpress.org/Dashboard_Widgets_API*.

Wiesz już, jak tworzyć wtyczki i widżety, a więc czas zająć się kwestią bazy danych. Zapisać informacje w bazie danych jest bardzo łatwo, ale nie zawsze jest to najlepsze rozwiązanie.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 7.6. Widżet kokpitu w całej okazałości



# KWESTIA KORZYSTANIA Z BAZY DANYCH WE WTYCZKACH

Czasami aby wtyczka działała prawidłowo, konieczne jest umożliwienie jej zapisywania informacji w bazie danych. Baza danych to bardzo przydatny magazyn, w którym można przechowywać wszystko — od prostych ustawień konfiguracyjnych po całe tabele danych do użytku użytkowników. Dane umieszczone w bazie danych bardzo łatwo jest pobrać i jest to bardzo wygodne rozwiązanie.

Niestety ta wygoda ma swoją cenę. Brak porządku w bazie utrudnia jej utrzymanie i dlatego trzeba pamiętać o tym, aby użytkownik mógł wraz z wtyczką usuwać dane tej wtyczki z bazy danych. Ponadto trzeba zdecydować, gdzie należy zapisywać dane. Można do tego użyć API ustawień, dzięki czemu mamy pewność, że dane trafią tam gdzie trzeba — jest to dobre rozwiązanie w przypadku niewielkich ilości danych konfiguracyjnych — albo utworzyć własną tabelę przeznaczoną tylko dla naszej wtyczki. Drugie rozwiązanie jest zwykle lepsze w przypadkach, gdy trzeba zapisywać duże ilości danych. Rozszerzanie bazy danych może powodować problemy z innymi wtyczkami również korzystającymi z tej bazy. Ponadto taka dodatkowa tabela nie należy do WordPressa i trzeba oddzielnie tworzyć jej kopię zapasową, a ponadto jeśli zechcesz przenieść instalację w inne miejsce, nie zostanie ona standardowo uwzględniona przez narzędzia eksportu i importu.

Pozostaje też kwestia buforowania, którą warto rozważyć w przypadku niektórych wtyczek, zwłaszcza regularnie pobierających dane z zewnętrznego źródła. Oczywiście do buforowania można używać plików, ale można też wykorzystać API Transients, które zostało właśnie do tego stworzone. Opis tego API znajduje się na stronie *http://codex.wordpress.org/Transients_API*.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Które rozwiązanie jest najlepsze? Zapisywać dane w opcjach, czy we własnej tabeli? Może należałoby użyć tabeli metadanych albo wpisów? Wybór zależy od tego, co chcesz przechowywać. Na podstawie własnego doświadczenia mogę powiedzieć, że ustawienia najlepiej zapisywać w tabeli opcji; realną treść i większe ilości danych często umieszczam w osobnej tabeli.

Pamiętaj tylko, aby poinformować użytkownika o sytuacji, i dodaj możliwość zrobienia porządku w bazie danych po odinstalowaniu wtyczki.

# ZGODNOŚĆ WSTECZNA WTYCZEK

Kolejnym aspektem, jaki należy rozważyć podczas budowy wtyczki, jest to, czy wtyczka ta ma być zgodna ze starszymi wersjami systemu. Do WordPressa wciąż są dodawane nowe funkcje i haki i jeśli będziemy używać tych najnowszych, to naturalnie w starszych wersjach systemu nasza wtyczka nie będzie działać. Dlatego trzeba ostrożnie korzystać z najnowszych dodatków do systemu, ponieważ mogą one nie działać w jego starszych wersjach.

Co więcej, nie tylko funkcjonalność WordPressa się zmienia. Także wymagania systemowe co jakiś czas podlegają modyfikacjom. W tej chwili do działania systemu wymagany jest PHP 5.2.4. Wtyczka powinna obsługiwać te same wersje oprogramowania co sam WordPress, ale jeśli chcesz użyć nowszych wersji PHP lub MySQL, koniecznie wyświetl informację o błędzie, gdy ktoś spróbuje uruchomić ją w systemie ze starszymi wersjami PHP lub MySQL.



Jak daleko chcesz sięgać w przeszłość w swojej wtyczce, zależy tylko od Ciebie. Od kiedy wprowadzono możliwość automatycznej aktualizacji systemu, częstotliwość aktualizowania wzrosła. Jednak wciąż jest zaskakująco dużo witryn działających w oparciu o stare wersje WordPressa, co jest niekorzystne zarówno dla programistów, jak i właścicieli witryn. W każdym razie przestarzałe oprogramowanie jest mniej bezpieczne i powoduje problemy ze zgodnością z nowszymi modułami.

# PUBLIKOWANIE WTYCZEK W PORTALU WORDPRESS.ORG

Dla wtyczek, podobnie jak dla motywów, istnieje oficjalny serwis w portalu WordPress.org, w którym można je publikować. Oczywiście nie musisz tego robić, ale dzięki temu użytkownicy otrzymują automatycznie powiadomienia o dostępności aktualizacji i mogą ich dokonywać na bieżąco.

Aby wtyczka została opublikowana w portalu WordPress.org, musi spełniać pewne wymagania:

- musi być na licencji zgodnej z GPL;
- nie może wykonywać nielegalnych ani „niemoralnych” działań;

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- musi zostać wysłana do repozytorium Subversion WordPress.org;
- musi zawierać poprawny plik *readme.txt*.

Aby móc opublikować wtyczkę, trzeba być użytkownikiem portalu WordPress.org. Następnie wysyła się wtyczkę (*http://wordpress.org/extend/plugins/add*) i czeka na jej zatwierdzenie. Czas oczekiwania zależy od tego, ile aktualnie pracy ma zespół zajmujący się weryfikowaniem wtyczek.

Gdy wtyczka zostanie zatwierdzona, otrzymasz dostęp do katalogu Subversion, do którego możesz wysłać wtyczkę z plikiem *readme.txt*. Poprawność pliku *readme.txt* sprawdza specjalne narzędzie (*http://wordpress.org/extend/plugins/about/validator*). Weryfikuje ono, czy w pliku znajdują się wszystkie informacje potrzebne do publikacji wtyczki.

Przed wysłaniem wtyczki należy przeczytać odpowiedzi na najczęściej zadawane pytania dotyczące wtyczek opublikowane pod adresem *http://wordpress.org/extend/plugins/about/faq*. To na pewno usprawni proces zatwierdzania wtyczki.

Korzyścią z opublikowania wtyczki w zbiorach WordPress.org są nie tylko automatyczne aktualizacje, ale również prowadzenie statystyk. Dowiesz się, ile osób pobrało Twoją wtyczkę, oraz otrzymasz oceny i komentarze od użytkowników. Ponadto witryna WordPress.org jest centrum społeczności WordPressa, a więc stwarza największą szansę, że użytkownicy w ogóle znajdą wtyczkę. Jeśli opublikujesz ją na własnym serwerze, to z jej znalezieniem może być różnie, tymczasem dzięki narzędziu wyszukiwania wtyczek w panelu administracyjnym Twoją wtyczkę będzie mógł znaleźć i zainstalować dosłownie każdy użytkownik WordPressa.



Ale warunkiem jest obecność w zbiorach WordPress.org. Dlatego postaraj się tam zaistnieć ze swoją wtyczką!

---

*Szukasz innego miejsca do opublikowania swojej wtyczki niż WordPress.org? Sprawdź GitHub (http://github.com). Jest to serwis do publikowania programów z otwartym kodem źródłowym. Bardzo wygodnie się z niego korzysta, zwłaszcza jeśli zna się obsługę programu Git. Jeśli go nie znasz, zawsze możesz poznać.*

---

# SŁOWO OSTRZEŻENIA NA TEMAT TWORZENIA WTYCZEK

Wtyczka to nie to samo co motyw. Oczywiście można znaleźć jakieś podobieństwa w procesie powstawania, ale jednak w przypadku wtyczek tak naprawdę pisze się zwykłe skrypty PHP podłączone do WordPressa. W związku z tym podczas gdy każdy posiadający podstawową wiedzę na temat pisania skryptów może nagiąć WordPressa do swoich potrzeb w motywie, z pisaniem kodu wtyczek nie będzie już miał tak łatwo. Do tego potrzebna jest praktyczna umiejętność pisania skryptów PHP i trzeba przy tym bardzo uważać, ponieważ łatwo można coś zepsuć, zwłaszcza jeśli szpera się w bazie danych.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Biorąc pod uwagę te kwestie i znając się na programowaniu w PHP, za pomocą wtyczek możesz sprawić, że Twoja witryna będzie działała dokładnie tak, jak sobie zaplanujesz.

W następnym rozdziale dowiesz się, kiedy używać wtyczek, a kiedy lepiej jest pozostać przy pliku *functions.php*.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 8 WTYCZKA CZY PLIK FUNCTIONS.PHP?

**MOTYWY I WTYCZKI WORDPRESSA** działają niezależnie od siebie, a jedynym punktem ich styku jest implementacja funkcji. Ta współpraca zwykle wyraża się w przygotowaniu przez programistę obszarów na widżety w plikach szablonowych motywu, w których użytkownik może umieszczać widżety tworzone przez wtyczki. Można też umieszczać kod PHP wtyczek bezpośrednio w plikach motywów. Niektóre wtyczki zwracają lub włączają funkcje przy użyciu haków WordPressa. W takich przypadkach wystarczy, aby motyw był zgodny z WordPressem, by można było w nim używać tych wtyczek. To wszystko jest bardzo proste.

Jest jedno miejsce, w którym motywy i wtyczki mogą wchodzić sobie wzajemnie w drogę. Jest to plik *functions.php*. Plik motywu może robić mniej więcej to samo co wtyczka, co oznacza, że użycie takiego pliku może być alternatywą dla wtyczki. Oznacza to również, że jeśli plik *functions.php* nie będzie używany z rozwagą, może wchodzić w konflikty z wtyczkami.

W tym rozdziale znajdziesz wyjaśnienie, kiedy należy tworzyć wtyczki, a kiedy używać pliku *functions.php*.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# KIEDY TWORZYĆ WTYCZKI

Gdy podejmujesz decyzję, czy utworzyć wtyczkę, czy skorzystać z pliku *functions.php*, albo gdy porównujesz wady i zalety wtyczek oraz własnych haków w motywie, rozważ następujące kwestie:

- Czy wtyczka jest na pewno potrzebna?
- Czy to, co ma robić wtyczka, jest na tyle skomplikowane lub niezależne od systemu, że nie ma sensu zastanawiać się nad zaimplementowaniem tego w plikach motywu?
- Czy wtyczka musi działać poza wybranym motywem?

Jeśli odpowiedziałeś twierdząco na wszystkie powyższe pytania, to znaczy, że powinieneś utworzyć wtyczkę.

## ROZSZERZANIE FUNKCJONALNOŚCI ZA POMOCĄ WTYCZEK

Jest prosta zasada określająca, kiedy należy wybrać wariant z utworzeniem wtyczki: jako wtyczkę należy implementować wszystko, co rozszerza funkcjonalność WordPressa. To znaczy, że jeśli dodasz do systemu coś, czego nie dałoby się zrobić w żaden inny sposób, dodatek ten powinien być zrealizowany jako wtyczka. Wśród przykładów takich rozszerzeń można wymienić wszystko od narzędzi do zarządzania danymi po integrację systemu z zewnętrznymi aplikacjami.



Jest tu też jednak spora szara strefa. Wiele wtyczek naśladuje działanie WordPressa i jego pętlę, dzięki czemu użytkownicy nie muszą pisać własnych pętli w plikach szablonowych motywu. Te zadania można zlecić wtyczce i jest to rozwiązanie wygodne dla użytkownika, który może zdefiniować różne ustawienia pętli na stronie opcji i użyć wygodnego widżetu, przeciągając go w dowolne miejsce. Porównaj to z koniecznością pisania kodu pętli bezpośrednio w pliku motywu w miejscu, w którym ma zostać wyświetlony efekt jej działania. Dla programisty PHP może to nie jest żaden wyczyn, ale dla większości użytkowników WordPressa stanowi to poważne utrudnienie. Wtyczki pomagające w takich czynnościach nie rozszerzają funkcjonalności WordPressa, ale umożliwiają użytkownikom robienie rzeczy, których w inny sposób nie potrafiliby osiągnąć.

Kolejnym przykładem wbudowanej funkcjonalności, która mogłaby być lepiej zrealizowana przy użyciu wtyczki, są własne pola. Pola te doskonale nadają się do wielu zadań, ale są niezbyt łatwe w użyciu i mają niezgrabny interfejs. Ich funkcję może przejąć wtyczka o bardziej przyjaznym dla użytkownika interfejsie. To może sprawić, że aktualizowanie witryny będzie łatwiejsze.

## OSTRZEŻENIE: WTYCZKI MOGĄ SPOWOLNIĆ TWOJĄ WITRYNĘ

Nikt nie lubi przeglądać powoli ładujących się stron, choćby nie wiadomo jak były atrakcyjne. Wymagania użytkowników w kwestii szybkości otwierania stron są uzależnione od charakteru samych użytkowników, jak również od typu prezentowanej treści. Ciągle jeszcze można spotkać strony z wprowadzającymi animacjami, dźwiękami i nie wiadomo czym jeszcze.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jednak takie rzeczy powinny znajdować się wyłącznie na stronach, których grupa docelowa naprawdę to lubi.

Żadna witryna nie powinna być wolniejsza, niż jest to konieczne, a jednym z najlepszych sposobów na rozdęcie WordPressa jest napakowanie go po brzegi najrozmaitszymi wtyczkami. Pomyśl: większość wtyczek musi zostać zainicjowana, gdy użytkownik otworzy stronę przedstawiającą efekt ich działania. Inicjacja może polegać na wykonaniu prostej funkcji PHP w najlepszym przypadku lub na wykonaniu pełnej pętli wymagającej połączenia z bazą danych w najgorszym. Powiedzmy, że mamy na stronie 10 wtyczek, z których każda inicjuje pętlę oraz wyświetla jakąś treść z bazy danych. To nie tylko wymaga czasu, ale również obciąża serwer, przez co działa on o wiele wolniej.

Jak wspomniałem wcześniej, wiele wtyczek to zwykłe pętle w przebraniu. Jeśli wyświetlasz wpisy, komentarze, tagi lub cokolwiek innego, co jest z nimi związane, to pobierasz dane z bazy danych. Jest to to samo co umieszczenie kilku własnych pętli w motywie, aby wyświetlały różne rodzaje treści.

Przyzwoity serwer da radę obsłużyć te dodatkowe zapytania do bazy danych, ale jeśli będzie wielu użytkowników w tym samym czasie, to każdy serwer może się „wyłożyć". Zapytania do bazy danych powodują duże obciążenie i dlatego należy ograniczać ich ilość na jednej stronie, jeśli nie ma się odpowiedniego zaplecza sprzętowego.



To samo dotyczy innych modułów, od prostych funkcji PHP po dołączaną treść zewnętrzną. Oczywiście największe problemy sprawia zawsze treść zewnętrzna, ponieważ w jej przypadku musimy martwić się nie tylko własnym sprzętem, ale jeszcze dodatkowo szybkością łączy internetowych oraz sprzętem hosta dostarczającego tę treść. Z tego powodu właśnie wtyczki wyświetlające informacje z różnych serwisów internetowych mogą znacznie spowolnić Twoją witrynę.

Dlatego nie należy używać zbyt wielu wtyczek obciążających serwer. Im ich mniej, tym szybciej będą się wczytywały nasze strony. Jeśli jednak musisz użyć dużej liczby „ciężkich" wtyczek, rozważ jednoczesne użycie jakiegoś rozwiązania buforującego. Co ciekawe, najlepszymi rozwiązaniami tego typu są właśnie wtyczki, np. W3 Total Cache (*http://wordpress.org/extend/plugins/w3-total-cache*). Dzięki tej wtyczce użytkownikom zamiast dynamicznej treści w większości przypadków wysyłane są statyczne strony internetowe. Więcej szczegółów na temat tych rozwiązań znajduje się w dodatku A. Warto też pamiętać, że mimo iż duża liczba wtyczek może spowodować pogorszenie wydajności witryny, nie zawsze tak jest. Można mieć dużo wtyczek, a nie ponieść szkód na wydajności. Wszystko zależy od sposobu działania samych wtyczek.

# KIEDY UŻYWAĆ PLIKU FUNCTIONS.PHP

Kiedy tak naprawdę powinno się używać pliku *functions.php*? Ja kieruję się prostą regułą: pliku *functions.php* używam tylko wtedy, gdy funkcja, którą chcę dodać, dotyczy tylko jednego konkretnego motywu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Uzasadnienie tej reguły jest proste: plik *functions.php* jest związany z motywem, co oznacza, że jeśli zmienimy motyw, to funkcje zdefiniowane w tym pliku staną się niedostępne, przez co trzeba je będzie przepisać albo porzucić. To dyskwalifikuje ten plik dla wszelkich funkcji związanych z kontrolowaniem wyświetlania danych, ponieważ po zmianie motywu dane przestałyby być wyświetlane we właściwy sposób albo w najgorszym razie na stronie pojawiłyby się różne komunikaty o błędach.

Najczęściej wprowadza się tylko drobne zmiany w funkcjonalności witryny i w wielu przypadkach lepszym rozwiązaniem jest zaimplementowanie ich w plikach szablonowych niż tworzenie kolejnej wtyczki. Zdarza się jednak, że modyfikacje są bardzo duże — wówczas powinno się je dodać jako wtyczkę, bo mogą być potrzebne w wielu sytuacjach. I nawet jeśli w pewnym momencie wyda Ci się, że Twój motyw jest idealny, to najpewniej za jakiś czas i tak przełączysz się na inny, idealny motyw, przez co będziesz musiał rozwiązać problem z utratą wszystkiego, co jest w pliku *functions.php*. Oczywiście zawsze możesz przenieść kod do nowego pliku, ale po co, jeśli nie musisz?

Moim zdaniem reguła umieszczania w pliku *functions.php* tylko tego, co dotyczy bezpośrednio konkretnego motywu, jest logiczna. Do tego typu funkcji można zaliczyć ustawienia dotyczące układu (które w innym motywie są nieważne) oraz strony opcji motywu, które również mogą w każdym motywie wyglądać trochę inaczej. Ponieważ wtyczka realizująca tego typu funkcje działałaby poprawnie tylko w jednym motywie, lepiej jest je umieścić w pliku *functions.php*.



Podsumowując, wtyczki służą do rozszerzania funkcjonalności WordPressa, a plik *functions.php* służy do modyfikowania motywu.

# DYLEMAT ZE SKRÓTAMI KODOWYMI

**Skróty kodowe** (ang. *shortcode*), o których była mowa w poprzednim rozdziale, czasami mogą się wydawać doskonałym rozwiązaniem i rzeczywiście często nim są. Niewielu użytkowników WordPressa jest skłonnych do posługiwania się językiem HTML i dla nich dobrym rozwiązaniem jest skrót kodowy. Problem może się pojawić, gdy kod obsługi tych skrótów będzie zapisany w pliku *functions.php*. Przypuśćmy, że masz w swoim motywie skrót kodowy do tworzenia atrakcyjnych wizualnie blokowych cytatów o nazwie `[cytat]`. Funkcja ta była standardowo dostępna w motywie i od kiedy zacząłeś go używać, w wielu wpisach umieściłeś sporo takich cytatów. Ale czas leci, Twoje potrzeby się zmieniają i postanowiłeś zmienić motyw. Włączasz go i okazuje się, że Twoje piękne cytaty gdzieś zniknęły! Zamiast nich wyświetlony jest tylko tekst `[cytat]`. Jak to możliwe?

To proste. Kod obsługujący skrót kodowy `[cytat]` znajdował się w pliku *functions.php* starego motywu, który został wyłączony. W związku z tym skrót `[cytat]` jest traktowany jak zwykły tekst, co nie jest dla nas miłym zaskoczeniem.

Na szczęście rozwiązanie tego problemu jest proste. Wystarczy skopiować kod obsługi cytatów z pliku *functions.php* ze starego motywu i wkleić go do pliku *functions.php* w nowym motywie. Jeszcze lepszym rozwiązaniem może być użycie pliku *functions.php* motywu potomnego,

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

do czego za chwilę dojdziemy. Można też utworzyć specjalną wtyczkę do przechowywania różnych przydatnych funkcji, której można by było używać w różnych motywach. W takiej wtyczce można by było zaimplementować wszystko, co się chce. Więcej na temat tego rodzaju wtyczek napisałem nieco dalej.

# ROZWIĄZANIE PROBLEMU POPRZEZ UŻYCIE MOTYWU POTOMNEGO

Innym sposobem na rozwiązanie problemu ze znikającymi funkcjami jest użycie motywu potomnego. Jeśli utworzyłeś lub znalazłeś idealny motyw do rozbudowy, możesz na jego bazie tworzyć różne motywy potomne dla kolejnych wersji swojej witryny. Jak zawsze w przypadku motywów potomnych podstawowe funkcje — jak wszystko to, co zapisujesz w pliku *functions.php* — trzymaj w motywie nadrzędnym, a zmiany projektu i drobne modyfikacje wykonuj w motywie potomnym.

Dzięki temu wystarczy, że utworzysz nowy motyw potomny, gdy zechcesz uruchomić nową wersję witryny z nowym stylem. Nie trzeba będzie niczego kopiować z pliku *functions.php*, ponieważ wszystko co potrzebne będzie znajdować się w pliku o tej nazwie w motywie nadrzędnym.

Więcej informacji na temat motywów potomnych znajduje się w rozdziale 5.



# ROZWIĄZANIE UNIWERSALNE: WTYCZKA NA FUNKCJE

Wtyczka na funkcje łączy w sobie zalety wszystkich opisywanych rozwiązań. We wtyczce takiej zapisuje się wszystkie funkcje, które chcemy mieć dostępne niezależnie od motywu. Wśród tych funkcji mogą być obsługa skrótów kodowych, a także deklaracje własnych taksonomii i typów wpisów, ponieważ żadne z nich nie jest przywiązane do jednego motywu. Przecież nie chciałbyś, aby po zmianie motywu utworzone przez Ciebie taksonomie nagle zniknęły, prawda?

We wtyczce na funkcje nie powinno się m.in. zapisywać pasków bocznych i menu, ponieważ budowa tych elementów zależy od budowy i projektu motywu. To samo dotyczy ikon wpisów, obrazów nagłówkowych, obsługi funkcji ustawiania tła przez użytkownika itd. Te wszystkie rzeczy są zależne od konkretnego motywu.

## TWORZENIE WTYCZKI NA FUNKCJE

Wtyczkę na funkcje tworzy się bardzo łatwo. Dzięki temu, że kod znajdujący się w pliku *functions.php* jest w zasadzie gotowy do użycia we wtyczce, wystarczy tylko utworzyć plik wtyczki i wkleić do niego odpowiednie fragmenty kodu. Poniżej znajduje się kod nagłówka

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

przykładowej wtyczki, którą możesz zapisać w pliku o nazwie *smashing-functionality.php*. W pliku tym możesz zapisać wszystkie, jakie tylko chcesz funkcje:

```
<?php
/*
Plugin Name: Smashing Functionality, Baby!
Description: Dzięki tej wtyczce moje funkcje są przenośne.
Version: 1.0
Author: Thord Daniel Hedengren
Author URI: http://tdh.me
*/
?>
```

Kod ten to prosty nagłówek wtyczki. Opuściłem *URI*, ponieważ nie ma sensu go podawać dla wtyczki, której jedynym zadaniem jest przechowywanie w wygodny sposób naszych własnych funkcji. Można by było jeszcze dodać informację o licencji, ale pominąłem ją dla uproszczenia kodu.

Teraz dodamy do naszej wtyczki jakąś funkcję z pliku *functions.php*. Powiedzmy, że w pliku tym znajduje się kod obsługujący skrót kodowy `[cytat]`, który chcielibyśmy móc przenosić między motywami. Kod ten po prostu wpisz pod deklaracją wtyczki:



```
<?php
/*
Plugin Name: Smashing Functionality, Baby!
Description: Dzięki tej wtyczce moje funkcje są przenośne.
Version: 1.0
Author: Thord Daniel Hedengren
Author URI: http://tdh.me
*/

// Skrót kodowy dla cytatów blokowych
function smashing_pullquote( $atts, $content = null ) {
    extract( shortcode_atts( array(
         'float' => '$align',
    ), $atts ) );
if ($content) {
    return '<blockquote class="pullquote ' . $float . '">' . $content .
         '</blockquote>';
    }
}
add_shortcode( 'cytat', 'smashing_pullquote' );
?>
```

Wystarczy po prostu przekleić kod z pliku *functions.php* w niezmienionej postaci. Oczywiście kod z pliku *functions.php* należy już usunąć. Jeśli tego nie zrobisz, powstaną konflikty funkcji. W tym przypadku dobrym pomysłem może być też dodanie arkusza stylów dla klasy CSS `pullquote`, ale nie zrobiłem tego, ponieważ w każdym motywie taki blokowy cytat może wyglądać inaczej. To, czy arkusz stylów dodasz we wtyczce (za pomocą funkcji `wp_enqueue_style()`), czy w pliku *style.css* motywu, zależy od Ciebie. Ja w przykładzie skupiłem się tylko na samej funkcjonalności.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W takiej wtyczce można zdefiniować dowolną funkcję, która dzięki temu stanie się przenośna między różnymi motywami, ponieważ aby jej użyć, wystarczy tylko włączyć tę wtyczkę. Oczywiście najpierw trzeba sprawdzić, czy dodawane funkcje mogą zostać obsłużone przez dany motyw. Jeśli np. we wtyczce umieścisz kod tworzący własne taksonomie, to zostaną one tylko włączone i nic więcej. Aby ich rzeczywiście użyć we wpisach w witrynie, trzeba jeszcze dodać ich obsługę.

Wtyczki na funkcje dobrze sprawdzają się także w innych zastosowaniach, nie tylko jako magazyny funkcji z pliku *functions.php*. Powiedzmy, że nie planujesz używać żadnej z wtyczek statystyk Google Analytics, ale nie chcesz też ryzykować, że zapomnisz wstawić kod śledzący do plików szablonowych. Nie ma problemu — wystarczy, że dodasz odpowiedni kod za pomocą haków `wp_head` i `wp_footer` poprzez wtyczkę na funkcje. W ten sposób kod Google Analytics będziesz mieć zawsze poza plikami motywu, a jednocześnie nie zapomnisz go wstawić we właściwe miejsce i nie stracisz cennych danych na temat ruchu w Twojej witrynie, gdy zmienisz motyw. Oczywiście możesz nawet utworzyć osobną taką wtyczkę. Wtyczek na funkcje możesz przecież utworzyć dowolną liczbę.

# JAK WAŻNA JEST PRZENOŚNOŚĆ



Już kilka razy w tym rozdziale poruszałem temat przenośności, a ponieważ jest on bardzo ważny, poświęcę mu jeszcze trochę miejsca. Wcześniej napisałem, że jeśli chce się mieć możliwość używania wybranych funkcji także po zmianie motywu, funkcje te można zapisać w specjalnej wtyczce na funkcje, zamiast umieszczać je w przerośniętym pliku *functions.php*.

Jeśli którykolwiek z poniższych składników ma działać niezależnie od używanego motywu, powinien on być zaimplementowany w przenośny sposób, tzn. znajdować się we wtyczce na funkcje zamiast w pliku *functions.php*:

- własne typy wpisów,
- własne taksonomie,
- funkcje obsługi skrótów kodowych,
- modyfikacje panelu administracyjnego.

Nie wszystkie te rzeczy zawsze powinny znajdować się we wtyczce, ale na ogół wtyczka powinna je zawierać. Zwłaszcza modyfikacje panelu administracyjnego — takie jak ukrycie wybranych modułów i funkcji przed użytkownikiem — są bezpośrednio związane z konkretnym motywem i mogą stać się bezużyteczne po jego zmianie na inny.

Własne typy wpisów i taksonomie zazwyczaj mają charakter globalny, tzn. są równie ważne w każdym motywie. Przecież nawet po zmianie motywu nadal będziesz chciał używać swoich własnych typów wpisów i taksonomii, prawda?

Dbaj o to, aby funkcje, które nie są związane z motywem, bezproblemowo dały się przenosić między motywami. Aby tak było, należy używać wtyczek, a nie pliku *functions.php*, dzięki czemu uda się uniknąć sporo pracy typu kopiuj i wklej.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PLANOWANIE ROZSZERZANIA FUNKCJONALNOŚCI WORDPRESSA

Gdy poczujesz potrzebę rozszerzenia funkcjonalności WordPressa w jakimś projekcie, musisz zastanowić się, czy cel lepiej będzie zrealizować bezpośrednio w plikach motywu, czy jednak przy użyciu wtyczki. Często gdy zmienia się typowy układ treści, stosuje się rozwiązanie mieszane. Ważne jest tylko, aby decyzje były podejmowane na podstawie solidnych argumentów. Oto kilka oczywistych pytań, na które należy sobie odpowiedzieć:

- Czy potrzebuję wtyczki dla funkcji X?
- Czy da się rozszerzyć funkcję X, nie zastępując jej całkiem nową funkcją?
- Czy funkcja X będzie działać, gdy zmieni się motyw?
- Czy funkcja X może spowodować problemy, gdy nastąpi aktualizacja WordPressa?
- Jak funkcja X będzie się integrować z motywem oraz wtyczkami i innymi funkcjami?

Ogólnie chodzi o to, aby przed rozpoczęciem budowy witryny opracować dokładny plan działania. Dotyczy to zarówno tworzenia małych blogów, publikowania czasopism internetowych, jak również całkiem innych projektów opartych na WordPressie. Oczywiście im bardziej ekstremalne rozwiązania planujesz i im mniej Twoja witryna ma przypominać zwykłego bloga, tym więcej czasu należy przeznaczyć na planowanie. Przykłady różnych zastosowań WordPressa znajdziesz w następnej części tej książki.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# IV

# DODATKOWE FUNKCJE I ROZSZERZANIE FUNKCJONALNOŚCI WORDPRESSA

Rozdział 9. Używanie WordPressa jako systemu CMS

Rozdział 10. Integracja WordPressa z mediami społecznościowymi

Rozdział 11. Sztuczki projektowe

Rozdział 12. Zabawa z mediami

Rozdział 13. Dodatkowe funkcje

Rozdział 14. Nietypowe zastosowania WordPressa

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 9

# UŻYWANIE WORDPRESSA JAKO SYSTEMU CMS

**AKRONIM CMS POCHODZI** od angielskich słów *content management system* oznaczających **system zarządzania treścią**. CMS-ów używa się do pisania, edytowania i publikowania treści w internecie, najczęściej w bazie danych lub zwykłych plikach. Jest to znacznie wygodniejszy sposób niż edytowanie plików lub bazy danych w sposób bezpośredni, chociaż kiedyś tak właśnie się robiło.

Mimo iż WordPress powstał jako system do blogowania, z czasem ewoluował do postaci wszechstronnego i potężnego CMS-a. W tym rozdziale opisane są problemy i kwestie, jakie trzeba rozwiązać, gdy planuje się użyć WordPressa jako tradycyjnego systemu CMS. Wykorzystanie go w ten sposób jest nie tylko możliwe, lecz również jest doskonałym wyborem, dzięki któremu możesz zaoszczędzić sporo czasu i pieniędzy.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# CZY WORDPRESS JAKO CMS TO DOBRY WYBÓR

Po lekturze poprzednich rozdziałów wiesz już, że WordPress reprezentuje znacznie więcej niż prostą platformę do tworzenia blogów (jeśli nadal to do Ciebie nie dociera, zmienisz zdanie po przeczytaniu rozdziału 14.). Przy użyciu tego systemu z tekstem, obrazami i filmami można zrobić w zasadzie wszystko. Często mówię właścicielom gazet i różnych innych czasopism, że nie ma takiej witryny czasopisma, której nie umiałbym przenieść na WordPressa. Witryny te zazwyczaj są tworzone w oparciu o drogie licencjonowane systemy i podczas gdy jeszcze kilka lat temu mało kto wziąłby moje deklaracje na poważnie, dziś nikogo to już tak bardzo nie dziwi. Już teraz duże grupy wydawnicze wykorzystują WordPressa do budowy swoich serwisów i obsługi sekcji zbierania opinii o publikacjach.

Wybór CMS-a jest bardzo ważny, ponieważ należy wybrać taki system, który jest najlepszy w robieniu tego, co nas interesuje. WordPress jest najlepszym wyborem, gdy potrzebujemy CMS-a, który:

- jest darmowy i ma otwarty kod źródłowy,
- jest szybki i łatwy w użyciu,
- jest łatwy do rozbudowania,
- udostępnia łatwe w użyciu API do tworzenia wtyczek,
- doskonale radzi sobie z tekstem,
- jest przyjazny dla wyszukiwarek,
- dobrze radzi sobie z wyświetlaniem obrazów graficznych.



Krótko mówiąc, jeśli w swojej witrynie planujesz prezentować przede wszystkim treści tekstowe, to bez obaw możesz użyć WordPressa. System ten równie dobrze nadaje się do budowy witryn, w których główną treść stanowią obrazy, a także filmy wideo.

Kiedy zatem WordPress może być nie najlepszym rozwiązaniem? Oto kilka argumentów przemawiających za tym, aby użyć innego systemu CMS w niektórych sytuacjach:

- Funkcje społecznościowe — kiedyś WordPress przegrywał z innymi CMS-ami w kwestii funkcji społecznościowych, ale od czasu pojawienia się dodatku BuddyPress (*http://buddypress.org*) i uproszczenia integracji z forum bbPress (*http://bbpress.org*) sytuacja uległa radykalnej zmianie.
- Modularność — niektóre CMS-y (np. Drupal) mają bardziej modularną budowę, dzięki czemu można zbudować co tylko się chce i kiedy się chce. Ceną za to jest jednak pogorszenie walorów użytkowych i dlatego przed podjęciem ostatecznej decyzji należy dokonać porównania branych pod uwagę systemów oraz rozważyć wszystkie za i przeciw. Ponadto w najnowszych wersjach WordPressa również można wiele zdziałać, trzeba tylko dobrze znać się na budowie motywów i wtyczek.
- Pomoc — otwartym systemom CMS typu WordPress czasami wytyka się brak pomocy technicznej ze strony programistów, która jest oferowana z każdym komercyjnym systemem. Kontrargument, jaki przeciwstawia się tym twierdzeniom, jest taki, że za

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

pieniądze zaoszczędzone na kupnie licencji można wynająć konsultanta lub programistę. Za pieniądze zaoszczędzone na opłatach licencyjnych można utworzyć własne modyfikacje, jeśli nie znajdzie się darmowych rozwiązań.

WordPress jest znakomitym systemem CMS, który szczególnie dobrze sprawdza się w budowie witryn do publikacji artykułów. Niezależnie od tego, czy wydaje Ci się, że system ten jest dobrym wyborem do realizacji określonego projektu, czy nie, zawsze warto zastanowić się, czego przede wszystkim tak naprawdę potrzebujesz. Dopiero potem poszukaj idealnego CMS-a i przemyśl, jak będziesz go używać do realizacji zamierzonych celów.

# LISTA PUNKTÓW DO SPRAWDZENIA ZANIM WYBIERZE SIĘ WORDPRESSA

Rozważasz użycie WordPressa jako systemu CMS do budowy swojej witryny? Poniższa lista pomoże Ci podjąć decyzję. Warto rozważyć wszystkie wymienione punkty, aby w czasie pracy uniknąć niemiłych niespodzianek.

**Podstawowe kwestie**

- Czy do realizacji projektu na pewno potrzebny jest system CMS? Czasami lepiej jest utworzyć zwykłe strony HTML.
- Czy WordPress jest odpowiednim CMS-em do budowy takiego serwisu? Możliwe że tak, ale w niektórych przypadkach mogą być lepsze rozwiązania.



**Panel administracyjny WordPressa**

- Jakich dodatkowych funkcji potrzebujesz w panelu administracyjnym?
- Czy będziesz musiał użyć wtyczek (albo napisać własny kod), aby zredukować liczbę dostępnych opcji? Panel administracyjny niedoświadczonym użytkownikom może wydawać się skomplikowany.
- Jeśli planujesz używać archiwów użytkowników, zastanów się, jaką nazwę nadasz każdemu użytkownikowi.
- Czy chcesz umożliwić modyfikowanie menu, teł oraz nagłówków?
- Czy ma to jakieś znaczenie, że panel administracyjny ma styl WordPressa? Jeśli tak, musisz to zmienić i nie zapomnij przy okazji o formularzu logowania.

**Kategorie i tagi**

- W jaki sposób planujesz używać kategorii i tagów? Czy będziesz potrzebować dodatkowych taksonomii, np. osobnych grup tagów albo hierarchicznych kategorii? Czy taksonomie te powinny dać się bezproblemowo przenieść do nowego motywu w razie potrzeby?
- Jakich kategorii użyjesz i czy potrzebne będzie niestandardowe kodowanie kategorii albo utworzenie archiwów kategorii?
- Jak powinny być używane tagi i w jaki sposób objaśnisz zasady ich użycia użytkownikom?

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

**Strony i wpisy**

- Czy potrzebujesz własnych pól?
- Czy do działania niektórych części witryny potrzebne będą specjalne szablony stron? Jeśli tak, to utwórz je już na samym początku.
- Jakie są wzajemne relacje między stronami? Co będzie stroną nadrzędną, a co podrzędną? Powinieneś znać hierarchię i wiedzieć, jak będę prezentowane poszczególne sekcje.
- Jak będą prezentowane wpisy? Czy wymagają jakiegoś specjalnego traktowania?
- Czy będziesz używać własnych formatów wpisów? Jeśli tak, to czy te wpisy będą odróżniały się wyglądem od innych wpisów?
- Czy potrzebujesz dodatkowych rodzajów treści oddzielnych od wpisów i stron? Jeśli tak, to powinieneś zaplanować, jak będziesz ich używać, oraz zdecydować, czy mają być przenośne.
- Co będzie się znajdowało na stronie głównej? Statyczna strona, lista najnowszych artykułów czy coś jeszcze innego?

**Inne kwestie**

- Z góry zaplanuj strukturę bezpośrednich odnośników i już jej później nie zmieniaj. Możesz potrzebować wtyczek, aby ją dostosować do swoich potrzeb.
- Czy potrzebne będą jakieś skróty kodowe pozwalające przyspieszyć publikowanie tekstów i uniknięcie pisania kodu HTML (zobacz podrozdział o skrótach kodowych w rozdziale 13.)? Jeśli tak, to przemyśl też, w jaki sposób zachowasz te skróty, gdy będziesz zmieniać motyw.
- Czy potrzebujesz jakichś funkcji, których nie ma w WordPressie? Czy da się je zrealizować za pomocą wtyczek? Czy wtyczki te nie spowodują problemów z aktualizacją systemu w przyszłości?
- Jeśli projekt ma być poddany lokalizacji, to czy dostępne są funkcje językowe? Jest mnóstwo plików językowych dla WordPressa, ale co z wtyczkami i motywami? Czy będziesz musiał w tej kwestii wykonać dodatkową pracę?



# OGRANICZANIE WORDPRESSA DO MINIMUM

Tworząc witrynę dla klienta albo kogoś znajomego, musisz zastosować nieco inne podejście, niż gdy tworzysz serwis dla siebie. To, że Ty jesteś zaawansowanym użytkownikiem, nie oznacza, że wszyscy tacy są. Dlatego należy okroić system do niezbędnych składników i usunąć z niego wszystkie niepotrzebne opcje.

Najważniejszą czynnością, od której należy zacząć ograniczanie WordPressa, jest zredukowanie uprawnień użytkowników. Jak napisałem w rozdziale 1., nie wszyscy użytkownicy muszą być administratorami. Do większości prac wystarczy konto redaktora, a czasami jakieś nawet z mniejszymi uprawnieniami. Im mniejsze uprawnienia ma użytkownik, tym mniej opcji wyświetla mu się w panelu administracyjnym. I bardzo dobrze.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Nie trzeba chyba nawet wspominać, że włączone powinny być tylko te wtyczki, które są używane. Niepotrzebne wtyczki nie tylko spowalniają działanie witryny, ale również zagracają panel administracyjny swoimi opcjami. Dlatego wyłącz wszystkie wtyczki, których nie używasz.

## DOSTOSOWYWANIE PANELU ADMINISTRACYJNEGO

Panel administracyjny WordPressa można dostosować do swoich potrzeb, korzystając z jednej z wtyczek CMS. Jest ich kilka, ale ja polecam WP-CMS Post Control (*http://wordpress.org/extend/plugins/wp-cms-post-control*). Dzięki niej można m.in. ukryć przed użytkownikami wybrane elementy panelu administracyjnego, wyłączyć funkcję automatycznego zapisywania wpisów (która czasami jest uciążliwa), wybrać mechanizm wysyłania plików itd. Za jej pomocą można sprawić, że interfejs WordPressa będzie wyglądał trochę bardziej zachęcająco dla początkującego użytkownika. Najbardziej podoba mi się opcja wiadomości, w której można zapisać informacje dla użytkowników i podać odnośniki do dodatkowych pomocy.

Istnieje kilka wtyczek umożliwiających ukrywanie przed użytkownikami wybranych elementów panelu administracyjnego. Warto się nimi zainteresować, gdy tworzy się dla swojej firmy albo klienta witrynę, która ma być używana przez długi czas. Uważaj tylko, żeby nie wybrać jakiejś „jednorazowej" wtyczki. WordPress jest cały czas rozwijany i jeśli wtyczki nie będą na bieżąco uaktualniane, mogą szybko przestać działać w którejś z kolejnych wersji systemu. Dodatkowo należy nauczyć klienta, jak się to robi, jeśli nie oferuje się stałej opieki nad witryną.



Używanie WordPressa jako systemu CMS ma największy sens wtedy, gdy ukryje się przed użytkownikiem wszystkie niepotrzebne rzeczy. Konkurencja może tego nie robić, ale jeśli używasz WordPressa do budowy półstatycznej witryny firmowej, to na pewno dobrze by było usunąć z widoku wszystko to, co użytkownikom nie jest potrzebne. Pamiętaj tylko o dokonywaniu aktualizacji.

W grę wchodzi też pozbycie się wybranych elementów panelu administracyjnego samodzielnie. Mimo że niektóre moduły można schować za pomocą opcji ekranowych znajdujących się na wysuwanej karcie u góry ekranu, to niektóre z nich chciałoby się usunąć całkowicie, tak żeby w ogóle nie były nigdzie dostępne. Sposób postępowania w takiej sytuacji zależy od tego, co chcemy usunąć, ale w wielu przypadkach należy użyć funkcji `remove_meta_box()`. Aby usunąć moduł *Wypisy*, należy w pliku *functions.php* lub wtyczce, jeśli ma to być przenośna funkcja, umieścić poniższy kod:

```
function smashing_remove_post_excerpt() {
    remove_meta_box( 'postexcerpt' , 'post' , 'normal' );
}
add_action( 'add_meta_boxes' , 'smashing_remove_post_excerpt' );
```

Jest to funkcja wywołująca funkcję `remove_meta_box()`. Jej wywołanie podłączone zostało do haka `add_meta_boxes`, który dotyczy ekranu edycji w panelu administracyjnym.

Można także usunąć widżety, których moim zdaniem użytkownicy nie powinni używać na stronach witryny. Poniższy kod również można umieścić w pliku *functions.php* lub wtyczce:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
function smashing_unregister_widgets(){
    unregister_widget( 'WP_Widget_Calendar' );
    unregister_widget( 'WP_Widget_Meta' );
}
add_action( 'widgets_init', 'smashing_unregister_widgets' );
```

Ta funkcja usuwa widżety *Kalendarz* i *Meta*. Podłączona została do haka `widgets_init` za pomocą funkcji `add_action()`.

I jeszcze jedna rzecz. Jeśli tworzysz witrynę przy użyciu motywu potomnego, to możliwe, że w tym motywie potomnym nie są potrzebne wszystkie obszary na widżety dostępne w motywie nadrzędnym. Możesz je wyrejestrować w pliku *functions.php*:

```
function smashing_remove_some_sidebars(){
    unregister_sidebar( 'name-of-sidebar' );
    unregister_sidebar( 'name-of-another-sidebar' );
}
add_action( 'widgets_init', 'smashing_remove_some_sidebars', 11 );
```

To bardzo przydatne narzędzie. Więcej na temat różnych możliwości WordPressa można przeczytać w dokumentacji na stronie *http://codex.wordpress.org*.



## USUWANIE FUNKCJI TYPOWYCH DLA BLOGA

Wiele funkcji WordPressa może być kompletnie niepotrzebnych, gdy używa się go jako systemu CMS do budowy prostej witryny. W statycznej firmowej witrynie zawierającej informacje o produktach funkcja komentarzy czy trackbacków może być niepotrzebna, a jedyną dynamiczną treścią mogą być najświeższe wiadomości lub informacje prasowe. Niektóre rzeczy są czasami po prostu niepotrzebne.

Jeśli więc tworzysz witrynę nie wymagającą tych wszystkich fajerwerków i funkcji, wystarczy, że nie dodasz ich do motywu. W kilku kolejnych podrozdziałach znajduje się opis modyfikacji, jakie należy wykonać w WordPressie, gdy ma on być używany jako CMS do budowy statycznych witryn.

### Pliki szablonowe

Można utworzyć tylko jeden plik szablonowy o nazwie *index.php*, który będzie obsługiwał wyświetlanie wszystkich list i wyszukiwarkę, albo podzielić go na kilka bardziej wyspecjalizowanych szablonów. Jeśli funkcja komentarzy jest niepotrzebna, to cały kod związany z jej obsługą również jest zbędny. Natomiast szablony stron są bardzo przydatne i najprawdopodobniej użyjesz jednego z nich jako statycznej strony głównej.

Możesz też usunąć tradycyjne informacje o każdym wpisie, jeśli ich nie potrzebujesz. Wielu osobom data publikacji wpisu i kategorie oraz wyświetlone nazwisko autora i tagi przywołują na myśl bloga. Usuń te części, jeśli ich nie potrzebujesz, albo je zmodyfikuj.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### Blogowy język

W WordPressie można znaleźć wiele słów i wyrażeń charakterystycznych dla blogów, np. słowa *kategorie* i *tagi* są obecne w bezpośrednich odnośnikach. Można to zmienić w ustawieniach bezpośrednich odnośników i np. zamiast *kategorie* wpisać słowo *wiadomości* lub *nowości*, natomiast *tagi* można zamienić na *pisaliśmy* albo *tematy*. Dobrym sposobem na utworzenie sekcji z najświeższymi wiadomościami w witrynie firmowej jest zapisanie wszelkich ważnych informacji o firmie i opisów produktów na stronach oraz utworzenie kategorii o nazwie np. *wiadomości* (i uproszczonej nazwie `wiadomosci`) i publikowanie w niej najświeższych informacji. Następnie w ustawieniach bezpośrednich odnośników można ustawić, aby adresy wpisów miały postać */wiadomosci/nazwa-wpisu/*. Można do tego celu użyć własnego formatu wpisów w postaci `/%category%/%postname%/`. Jest to bardzo przydatne i nie wymaga tworzenia osobnej sekcji dla wiadomości ani niczego takiego.

Gdyby było więcej kategorii z wiadomościami, byłby potrzebny jakiś bardziej precyzyjny sposób kontroli treści. W takim przypadku dobrym rozwiązaniem byłoby utworzenie szablonów stron wyświetlających najnowsze wpisy należące do wybranych kategorii. Każda taka strona, zawierającą odpowiednią pętlę, reprezentowałaby osobną sekcję. Wszystko sprowadza się do tego, czego potrzebujesz i jak dużej elastyczności wymagasz.

To skłania nas do zastanowienia się, co można zrobić przy użyciu statycznych stron oraz jak to, co początkowo zostało utworzone jako wpis bloga, może zostać wykorzystane do budowy bardziej tradycyjnej statycznej witryny.



## IDEALNA KONFIGURACJA PROSTEJ STATYCZNEJ WITRYNY

Użycie statycznych stron i kategorii do budowy sekcji najświeższych wiadomości w staroświeckiej statycznej witrynie jest bardzo dobrym pomysłem. Podejście to można zastosować np. w serwisach mających na celu promocję jakiegoś produktu, witrynach firmowych oraz innych, w których format blogowy się nie sprawdza.

Strony w WordPressie miały pierwotnie służyć do prezentowania treści statycznej. Dzięki możliwości utworzenia indywidualnego szablonu dla każdej strony (przypomnę, że dla stron tworzy się indywidualne szablony) można każdej stronie nadać dowolny wygląd. Można nawet kompletnie zrezygnować z ogólnego projektu, ponieważ w szablonie takim nie trzeba wywoływać funkcji wstawiających nagłówek i stopkę, można napisać dowolny kod HTML i w ogóle usunąć wszystko, co jest związane z WordPressem, i utworzyć coś całkiem innego. Szablony stron to potężne narzędzie, dzięki któremu możemy z łatwością wyświetlić treść za pomocą kilku pętli albo zawartość kanałów RSS z innych witryn. Szablony stron są początkowo puste i są one dla nas najważniejszym narzędziem do wykorzystania WordPressa jako systemu CMS. To dzięki nim można wyrwać się z blogowych ram będących pochodną przeszłości tego systemu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Dzięki temu, że jako stronę główną można w WordPressie wyświetlić dowolną stronę statyczną (należy skorzystać z odpowiednich ustawień w panelu administracyjnym), a listę wpisów wyświetlić na dowolnej innej stronie, to tak naprawdę w systemie tym można zrobić wszystko. Można nawet dodać paskudne intro we Flashu automatycznie odtwarzające dźwięk (ale naprawdę nie należy tego robić). Ogólnie rzecz biorąc, szablony stron w WordPressie są jak puste pliki HTML czy PHP, które dopiero trzeba wypełnić treścią niekoniecznie powiązaną z WordPressem, ale korzystając z nich, ma się do dyspozycji wszystkie dobrodziejstwa systemu!

Z drugiej strony, mamy tradycyjne wpisy blogowe. Muszą one należeć od jakiejś kategorii i wszystkie wyglądają mniej więcej tak samo. Oczywiście można ich używać, ale w półstatycznej witrynie lepiej jest używać stron.

---

*Oczywiście sytuacja wyglądałaby całkiem inaczej, gdybyśmy planowali publikować ogromne ilości treści, ale tutaj nie zajmujemy się tego typu sytuacją.*

---

Na stronach powinno się publikować statyczne treści, a we wpisach teksty typu najświeższe wiadomości. Jest to świetny model dla większości firmowych witryn, w których WordPress jest wykorzystywany jako CMS. Na stronach prezentowane są opisy produktów, a we wpisach publikuje się ogłoszenia, wiadomości i informacje prasowe. Zakładając, że projekt witryny jest już gotowy, należy postępować według poniższej procedury:



1. Zdecyduj, co będzie stroną, a co nie. Najczęściej stroną jest wszystko, co nie jest wiadomością lub ogłoszeniem. Utwórz odpowiednie strony i upewnij się, że mają właściwe uproszczone nazwy. Sprawdź też strukturę łączy bezpośrednich oraz czy adresy URL wpisów i kategorii wyglądają tak jak powinny. Możliwe, że aby osiągnąć żądany efekt, użyjesz jakiejś wtyczki, ale wszystko zależy od konkretnych potrzeb.
2. Utwórz szablony stron. Strona zawierająca opis profilu działalności firmy może mieć nieco inny styl niż strony produktów. W każdym z tych przypadków należy wyróżnić inne elementy oraz utworzyć odpowiednie podmenu i pola informacyjne pasujące stylem do reszty.
3. Utwórz kategorie, po jednej dla każdej sekcji wiadomości. Często tworzy się tylko jedną kategorię, np. *News* albo *Ogłoszenia*, chociaż czasami tworzy się obie lub jeszcze jakieś inne. W niektórych przypadkach wystarczy tylko jedna kategoria — np. Ogłoszenia — i wtedy można do sortowania wpisów w jej obrębie używać tagów, np. po jednym dla wiadomości, informacji prasowych, produktów, ogłoszeń itd. Oczywiście należy zadbać o to, aby listy kategorii i pojedyncze wpisy dobrze wyglądały.
4. Połącz wszystko, tworząc menu (użyj do tego funkcji tworzenia menu, aby klient mógł w przyszłości sam je modyfikować, ale na początek utwórz sam podstawowe menu) łączące strony i kategorie.

To wszystko, jeśli chodzi o budowę prostej witryny przy użyciu WordPressa jako systemu CMS. Oczywiście opisaną strukturę można dalej do woli rozbudowywać. WordPress jest bardzo elastyczny i nie musimy się z niczym ograniczać.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*Chcesz utworzyć prostą statyczną witrynę na sterydach? Utwórz własne typy wpisów, dzięki którym będziesz mieć lepszą kontrolę nad statyczną treścią.*

# BARDZIEJ ZAAWANSOWANE ROZWIĄZANIA

Czasami potrzebne jest coś więcej niż prosta statyczna witryna opisana wcześniej. Na przykład może być konieczne utworzenie dodatkowych typów wpisów, np. do wyświetlania list produktów mających przypisane własne taksonomie itp.

Oczywiście niektóre witryny wymagają więcej uwagi. Im bardziej zaawansowana jest witryna, tym więcej modyfikacji i poprawek trzeba dokonać, aby prawidłowo działała. Czasami trzeba w tym celu napisać własną pętlę albo utworzyć własny szablon strony. Może się też zdarzyć, że będziesz chciał dokonać zmian, ale bez ruszania kodu źródłowego. Wtedy mogą być przydatne widżety. Ich zaletą jest nie tylko to, że pozwalają na robienie wielu różnych ciekawych rzeczy, ale również to, że aby ich użyć, wystarczy je przeciągnąć myszą w wybrane miejsce, co na pewno się spodoba użytkownikom nie mającym wiedzy programistycznej.

W kolejnych podrozdziałach opisuję, jak dodać tego rodzaju funkcjonalność do prostej konfiguracji WordPressa jako systemu CMS.



## ZASTOSOWANIE WŁASNYCH TYPÓW WPISÓW I TAKSONOMII W WORDPRESSIE UŻYWANYM JAKO CMS

Własne typy wpisów są najlepszym dodatkiem do WordPressa od czasu tagów. Sposób ich tworzenia opisałem w rozdziale 6., dlatego też w tym podrozdziale opisuję tylko, jak je właściwie wykorzystać w WordPressie używanym jako CMS.

Możliwość tworzenia innych rodzajów wpisów niż tylko zwykłe wpisy i strony jest bardzo przydatna. Dzięki własnym typom wpisów można tworzyć dowolne rodzaje treści, nie mieszając jej ze zwykłymi wpisami i stronami. Kilka przykładów, jakie przychodzą mi w tej chwili do głowy:

- Produkty — można utworzyć typ wpisów o nazwie Produkty i używać go do tworzenia opisów produktów z tagami itd.
- Osoby — można utworzyć typ wpisów do opisywania osób.
- Portfolio — teraz prawdopodobnie portfolio masz zapisane w jakiejś kategorii, ale nie musi tak być. Jeśli chcesz je oddzielić od reszty treści, po prostu utwórz odpowiedni typ wpisów!
- Instrukcja obsługi — chcesz opublikować statyczną instrukcję obsługi? Nie musisz jej mieszać ze stronami, ponieważ możesz dla niej utworzyć osobny typ wpisów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Listę tę można by jeszcze długo ciągnąć. Najlepsze jest to, że dla każdego własnego typu wpisów w panelu administracyjnym jest tworzone osobne menu podobne do menu wpisów i stron. Dzięki temu użytkownik nie ma żadnego problemu z ich znalezieniem i używaniem, w odróżnieniu od własnych pól itp. metod.

Podobne zalety mają własne taksonomie, o których również była mowa wcześniej. Dzięki nim można np. utworzyć osobną grupę tagów dla produktów. Jest to doskonały sposób na uzyskanie dodatkowej kontroli nad sposobem prezentacji i sortowania treści. System WordPress udostępnia doskonałą mieszankę standardowych narzędzi, takich jak wpisy i strony oraz kategorie i tagi, a także umożliwia korzystanie z niestandardowych rozwiązań, jak własne typy wpisów i taksonomie. Czasami, aby wypracować sobie idealną konfigurację, potrzeba trochę czasu, ale warto go poświęcić.

---

*Tworząc własne typy wpisów i taksonomie, nie zapominaj o przenośności. Zazwyczaj własne typy wpisów i taksonomie muszą być dostępne także po zmianie motywu, a więc weź to pod uwagę. Więcej na temat przenośności piszę w rozdziale 8.*

---



## WYKORZYSTANIE WIDŻETÓW W CMS-IE

Widżety i obszary widżetów są jednym z naszych najlepszych przyjaciół, gdy używamy WordPressa jako systemu CMS. Może nie są one aż tak bardzo ważne w małych statycznych witrynach firmowych, ale w większych serwisach są nieocenione. Rozejrzyj się po witrynach dostępnych w internecie. Znajdziesz w nich (zwłaszcza na stronach głównych) rozmaite dodatki, takie jak obrazki zachęcające do czytania całych artykułów itp. Wiele z tych efektów można uzyskać dzięki widżetom.

### Proste zastosowanie obszarów na widżety

Najprostszym sposobem użycia obszarów na widżety jest rozmieszczenie ich w wielu różnych miejscach, w których może kiedyś ktoś zechce coś wyświetlić. Należy tylko pamiętać o tym, aby domyślnie obszar na widżety niczego nie wyświetlał, dzięki czemu jeśli będzie pusty, to nie będzie go widać. Jednak lepszym rozwiązaniem jest dokładne przemyślnie tej kwestii i umieszczenie obszarów na widżety tylko w tych miejscach, w których wiadomo, że będą potrzebne, a nie w tych, w których mogą być przydatne może za dwa lata. Do tej pory i tak zmienisz projekt witryny.

Zatem masz już zdefiniowane obszary widżetów. Jak ich użyć? Oczywiście należy na nie przeciągnąć widżety dostępne w panelu administracyjnym w sekcji *Wygląd/Widgety*. Standardowe widżety dostępne w WordPressie są dość proste i raczej niewyszukane. Na przykład widżet *Strony* nie pozwala nawet na wyświetlenie samych tylko stron, ponieważ automatycznie dodaje nagłówek `h2`, który przecież nie zawsze jest nam potrzebny. Jeśli chcesz mieć kontrolę nad sposobem wyświetlania treści w ten sposób, poszukaj wtyczek dodających odpowiednie widżety. Jest kilka takich, które wyświetlają listę stron bez dodawania nagłówka `h2`.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Zdecydowanie najbardziej przydatny ze wszystkich jest widżet Tekst. Można w nim używać kodu HTML, dzięki czemu bez problemu umieścisz w nim zarówno tekst, jak i obrazy. W widżecie tym można np. przedstawić najnowszą ofertę promocyjną (albo zareklamować stronę firmy na Facebooku) zaraz pod menu w nagłówku. W tym celu wystarczy tylko wpisać odpowiedni kod HTML w widżecie, a następnie zapisać go w wybranym miejscu na czas trwania kampanii. Gdy kampania się skończy, widżet można po prostu usunąć. Opis kilku przydatnych wtyczek znajdziesz w dodatku A.

## Dodawanie dynamizmu do widżetów

A może by tak umieścić puste obszary na widżety w różnych miejscach, w których mogłyby się znajdować różne elementy witryny? To mogłoby się sprawdzić, ale musiałbyś ich napchać wszędzie, gdzie się da, i trzeba by było wziąć pod uwagę resztę projektu oraz treść. Oczywiście i tak musisz to zrobić, ale jest też inne wyjście: zastępowanie części treści obszarami na widżety.

Jeśli prowadzisz witrynę magazynową, to możesz wyświetlać w niej wielkie reklamy, tzw. ***roadblocks***, które zajmują o wiele więcej miejsca niż zwykłe banery czy „wieżowce". Reklamy te zazwyczaj mają krótszy czas prezentacji, ale są dobrze płatne w porównaniu ze zwykłymi jednostkami.

Albo rozważmy taką sytuację: wyobraź sobie, że prowadzisz bloga o produktach firmy Apple i chcesz regularnie zamieszczać informacje o konferencjach WWDC lub nowych numerach pisma „MacWorld", aby nic nie ominęło Twoich czytelników. Jak? Oczywiście oblepiając witrynę promocyjnymi grafikami!



W oby tych przypadkach dodanie treści do obszarów na widżety umieszczonych wokół normalnych elementów witryny będzie słabym rozwiązaniem. O wiele lepsze byłoby zastąpienie części treści witryny obszarami na widżety. Pomysł ten można łatwo zrealizować dzięki temu, że obszary widżetów można wczytywać z treścią.

Poniżej znajduje się kod obszaru na widżety o nazwie *Teaser*, który zostanie załadowany wraz z normalną treścią wyświetlaną w tym miejscu na co dzień:

```
<?php
    if ( is_active_sidebar( 'Teaser' ) ) {
        dynamic_sidebar( 'Teaser' );
    } else { ?>
          [Normalna treść: łącza, nagłówki itd. ]
<?php }; ?>
```

W treści można wstawić, co tylko się chce: nagłówki, ankiety, listy, pętle, łącza, obrazy itd.

Gdy jednak w panelu administracyjnym do obszaru Teaser zostanie dodany widżet, nie zostanie wyświetlona domyślna treść, tylko system wyświetli ten widżet. Dzieje się tak dzięki użyciu instrukcji warunkowej zawierającej funkcję `is_active_sidebar()`. Gdy dodasz w widżecie wielką grafikę promującą najnowsze wydarzenie w branży Apple albo Twoją najnowszą książkę, zostanie ona wyświetlona w miejsce zwykłej treści. Gdy usuniesz widżet z tego obszaru, na miejsce powróci domyślna treść.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Sprytne, prawda? I na dodatek łatwe w obsłudze. Jeśli coś się nie uda, możesz usunąć widżet i wrócić do domyślnej treści.

Innym doskonałym zastosowaniem widżetów jest tworzenie menu nawigacyjnych. Dzięki nim użytkownik może dodawać elementy do menu samodzielnie, bez pomocy projektanta motywu.

# OBSŁUGA MENU

Menu musi dobrze wyglądać i być łatwe w obsłudze. Podczas gdy jest to ważne w każdej witrynie, to w przypadku serwisów, w których WordPress jest używany jako CMS, wydaje się nawet jeszcze ważniejsze. Dzięki znakomitej funkcji menu WordPressa, której opis znajduje się w rozdziale 6., tworzenie tego elementu jest bardzo łatwe. Ponieważ jednak jest to bardzo ważna część witryny, należy jej poświęcić trochę czasu.

Są dwa główne sposoby tworzenia menu. Pierwszy polega na użyciu obszaru menu i funkcji `wp_nav_menu()` wyświetlającej zdefiniowane menu. Zastosowanie tej funkcji — w odróżnieniu od drugiej metody, polegającej na wykorzystaniu obszarów widżetów — ogranicza się tylko do tworzenia menu. Ale w technice tej istnieje dodatkowa możliwość nadawania nazw menu, co ułatwia pracę użytkownikom.



Druga metoda polega na użyciu obszarów na widżety. Menu utworzone za pomocą widżetu można umieścić w każdym miejscu, w którym można umieścić widżet. Czasami takie rozwiązanie jest wygodne, chociaż daje ono mniejszą kontrolę nad menu. Mimo wszystko warto pamiętać o takiej możliwości, ponieważ dzięki niej można dodawać menu w różnych częściach witryny. Dzięki temu można umożliwić użytkownikom tworzenie własnych nawigacji, gdy zajdzie taka potrzeba, bez konieczności modyfikowania kodu czy dodawania funkcji `wp_nav_menu()` w różnych miejscach.

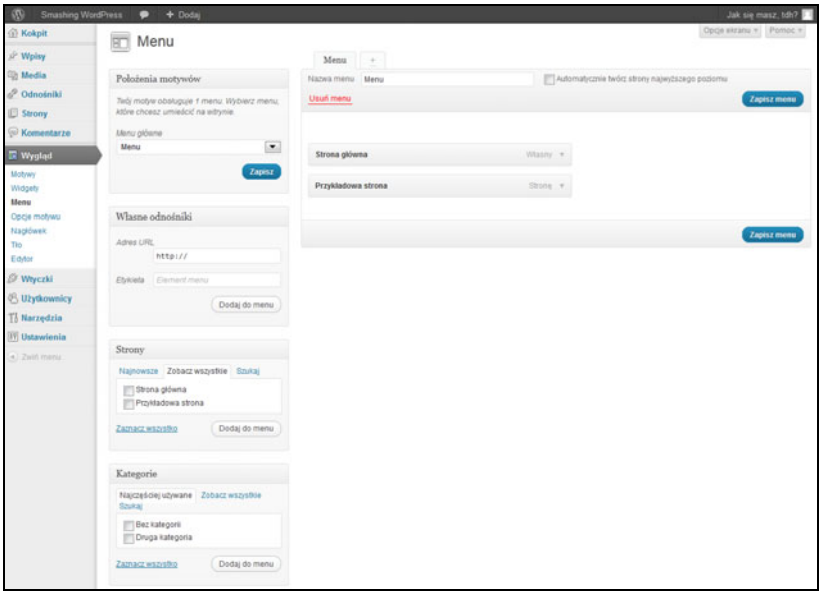
Najważniejsze jest jednak, aby wykorzystać funkcję menu, ponieważ dzięki temu administrator może z łatwością obsługiwać menu z poziomu panelu administracyjnego. Jeśli chcesz się dowiedzieć, czy w Twoim motywie włączona jest funkcja tworzenia menu, przejdź w panelu administracyjnym do sekcji *Wygląd/Menu* (rysunek 9.1).

# INTEGRACJA TREŚCI SPOZA WORDPRESSA

Czasami chcemy wyświetlić w WordPressie treść pochodzącą spoza systemu. To może być trudne, ponieważ nie wszystkie platformy dobrze ze sobą współpracują. Treść taką można spróbować wyświetlić w widżecie, dołączyć w całości za pomocą PHP, a jeśli wszystkie inne metody zawiodą — zaprezentować w ramce wewnętrznej.

Jeśli potrzebujesz treści tekstowej, a także graficznej i wideo, Twoim zbawcą mogą się okazać kanały RSS. Jeśli treść zewnętrzna pochodzi np. z serwisów typu Flickr, to możesz pobrać kanał RSS za pomocą specjalnej wtyczki (są dostępne wtyczki Flickr z obsługą RSS) albo

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 9.1. Interfejs tworzenia menu

skorzystać z wbudowanej w WordPressa funkcji wyświetlania kanałów RSS. Kilka przykładowych sposobów wyświetlania treści pobranej z kanałów RSS opisałem w rozdziale 13. Innym rozwiązaniem jest użycie wtyczki typu FeedWordPress (*http://wordpress.org/extend/plugins/feedwordpress/*), która cyklicznie sprawdza zawartość kanałów RSS i tworzy wpisy w bazie danych, których można używać w dowolny sposób, tak jak wpisów utworzonych bezpośrednio w WordPressie. Jest kilka podobnych rozwiązań tego rodzaju, więc jeśli chcesz publikować treść z kanałów RSS, to masz w czym wybierać.



Gdy już zdecydujesz się na jakiś sposób wyświetlania treści zewnętrznej, jest duże prawdopodobieństwo, że będziesz musiał utworzyć dla niej osobny szablon strony. Zdecydowanie zalecam to rozwiązanie zamiast używania wtyczek typu Exec-PHP umożliwiających uruchamianie kodu PHP we wpisach i na stronach, ponieważ wtyczki te czasami nie działają tak jak powinny.

Czasami rozwiązanie może polegać na utworzeniu wtyczki mającej pełnić rolę pośrednika. Robiłem to już kiedyś, gdy miałem witrynę opartą na systemie CMS i postanowiłem przenieść ją do WordPressa, ale musiałem poradzić sobie z ogromną bazą danych gier i informacji. Import tekstu i obrazów do WordPressa nie nastręczył mi trudności, ale wtedy jeszcze nie istniały elastyczne taksonomie i tagi, przez co musiałem nieźle się napocić przy grach. Rozwiązaniem było napisanie wtyczki wczytującej dane i wyświetlającej je przy użyciu skrótów kodowych. To było o wiele trudniejsze niż proste wyświetlenie zawartości kanału RSS, ale czasami po prostu nie da się przenieść treści do WordPressa, uprzednio go nie rozszerzając.

Jeśli wszystkie inne metody zawiodą, posłuż się fałszerstwem. Zrób stronę wyglądającą dokładnie tak, jakby należała do WordPressa, ale umieść ją poza motywem. Wiem, że to nie jest tak zabawne, ale czasami systemy nie chcą ze sobą współpracować i mimo że prawie zawsze da się je w końcu jakoś pogodzić, czasami po prostu szkoda na to czasu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# NIE ZAPOMNIJ DODAĆ INSTRUKCJI OBSŁUGI

Osobom obeznanym z internetowymi systemami publikacji treści obsługa WordPressa może wydawać się banalna. Jednak nie wszyscy się na tym znają. Dla osób z innych branż WordPress może być skomplikowany nawet pomimo faktu, że osoby te mogą sprawnie posługiwać się edytorami tekstu i innymi typowymi programami komputerowymi. W WordPressie są dostępne instrukcje obsługi dla użytkowników, ale czy to wystarczy?

Choć zasoby dostępne w witrynach WordPress.org i WordPress.tv mogą być bardzo przydatne, to myślę, że warto dodatkowo samodzielnie opracować krótki przewodnik po najważniejszych funkcjach systemu, zwłaszcza gdy używa się WordPressa jako systemu CMS do budowy statycznej witryny. Jeśli odeślesz użytkowników takiej witryny do dokumentacji WordPressa, mogą być bardzo zaskoczeni.

Jeśli jesteś programistą lub projektantem i często używasz WordPressa do budowy witryn internetowych, możesz napisać przewodnik dla początkujących zawierający opis najczęściej wykonywanych codziennych czynności. Poradnik taki może mieć postać prostego zwięzłego dokumentu, jak również przygotowanego do druku e-booka. Ważne jest, aby dało się go łatwo aktualizować, gdy pojawią się nowe wersje systemu. Ponadto jego struktura powinna pozwalać na dodawanie opisów specyficznych rozwiązań zaimplementowanych w serwisach tworzonych dla różnych klientów. Może w witrynie została utworzona kategoria wideo, która działa inaczej niż inne kategorie, albo może występuje często spotykany problem z własnymi polami i ich użytecznością? Dodaj opisy sposobów użycia wtyczek, widżetów i ustawień zaprojektowanych specjalnie dla klienta. Dzięki temu unikniesz konieczności odpowiadania na wiele pytań.



# OSTATNIE SŁOWO NA TEMAT UŻYWANIA WORDPRESSA JAKO SYSTEMU CMS

Często można spotkać opinię, że WordPress nie nadaje się do użycia jako system CMS. Oczywiście system ten nie w każdej sytuacji jest idealnym rozwiązaniem — żaden system nigdy taki nie będzie — ale łatwość, z jaką można utworzyć witrynę, używając WordPressa, jest wielką zaletą tego systemu. Jeśli doda się do tego prosty interfejs użytkownika i świetne możliwości rozbudowy dostępne dzięki wtyczkom i motywom, to można stwierdzić, że WordPress zasługuje na potraktowanie go jako poważnego kandydata do budowy każdego serwisu.

Jedną z zalet WordPressa używanego jako CMS stanowi to, że jest przyjazny dla wyszukiwarek. Dzięki dobrze napisanemu kodowi i pomocy paru wtyczek (poszukaj ich opisów w dodatku A) nie będziesz miał problemów z wdrapaniem się na wyżyny wyników wyszukiwania, oczywiście pod warunkiem że będziesz publikować dobrą treść. To właśnie z wyszukiwarek i portali społecznościowych pochodzi ogromna część ruchu w witrynach internetowych. Dlatego w następnym rozdziale pokażę Ci, jak zintegrować WordPressa z serwisami społecznościowymi.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 10 INTEGRACJA WORDPRESSA Z MEDIAMI SPOŁECZNOŚCIOWYMI

**BIORĄC POD UWAGĘ POPULARNOŚĆ** takich serwisów jak Facebook, Twitter czy Google+, nie można mieć wątpliwości, że media społecznościowe w tych czasach mają ogromne znaczenie. Jeśli potrafi się je dobrze wykorzystać, można dzięki nim zdobyć sporo ruchu na swoich stronach, jak również wypromować swoje treści, uzyskać teksty do publikacji, a także zachęcić użytkowników do kontaktu z nami. Jest czymś naturalnym, że większość obecnie działających w sieci witryn jest w jakiś sposób zintegrowana z popularnymi serwisami społecznościowymi, choćby poprzez umieszczenie na stronach przycisków polecających Facebooka czy Twittera.

Zanim zaczniesz poznawać różne techniki łączenia witryn z mediami społecznościowymi, powinieneś zainteresować się najbardziej oczywistym narzędziem tego typu, czyli kanałami RSS. W najprostszej formie kanałów tych można użyć poprzez standardowy widżet RSS WordPressa, dzięki któremu można łatwo i szybko wyświetlić zawartość dowolnego kanału RSS. Obecnie każda usługa sieci społecznościowych, każda aplikacja i w ogóle każda społeczność udostępnia kanał RSS, dzięki czemu w wielu przypadkach wystarczy tylko skorzystać z tej metody integracji. Za pomocą kanałów RSS można wyświetlić swoje najnowsze wpisy na Twitterze, jak również opublikować pamiętnik.

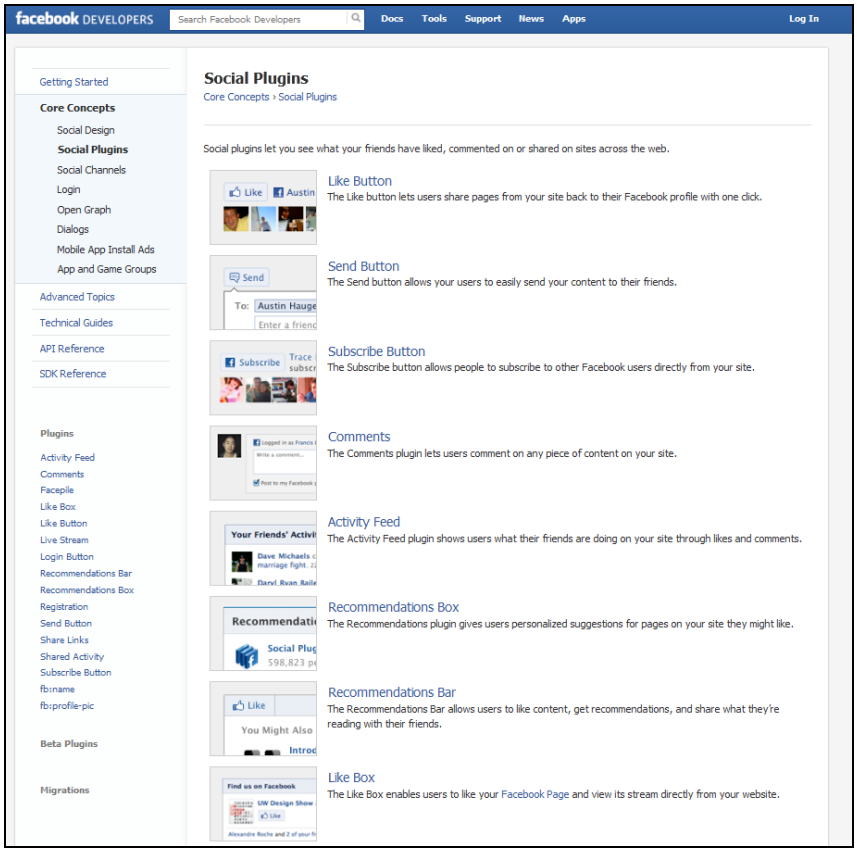
Jeśli zatem chcesz się pochwalić swoją najnowszą książką opublikowaną w aNobii (*www.anobii.com*), to zanim zaczniesz szukać wtyczek i szykować inne rozwiązania, zastanów się, czy nie wystarczy użyć RSS. Możliwe, że istnieje kanał RSS, w którym publikowane są Twoje najnowsze działania, publikacje itp.

Wystarczy gadania. Czas wziąć się do roboty!

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# INTEGRACJA WORDPRESSA Z FACEBOOKIEM

Zaczynam od Facebooka, ponieważ nie dość że w chwili pisania tej książki jest to najpopularniejszy serwis społecznościowy na świecie, to na dodatek jest on najłatwiejszy w integracji z witrynami internetowymi, przynajmniej w najprostszej postaci. Facebook udostępnia doskonałe narzędzia umożliwiające integrację jego narzędzi z witrynami internetowymi. Wystarczy zajrzeć na jego stronę wtyczek społecznościowych (*http://developers.facebook.com/docs/plugins/*), aby się przekonać, że jest w czym wybierać (rysunek 10.1).




Rysunek 10.1. Narzędzia Facebooka do użytku w witrynach internetowych

Najczęściej w witrynach internetowych używa się dwóch narzędzi promocyjnych Facebooka: przycisku *Lubię to* i strony dla fanów, tzw. fanpage'a.

## PRZYCISK LUBIĘ TO

Przycisk *Lubię to* Facebooka można zintegrować ze swoją witryną dosłownie w kilka sekund. Wystarczy wejść na stronę *http://developers.facebook.com/docs/reference/plugins/like* i utworzyć taki przycisk, jaki się chce mieć.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Nie przejmuj się polem *URL to Like* (adres do polubienia), ponieważ zaraz i tak pokażę Ci, jak je zamienić na tag `the_permalink()`. Wpisz w nim cokolwiek, żeby nie było puste. Ja np. wpisałem adres *http://tdh.me*.

Gdy klikniesz przycisk *Get Code* (pobierz kod), otrzymasz do wyboru trzy fragmenty kodu. Jeden będzie w języku HTML5 (najczęściej używany), drugi w XFBML (ang. *extended Facebook markup language*), a trzeci to zwykła ramka wewnętrzna (`iframe`). Najczęściej używany jest kod HTML5 i my też zdecydujemy się na jego użycie. Jak widać, kod ten składa się z dwóch części. Pierwsza z nich pobiera API JavaScript Facebooka do naszego serwisu:

```
<div id="fb-root"></div>
<script>(function(d, s, id) {
    var js, fjs = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) return;
    js = d.createElement(s); js.id = id;
    js.src = "//connect.facebook.net/en_US/all.js#xfbml=1&appId=218861188171083";
    fjs.parentNode.insertBefore(js, fjs);
}(document, 'script', 'facebook-jssdk'));</script>
```

Druga część jest realnie odpowiedzialna za wyświetlanie przycisku. W niej zapisane są zdefiniowane przez Ciebie ustawienia, np. czy mają być pokazywane twarze osób, które kliknęły przycisk, itp.:

```
<div class="fb-like" data-href="http://tdh.me" data-send="true" data-width="450"
    data-show-faces="true"></div>
```



Adres *http://tdh.me/* jest zapisany w atrybucie `data-href`. W jego miejscu należy użyć tagu `the_permalink()`:

```
<div class="fb-like" data-href="<?php the_permalink(); ?>" data-send="true"
↪data-width="450" data-show-faces="true"></div>
```

Wystarczy, że umieścisz ten kod wewnątrz pętli w plikach szablonowych (np. *single.php*), a na stronach przez nie obsługiwanych pojawi się przycisk *Lubię to*.

A do czego służy kod XFBML? Ten kod jest nieco dłuższy i warto z niego skorzystać, jeśli w swojej witrynie używa się wielu narzędzi Facebooka. Jeśli natomiast zadowalasz się jedynie opisanym wyżej przyciskiem, to lepiej pozostać przy HTML5, ponieważ kod XFBML trudniej jest zrozumieć, gdyż nie są w nim używane dobrze znane wszystkim znaczniki HTML. Gdybyś jednak zdecydował się na XFBML, to na stronie *http://developers.facebook.com/docs/reference/javascript* znajdziesz informacje na temat ładowania SDK JavaScript, które jest potrzebne do jego używania.

Oczywiście w tym przypadku również trzeba dostosować kod do użytku w WordPressie. Podobnie jak było z kodem HTML5, w odpowiednim miejscu należy zastąpić adres, w przykładzie *http://tdh.me/*, tagiem `the_permalink()`. Zmian należy dokonać w kodzie powodującym wyświetlenie przycisku:

```
<fb:like href="http://tdh.me" send="true" width="450" show_faces="true"></fb:like>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Wystarczy tylko zamienić adres `http://tdh.me` na wywołanie funkcji `the_permalink()` w atrybucie `href` w pętli i gotowe:

```
<fb:like href="<?php the_permalink(); ?>" send="true" width="450"
↪show_faces="true"></fb:like>
```

Korzystając z kodu w XFBML, można używać elementu `fb` do wstawiania różnych funkcji Facebooka — tu np. `fb-like`. Nie zapomnij tylko dołączyć SDK JavaScript, gdyż inaczej element ten nie będzie działać.

### WIDŻETY PROFILOWE

Jeśli chcesz promować swoją stronę dla fanów albo po prostu swoją zwykłą stronę na Facebooku, to również możesz skorzystać z łatwego w użyciu widżetu. Na stronie *http://www.facebook.com/badges/* dostępne są proste wizytówki, ale jeśli to dla Ciebie za mało, to są też ciekawsze rozwiązania. Chcesz wyświetlić kanał swojej aktywności albo umożliwić ludziom dopisywanie się do listy Twoich fanów (tzn. Twojej strony dla fanów)? Jeśli tak, poszukaj widżetu dla siebie na stronie *http://developers.facebook.com/docs/plugins/*.

Wyobraź sobie, że chcesz umieścić w witrynie przycisk *Lubię to* dotyczący Twojej strony na Facebooku. Narzędzie do tworzenia takiego przycisku działa tak samo jak opisane wcześniej, tylko nie trzeba używać dodatkowego tagu `the_permalink()`. Wystarczy zdefiniować styl, skopiować kod i wkleić go w widżecie tekstowym lub dowolnym innym miejscu.



## INTEGRACJA WORDPRESSA Z TWITTEREM

Bez wątpienia wzrost popularności Twittera ma wpływ na kształt sieci. Dziś mało kto nie korzysta z tego serwisu, przynajmniej spośród osób związanych z branżą internetową, a należy zauważyć, że serwis cały czas się rozwija. Mimo że 140 znaków dla osoby przyzwyczajonej do pisania 3000-znakowych wpisów na blogu może wydawać się mało, Twitter może być znakomitym narzędziem promocji dla wydawców internetowych.

Jeśli jesteś obecny na Twitterze osobiście lub jako marka, to prawdopodobnie zechcesz promować swoje twitterowe konto w swojej witrynie. Można to łatwo zrobić przy użyciu grafik, ale warto poszukać też innych rozwiązań. Zamiast promować konto Twittera na swojej stronie, możesz promować swoją stronę poprzez Twittera. W ten sposób ściągniesz na swoją witrynę osoby, które znają Cię tylko z tego serwisu społecznościowego. A to dopiero początek. Dostępnych jest wiele projektów i API zbudowanych na bazie Twittera i jego API, z których przynajmniej niektóre mogą Cię zainteresować.

Ustalmy jednak pewną rzecz: to nie jest książka o pisaniu aplikacji na Twittera. A jednak fajnie by było wykorzystać promocyjny potencjał tego serwisu społecznościowego w bardziej wyrafinowany sposób niż tylko wyświetlenie prostej grafiki *Obserwuj* na swoich stronach, prawda?

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## DODAWANIE PRZYCISKÓW I WIDŻETÓW TWITTERA DO STRONY

Obecnie równie łatwo można wyświetlać swoje wpisy z Twittera (dzięki widżetowi Twittera), jak i promować swoje publikacje na Twitterze (dzięki oficjalnemu przyciskowi *Tweetnij*). Można pobrać najnowsze wpisy i wyniki wyszukiwania, oznaczać swoje wpisy jako ulubione itd. Wszystko to dzięki prostemu widżetowi, który tworzy się na stronie *http://twitter.com/about/resources/widgets*. Po zdefiniowaniu stylu widżetu otrzymasz kod podobny do poniższego:

```
<script src="http://widgets.twimg.com/j/2/widget.js"></script>
<script>
new TWTR.Widget({
    version: 2,
    type: 'profile',
    rpp: 4,
    interval: 30000,
    width: 250,
    height: 300,
    theme: {
        shell: {
            background: '#333333',
            color: '#ffffff'
        },
        tweets: {
            background: '#000000',
            color: '#ffffff',
            links: '#4aed05'
        }
    },
    features: {
        scrollbar: false,
        loop: false,
        live: false,
        behavior: 'all'
    }
}).render().setUser('tdh').start();
</script>
```

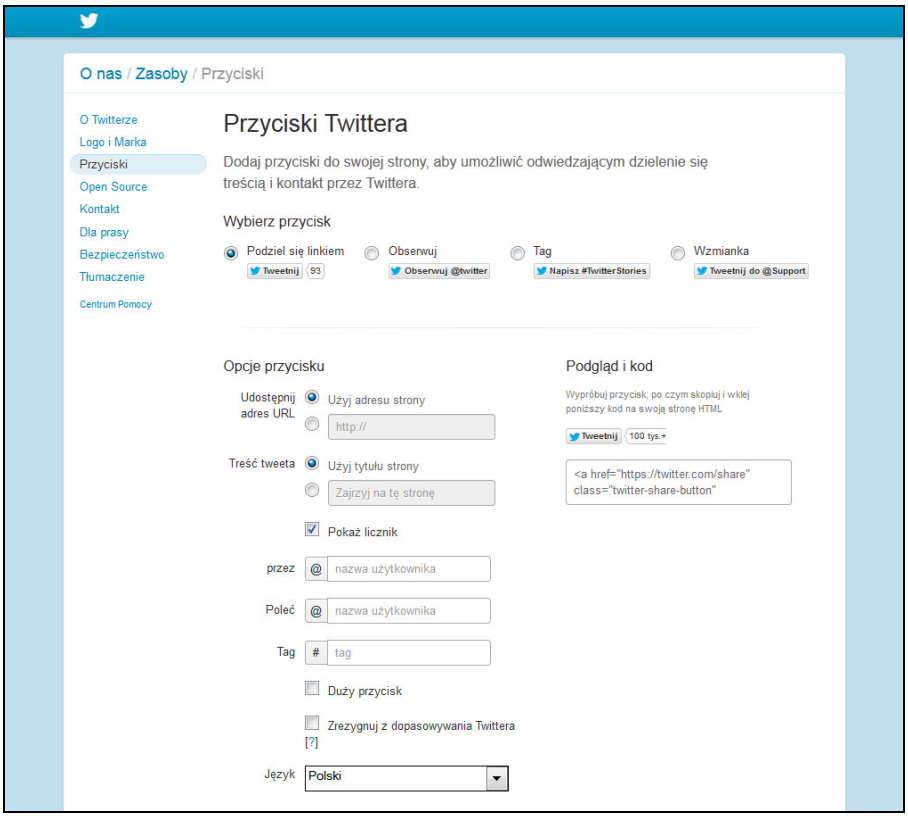
Wystarczy go wkleić w plikach szablonowych motywu albo widżecie tekstowym i gotowe. Jeśli postanowisz coś zmienić, najłatwiej jest wrócić do Twittera i od nowa zdefiniować cały widżet, ale można też przeanalizować ten kod, rozszyfrować jego działanie i dokonać zmian bezpośrednio w nim.

Tworzenie przycisku *Tweetnij* jest równie łatwe. Należy wejść na stronę *https://twitter.com/about/resources/buttons*, wybrać przycisk i dodać konto, które chcesz, aby zostało wspomniane (rysunek 10.2).

Otrzymasz krótki fragment kodu do umieszczenia np. w pliku *single.php*, aby ułatwić użytkownikom pisanie o Twoich wpisach:



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 10.2. Utworzenie przycisku Tweetnij jest bardzo łatwe

```
<a href="https://twitter.com/share" class="twitter-share-button" data-via="tdh"
  data-hashtags="smashing">Tweet</a>
<script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0];if(!d.

getElementById(id)){js=d.createElement(s);js.id=id;js.src="//platform.twitter.com/
  widgets.js";fjs.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-js");
  ↪</script>
```

Jeśli to Ci nie wystarcza, przeczytaj dokumentację Twittera na stronie *https://dev.twitter.com/docs/twitter-for-websites*. Są dostępne także bardziej zaawansowane opcje, jeśli masz ochotę pobawić się z API Twittera (*http://dev.twitter.com*).

## ROZSZERZENIA TWITTERA

Jest wiele widżetów, wtyczek, usług i aplikacji dla Twittera. Ta w miarę otwarta platforma mikroblogowania udostępnia wiele narzędzi ułatwiających budowanie różnych innych narzędzi, a ciągle rosnące zainteresowanie marką jeszcze przyspiesza jej rozwój. W paru kolejnych podrozdziałach opisane są przydatne usługi i narzędzia do skracania adresów URL.

Jeśli jednak szukasz opisów twitterowych wtyczek, zajrzyj do dodatku A.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## Usługi do wzbogacania treści witryn

Niektóre powiązane z Twitterem usługi wyróżniają się na tle innych tego typu usług. Poniżej znajduje się opis dwóch z nich:

- TweetMeme (*http://tweetmeme.com*) śledzi, co ciekawego dzieje się na Twitterze i ma sporo cech serwisu Digg.
- Twitterfeed (*http://twitterfeed.com*) publikuje łącza z dowolnego kanału RSS w koncie Twitter, dzięki czemu nie trzeba tego robić samodzielnie. Istnieją też inne serwisy tego typu, ale ten był pierwszy, ma obsługę OpenID itd.

## Usługi skracania adresów URL

Ograniczenie długości wpisów w Twitterze do 140 znaków sprawia, że nie da się publikować niektórych długich adresów. Gdy przydarzy Ci się ten problem, skorzystaj z usługi skracania adresów URL. Według jednych to konieczne zło, a dla innych wspaniałe rozwiązanie. Możesz uruchomić własną taką usługę, jeśli chcesz, ale jeśli wolisz, możesz też skorzystać z ogólnodostępnych usług. Pamiętaj, że czasami warto wybrać jedną usługę i korzystać tylko z niej, aby Twoi czytelnicy przyzwyczaili się do wyglądu Twoich odnośników. Warto też wiedzieć, że Twitter używa serwisu *t.co*, a więc możliwe, że w ogóle nie będziesz musiał korzystać z usług jakiegokolwiek serwisu tego typu. Ponadto także sam WordPress może skracać adresy URL. Zaloguj się w panelu administracyjnym i przejdź na stronę edycji wpisu, do którego chcesz uzyskać skrócony odnośnik. Następnie kliknij przycisk *Pobierz skrócony odnośnik* znajdujący się pod tytułem, jak pokazano na rysunku 10.3. Dzięki serwisom typu YOURLS (*http://yourls.org*) można nawet upiększyć adresy URL, jeśli ma się wolne domeny do wykorzystania.

Rysunek 10.3. Przycisk Pobierz skrócony odnośnik na stronie edycji wpisu

Oto kilka serwisów oferujących usługę skracania adresów URL:

- t.co (*http://t.co*) — oficjalny serwis skracania adresów Twittera, do którego można się dostać, korzystając z oficjalnych przycisków i aplikacji portalu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- TinyURL (*http://tinyurl.com*) — serwis, który był używany przez Twittera przed t.co. Wachlarz jego opcji jest dość skromny w porównaniu z konkurencją.
- bitly (*http://bitly.com*) — oprócz skracania adresów umożliwia także prowadzenie statystyk.
- goo.gl (*http://goo.gl*) — serwis skracania adresów URL należący do Google.

# INTEGRACJA WITRYNY Z GOOGLE+

Najnowsza i ciągle zyskująca popularność inicjatywa Google skierowana do użytkowników serwisów społecznościowych nosi nazwę Google+. Umieszczając przycisk *+1* na swoich stronach, masz szansę nie tylko otrzymać punkty widoczne w wyszukiwarce Google, ale również możesz zachęcić użytkowników do polecania Twoich publikacji w tej sieci społecznościowej.

Dodanie przycisku *+1* do strony jest łatwe, a jeśli masz większe wymagania, możesz też skorzystać z bardziej zaawansowanych opcji. Kod przycisku generuje się na stronie *http://www.google.com/webmasters/+1/button* i nie trzeba w nim nic zmieniać, podobnie jak np. w przypadku kodu Facebooka. Po prostu zdefiniuj ustawienia, a następnie skopiuj powstały kod i wklej go, gdzie chcesz. Jedna część tego kodu wczytuje kod JavaScript potrzebny do działania przycisku, a druga — wyświetla sam przycisk i należy ją umieścić tam, gdzie przycisk ten ma być wyświetlony.



Google zaleca, aby kod ładujący JavaScript umieścić bezpośrednio przed zamykającym znacznikiem elementu `body`, ale w WordPressie są lepsze sposoby na zarządzanie JavaScriptem. To jest kod, którego należy użyć:

```
<script type="text/javascript"
src="https://apis.google.com/js/plusone.js"></script>
```

O wiele lepiej zrobisz, jeśli dodasz go do motywu za pomocą funkcji `wp_enqueue_script()` służącej właśnie do dodawania skryptów JavaScript. Zatem w pliku *header.php* swojego motywu wpisz poniższy kod:

```
<?php wp_enqueue_script( 'gplus', 'https://apis.google.com/js/plusone.js', '', '',
↪TRUE ); ?>
```

Dzięki temu skrypt zostanie załadowany zgodnie ze standardami WordPressa, a nie wrzucony jak worek ziemniaków. Jeszcze lepszym sposobem na dodanie skryptu jest dodanie go do haka `wp_enqueue_script` w pliku *functions.php*. Możliwe, że w pliku tym i tak dołączasz już kilka innych skryptów w ten sposób, a więc wystarczy tylko dodać do nich jeszcze jeden skrypt. Jeśli nie, posłuż się następującym kodem:

```
function smashing_gplus() {
    wp_enqueue_script( 'gplus', 'https://apis.google.com/js/plusone.js', '', '', TRUE );
}
add_action( 'wp_enqueue_scripts', 'smashing_gplus' );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jeśli potrzebujesz odświeżenia pamięci, opis funkcji `wp_enqueue_script()` znajduje się w rozdziale 4.

Kod powodujący wyświetlenie samego przycisku wygląda tak:

```
<div class="g-plusone" data-annotation="inline"></div>
```

Należy go umieścić w miejscu, w którym ma być widoczny przycisk *+1*. Nie trzeba używać funkcji `the_permalink()`, ponieważ kod automatycznie sprawdza, na jakiej stronie znajduje się użytkownik.

# KORZYSTANIE Z ZEWNĘTRZNEJ OBSŁUGI KOMENTARZY

Biorąc pod uwagę, że w WordPressie istnieje możliwość zakorzeniania komentarzy i można używać doskonałych klas CSS do formatowania komentarzy, oraz wiedząc o powszechnie spotykanej obsłudze Grawatara (*http://gravatar.com*), zapewne zastanawiasz się, czy jest jeszcze jakiś sposób na ulepszenie funkcji komentowania. To proste: możesz użyć różnych wtyczek albo wynieść komentarze całkiem poza WordPressa. Jeśli chodzi o wtyczki, to można znaleźć rozwiązania pozwalające użytkownikom oceniać komentarze innych, jak również przenoszące dyskusje z Twittera i wszystko, co mieści się między nimi. Natomiast w drugim z wymienionych przypadków chodzi o to, żeby zarządzanie komentarzami zlecić zewnętrznej usłudze.



W dodatku A znajdziesz opisy kilku wtyczek, które pomogą Ci uatrakcyjnić komentarze w Twojej witrynie, ułatwią użytkownikom logowanie się itd. W tym podrozdziale zajmiemy się natomiast drugą z wymienionych opcji.

Skorzystanie z zewnętrznej usługi do obsługi komentarzy oznacza całkowitą rezygnację z tej funkcji w WordPressie. W dziedzinie tej liczą się w zasadzie dwaj najwięksi gracze: Disqus (*http://disqus.com*) i IntenseDebate (*www.intensedebate.com*). Pierwszy ma przewagę wieku, ale drugi należy do Automattic, a więc jest bardzo kuszącą propozycją dla wszystkich wielbicieli WordPressa. Każdy z tych systemów ma zalety i wady i każdy z nich jest aktywnie rozwijany, chociaż wielu twierdzi, że Disqus jest lepszy.

Aby rozpocząć korzystanie z któregokolwiek z wymienionych systemów, należy pobrać i zainstalować odpowiednią wtyczkę. Początki z oboma systemami są banalnie łatwe, chociaż niektórzy mieli problemy z lokalizacją i dostosowywaniem stylu.

---

*Co ciekawe, z usług tych można korzystać także w statycznych witrynach, dzięki czemu każda witryna może mieć funkcję komentarzy. Bardzo fajnie.*

---

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Korzystanie z zewnętrznego systemu komentarzy ma kilka zalet. Po pierwsze: spam jest zwalczany w bardzo skuteczny sposób. Oba systemy umożliwiają logowanie się na wiele sposobów, m.in. przy użyciu danych konta na Facebooku i Twitterze, oferują powiadamianie o odpowiedziach na e-maila, a także udostępniają kanały RSS i narzędzia do integracji z mediami społecznościowymi. Liczba funkcji, jakie otrzymasz, korzystając z usług Disqus lub IntenseDebate, jest tak duża, że aby mieć to samo bezpośrednio w WordPressie, musiałbyś zainstalować mnóstwo wtyczek. Co więcej, podobno w witrynach korzystających z tych systemów użytkownicy zostawiają więcej komentarzy i wydaje się, że nie jest to tylko chwyt marketingowy ich właścicieli.

Największą wadą tych systemów jest moim zdaniem to, że korzystanie z nich oznacza w istocie oddanie swojej treści do zarządzania komuś innemu. Jeśli usługa, z której korzystasz, ulegnie awarii albo jej właściciel po prostu ją wyłączy, to w najgorszym wypadku zostaniesz pozbawiony funkcji komentarzy, a w najlepszym wrócisz do standardowego rozwiązania WordPressa. To, że komentarze w przypadku awarii zewnętrznej usługi nadal działają tylko w standardowy sposób, jest dobrą wiadomością, ale może dziwić użytkowników. Do innych wad można zaliczyć krótkie przerwy w działaniu i trudności z dostosowaniem wyglądu systemu do motywu. Także problemy hostingu zdalnej usługi będą miały wpływ na działanie Twojej witryny, co również nie jest dobre.



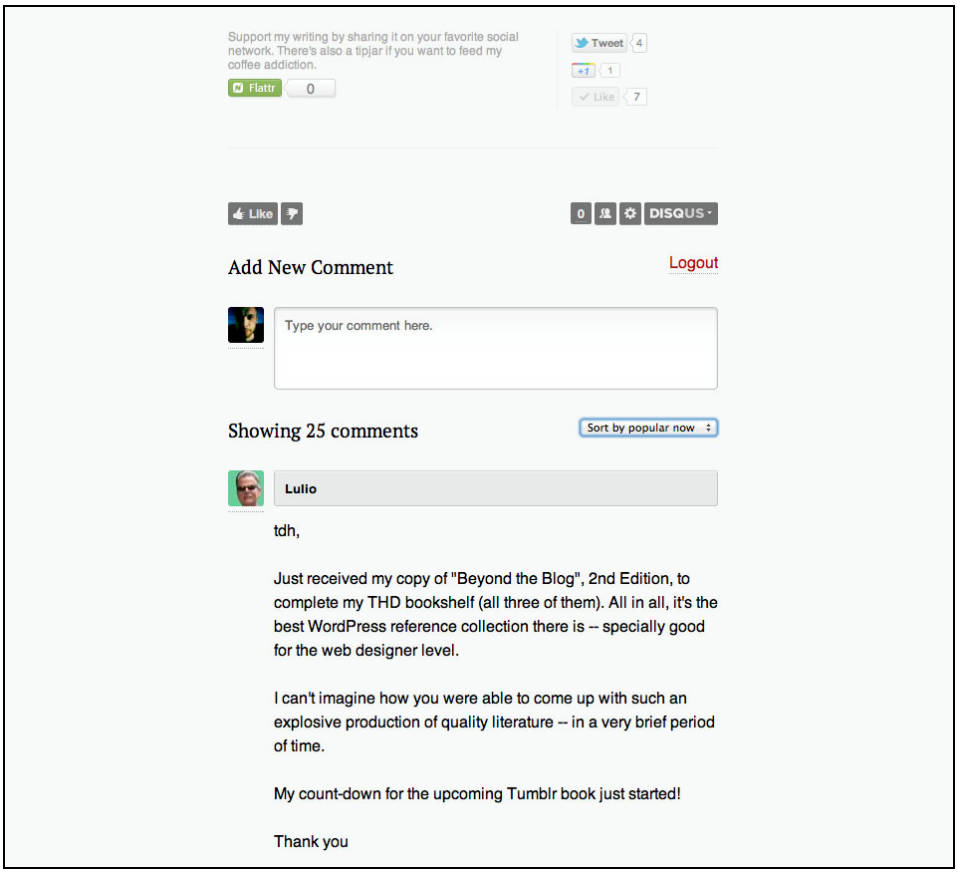
Gdybyś jednak zdecydował się na takie rozwiązanie, to biorąc pod uwagę wszystkie za i przeciw, które konkretnie byś wybrał? Ja częściej używam serwisu Disqus niż IntenseDebate (z moim własnym blogiem włącznie, czego dowód jest przedstawiony na rysunku 10.4) i jestem zadowolony, ale powinieneś sprawdzić każdy z nich i wybrać ten, który bardziej Ci się spodoba.

## JEDEN LOGIN DO WSZYSTKICH SERWISÓW

Pomysł stworzenia jednego globalnego systemu logowania jest świetny. Tylko pomyśl, czy nie byłoby wspaniale móc zalogować się w każdym miejscu przy użyciu tych samych danych? Nie musiałbyś zapamiętywać dziesiątek haseł i loginów i nie obawiałbyś się niebezpieczeństwa związanego z używaniem tego samego hasła do zbyt dużej liczby serwisów (choć z drugiej strony przecież i tak teraz miałbyś dostęp do wszystkich dzięki jednemu hasłu). W pomyśle tym chodzi jednak o to, aby nieliczni usługodawcy obsługujący takie superkonta byli maksymalnie bezpieczni. Jedyną możliwością złamania ich zabezpieczeń powinno być wykorzystanie błędu ludzkiego, a dokładniej mówiąc — nieuwagi samego użytkownika. Porównując ryzyko złamania zabezpieczeń jakiejś niewielkiej witryny do ryzyka złamania hasła do superkonta, można stwierdzić, że cały pomysł nie jest pozbawiony sensu.

Dlatego właśnie warto zainteresować się projektem OpenID (*www.openid.net*), którym interesują się nawet takie giganty jak Yahoo!, Google i Microsoft. Z tego samego powodu Facebook uruchomił usługę Facebook Connect (*https://developers.facebook.com/docs/guides/web*) do tworzenia ogólnego loginu przy użyciu konta na Facebooku. Podobne rozwiązanie ma też Twitter — Sign In with Twitter (*https://dev.twitter.com/docs/auth/sign-twitter*) — i wiele innych portali.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974






Rysunek 10.4. Kilka przycisków mediów społecznościowych i formularz komentarzy Disqus

Możesz się zastanawiać, po co w ogóle używać własnych narzędzi logowania, skoro można oprzeć się na barkach takich gigantów. W większości witryn opartych na WordPressie zwykłe logowanie jest zarezerwowane dla administratorów i redaktorów. Rzadko się zdarza, aby było wymagane tylko po to, aby pozostawić komentarz. Najczęściej wystarczy wpisać imię i podać adres e-mail. Jeśli jednak chcesz umożliwić użytkownikom logowanie się w witrynie, warto zainteresować się jednym z globalnych rozwiązań. Polecam OpenID, chociaż Facebook Connect jest bardziej przyjazny dla użytkownika (na razie), a poza tym Facebook również należy do OpenID.

Wkrótce będziesz używać swoich kont Google i Live.com do logowania się w różnych miejscach w sieci, także w Facebooku i Twitterze, a wszystko zapewne będzie połączone przez OpenId Foundation. Albo i nie. W każdym razie jeśli planujesz umożliwiać użytkownikom logowanie się w Twojej witrynie, powinieneś rozważyć opcję globalnego logowania. Są specjalne wtyczki, dzięki którym jest to proste (poszukaj w dodatku A), ale to nie oznacza, że nie powinieneś szukać jeszcze i innych rozwiązań. Poczytaj informacje opublikowane na stronach samych usług i podejmij decyzję, mając na uwadze potencjalne rejestracje użytkowników w przyszłości.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# JAK WAŻNE SĄ MEDIA SPOŁECZNOŚCIOWE

Obecnie sieci społecznościowe i wszystko, co jest w nich publikowane, dla wielu osób stanowi część internetowego życia. Kontaktujemy się z różnymi osobami, decydujemy, kogo śledzić lub z kim się zaprzyjaźnić, oraz subskrybujemy różne treści za pomocą przycisków „do lubienia" Facebooka i śledzenia kont użytkowników na Twitterze. Jest naturalne, że sieć społecznościowa jest częścią także Twoich witryn i że należy je promować także w tych mediach.

Nie jest to trudne. W sieci społecznościowej wybierasz, kogo chcesz śledzić, i jest duże prawdopodobieństwo, że osobom, które śledzisz, ufasz o wiele bardziej niż innym. Gdy zatem Twój znajomy na Facebooku publikuje link, jesteś o wiele bardziej skłonny go kliknąć niż jakiś inny link napotkany przypadkowo. Pomyśl: prędzej zdecydujesz się obejrzeć film polecany przez znajomego niż taki, którego reklamę właśnie obejrzałeś.

To właśnie dzięki temu takie narzędzia mediów społecznościowych jak przyciski *Lubię to* i *Tweetnij* oraz różne rodzaje zakładek dostały się do projektów witryn. Jeśli wszystko zrobisz dobrze, to odniesiesz sukces, ale jeśli się nie postarasz, to Twoje strony będą zabałaganione i mało popularne. Dlatego też w następnym rozdziale pokażę Ci, jak posługiwać się niektórymi elementami projektowymi, aby tworzyć dobrze wyglądające witryny internetowe.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 11 SZTUCZKI PROJEKTOWE

**GŁÓWNYM TEMATEM TEGO** rozdziału jest podnoszenie atrakcyjności motywu i wzbogacanie jego funkcjonalności tak, aby wyróżniał się z tłumu. Niektóre z opisanych tu technik są bezpośrednio związane z WordPressem i tym, co można zrobić z danymi przy użyciu dostępnych w nim narzędzi. Ponadto opisuję adaptacje innych metod, które również mogą być przydatne w pracy nad motywem. W rozdziale tym poznasz wiele ciekawych sztuczek, np. jak utworzyć projekt w oparciu o tagi, zmusić menu do działania w określony sposób, poprawnie dołączać pliki JavaScript i CSS, konstruować pomocne strony błędów 404 oraz wyświetlać reklamy w pętli.

Krótko mówiąc, przygotuj się na solidną dawkę wordpressowych sztuczek.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# ZWIĘKSZANIE KONTROLI NAD WPISAMI

W WordPressie kontrolę nad wpisami można zwiększyć na kilka sposobów. W tej części rozdziału zajmiemy się trzema następującymi metodami: tworzeniem projektów w oparciu o tagi, użyciem własnych pól oraz dodawaniem własnych taksonomii. Nie opisuję formatów wpisów, ponieważ zrobiłem to już w rozdziale 6., ale one również są doskonałym sposobem na uzyskanie większej kontroli nad wpisami.

## TWORZENIE PROJEKTÓW OPARTYCH NA TAGACH

Centralny punkt większości witryn internetowych opartych na WordPressie stanowią kategorie i strony. Kategorie umożliwiają wyświetlanie różnych list wpisów przy użyciu tagów szablonowych, jak np. `in_category()`. Dzięki temu archiwa kategorii można łatwo zamienić w sekcje witryny, a wpisom należącym do tych sekcji zdefiniować dowolne formatowanie, np. kolorystykę, ikonę lub cokolwiek innego.

Tagi mają trochę inne zastosowanie, ponieważ służą do precyzyjnego porządkowania indywidualnych wpisów. Z tego powodu w statycznych serwisach (raczej nienależących do nurtu Web 2.0) wydają się zbędne. Mimo to nawet w takich przypadkach można zrobić z nich dobry użytek. Są trzy główne sposoby wykorzystania tagów. Po pierwsze: można je wyświetlać na specjalnych stronach archiwalnych. Jak zapewne pamiętasz, sposób prezentacji tagów można kontrolować za pomocą pliku szablonowego *tag.php*, ale można też tworzyć pliki szablonowe o nazwach *tag-X.php*, gdzie *X* jest uproszczoną nazwą wybranego tagu. To co innego niż plik *category-Y.php*, gdzie *Y* jest identyfikatorem, a nie uproszczoną nazwą kategorii. Krótko mówiąc, jeśli chcesz, możesz utworzyć listę wpisów według tagów podobnie jak według kategorii. A to oznacza, że archiwa tagów podobnie jak archiwa kategorii mogą być sekcjami witryny.



Drugi sposób kontrolowania treści według tagów polega na użyciu tagu warunkowego `is_tag()`. Sposób jego działania jest zbliżony do `is_category()`, a więc należy mu przekazać uproszczoną nazwę tagu:

```
<?php is_tag( 'pirates' ); ?>
```

Oczywiście do tego potrzebne są jeszcze jakieś warunki. W ten sposób najczęściej pracuje się z kategoriami, ponieważ większość dobrze zbudowanych motywów do WordPressa zawiera jak najmniejszą liczbę kategorii. Tagi jednak są inne i można sprawdzać istnienie kilku tagów naraz. W tym celu należy użyć tablicy:

```
<?php is_tag( array( 'pirates', 'ninjas', 'mushrooms' ) ); ?>
```

Ten kod zwróci `true` dla stron archiwów tagów `pirates`, `ninjas` i `mushrooms`.

Trzeci sposób użycia tagów polega na wykorzystaniu faktu, że funkcja `post_class()` zwraca wszystkie tagi wpisu jako klasy elementu `div` będącego kontenerem tego wpisu. Dzięki temu mamy do dyspozycji wiele dodatkowych klas, które możemy wykorzystać do pisania arkuszy stylów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Poniżej np. znajduje się wpis o identyfikatorze 129, o czym można się dowiedzieć dzięki atrybutowi `id="post-129"` oraz przypisanej mu klasie `post-129`:

```
<div id="post-129" class="post-129 post hentry category-news tag-pirates
    tag-ninjas tag-mushrooms">
    <!-- Treść wpisu -->
</div>
```

Automatycznie tworzonych jest wiele klas, po jednej dla każdej kategorii i każdego tagu. W tym przypadku kategoria ma uproszczoną nazwę `news`, a więc funkcja `post_class()` utworzyła klasę `category-news`. To samo dotyczy tagów: mamy klasy dla `pirates`, `ninjas` i `mushrooms`, z których każda jest poprzedzona przedrostkiem `tag-`. Dlatego klasy tagów w tym przykładzie to: `tag-pirates`, `tag-ninjas` oraz `tag-mushrooms`.

Dzięki tym wszystkim klasom mamy bardzo precyzyjną kontrolę nad wpisami. Gdybyśmy chcieli niektóre wpisy wyróżnić, ale nie przypisywać ich w tym celu do żadnej kategorii, to najlepszym wyjściem byłoby przypisanie im jakiegoś tagu o nazwie w rodzaju „wyróżnione tagi" lub podobnej. Przypuśćmy, że chcemy między wpisami wyświetlić tekst „Dziękujemy naszym sponsorom", ale tak, żeby czytelnicy nie mylili go z właściwą treścią. Problem ten można rozwiązać poprzez dodanie tagu np. `sponsor` i napisanie kodu CSS zmieniającego wygląd wszystkich wpisów mających przypisany ten tag, czyli mających klasę `tag-sponsor`.



Podobnie jak w przypadku kategorii całą witrynę można zbudować na bazie tagów. Moim zdaniem z dodawaniem kategorii należy naprawdę **bardzo** uważać. Lepiej jest je zachować dla głównych sekcji witryny. Jeśli potrzebujesz kontroli nad indywidualnymi wpisami, używaj do tego tagów i kodu CSS.

## UŻYWANIE WŁASNYCH PÓL

Przypuśćmy, że z jakiegoś powodu nie chcemy opierać projektu witryny na tagach. Wyobraźmy sobie, że chcemy wyświetlać reklamy jako sponsorowane wpisy, które powinny odróżniać się od zwykłych wpisów, aby nie wprowadzać czytelników w błąd, ale jednocześnie nie chcemy, aby w chmurach tagów pojawiał się nam tag przeznaczony tylko do oznaczania reklam. Rozwiązaniem tego problemu jest wykluczenie takiego tagu z chmur. Poniżej znajduje się przykładowy kod wykluczający tag `sponsor`:

```
<?php wp_tag_cloud( 'exclude=sponsor' ); ?>
```

Może w Twojej sytuacji to konkretne rozwiązanie nie będzie właściwe i może nie masz zamiaru używać tagów w taki właśnie sposób, ponieważ wykorzystujesz je inaczej, ale zechcesz uzyskać podobny efekt. Użycie kategorii nie wchodzi w grę, ponieważ chcesz mieć kontrolę nad tym, w których sekcjach witryny wpisy reklamowe mają się pojawiać. W takim przypadku rozwiązaniem może być użycie własnych pól. W opisanym poniżej przykładzie tworzone jest własne pole o nazwie `Poststyle`. Należy je dodać tylko raz w dowolnym wpisie i nadać mu wartość. W tym przypadku zostanie użyta wartość `ad` (jeśli nie pamiętasz, na czym polega działanie pól własnych, zajrzyj do rozdziału 4.).

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W wartości pola `Poststyle` należy wpisać klasę, jakiej chce się używać. Później klasę tę będzie można łatwo pobrać, sprawdzając wartość pola `Poststyle` w każdym wpisie i zwracając ją. Przypomnę, że ostatni parametr `true` oznacza, że wyświetlona zostanie tylko jedna wartość, gdyby było ich więcej. W tym przypadku nie ma to znaczenia, ponieważ chcemy mieć tylko jedną wartość, której użyjemy jako klasy dla konkretnego wpisu. To oznacza, że jeśli polu `Poststyle` nadamy wartość `ad`, to znaczy, że chcemy utworzyć właśnie taką klasę, dla której napisaliśmy już odpowiedni arkusz stylów (oznacza to również, że dodając inne klasy, można załatwić wiele innych spraw, ale to już inna historia).

Kod wyświetlający wartość naszego pola jest następujący:

```
<?php
    $specialstyle = get_post_meta( get_the_ID(), 'Poststyle', true );
    echo $specialstyle;
?>
```

Jak dodać to pole do listy klas zwracanych przez tag `post_class()`? Poniżej znajduje się kod generujący identyfikator i klasy w elemencie `div` będącym kontenerem wpisu:

```
<div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
```



Teraz przypomnij sobie, że funkcja `post_class()` przyjmuje parametry określające dodatkowe klasy. Gdybyśmy zatem chcieli dodać klasę `turtles`, napisalibyśmy taką instrukcję:

```
<?php post_class( 'turtles' ); ?>
```

Tak wywołana funkcja `post_class()` oprócz standardowego zestawu klas dodałaby jeszcze klasę `turtles`. Tylko że my nie chcemy klasy `turtles`, ale wartość pola `Poststyle`. W tym celu funkcji `post_class()` przekażemy kod zwracający wartość tego pola, co na szczęście nie będzie trudne. Nie możemy jednak wpisać tego kodu ot, tak sobie. Musimy go trochę zmodyfikować:

```
<?php $specialstyle = get_post_meta( get_the_ID(), 'Poststyle', true ); ?>
```

Usunęliśmy instrukcję `echo`, ponieważ funkcja `post_class()` zwraca to, co zostanie jej przekazane. Na koniec wystarczy tylko usunąć znaczniki PHP i można działać:

```
<div id="post-<?php the_ID(); ?>" <?php post_class( $specialstyle =
↪get_post_meta( get_the_ID(), 'Poststyle', true ) ); ?>>
```

Ten kod spowoduje dodanie wartości pola `Poststyle` do listy klas elementu `div` będącego kontenerem wpisu w podobny sposób, jak wcześniej dodany został parametr `turtles`.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## PODPINANIE SIĘ DO FUNKCJI BODY_CLASS(), POST_CLASS() ORAZ COMMENT_CLASS()

Pozostajemy przy temacie funkcji `post_class()` i generowania przez nią klas, których można używać w CSS. Tym razem pokażę Ci, jak można się do niej podpiąć za pomocą haka `post_class`. Wszystko, co tu jest napisane, dotyczy także funkcji `body_class()` dla elementu `body` i funkcji `comment_class()` dla komentarzy. Wszystkie te trzy funkcje generują odpowiednie do sytuacji zestawy klas i każdej z nich można przekazywać parametry. Dla przypomnienia: poniższe wywołanie funkcji `body_class()` do standardowego zestawu klas dla elementu `body` doda jeszcze klasę `solid`.

```
<body <?php body_class( 'solid' ); ?>>
```

Aby podpiąć się do funkcji `body_class()`, `post_class()` oraz `comment_class()`, należy użyć haków o takich samych nazwach. Poniższy kod, który mógłby być użyty w pliku *functions.php*, dodaje klasę `solid` do funkcji `body_class()` przy użyciu filtra:

```
// Dodaje dodatkowe klasy do funkcji body_class()
add_filter( 'body_class', 'smashing_classes' );
function smashing_classes($classes) {
    // Dodaje klasę solid do tablicy $classes
    $classes[] = 'solid';
    return $classes;
}
```



Dzięki temu nie trzeba nic dodawać do wywołania funkcji `body_class()` w motywie, co sprawiłoby, że kod motywu byłby trochę prostszy. Te same zasady dotyczą funkcji `post_class()` i `comment_class()` i ich haków. Wystarczy tylko zastąpić nazwę `body_class` w funkcji `add_filter()` nazwą `post_class` lub `comments_class`.

Listy klas generowanych przez funkcje `body_class()`, `post_class()` oraz `comment_class()` można znaleźć w dokumentacji na następujących stronach:

- `body_class()` — *http://codex.wordpress.org/Function_Reference/body_class*
- `post_class()` — *http://codex.wordpress.org/Function_Reference/post_class*
- `comment_class()` — *http://codex.wordpress.org/Function_Reference/comment_class*

## DODAWANIE WŁASNYCH TAKSONOMII

Kolejnym sposobem na zwiększenie kontroli nad wpisami jest dodanie własnych taksonomii. W tej metodzie można używać tagu warunkowego `taxonomy_exists()` zwracającego wartość `true`, gdy taksonomia podana w argumencie wywołania jest dostępna (pamiętaj tylko, że najpierw musisz tę taksonomię utworzyć zgodnie z opisem zamieszczonym w rozdziale 6.).

Podobnie jak każdy inny tag warunkowy `taxonomy_exists()` umożliwia sprawdzenie, czy obecna jest określona taksonomia. Jeśli tak, można coś zrobić, jeśli nie — to można zrobić coś innego. Niestety własne taksonomie nie są dodawane do funkcji `post_class()` wraz ze

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

standardowymi taksonomiami, ale w razie potrzeby można je dodać własnoręcznie. Można je dodać do elementu `div` za pomocą instrukcji `echo` albo zastosować podejście podobne do pokazanego wcześniej rozwiązania z funkcją `post_class()` i tagami.

Więcej na temat funkcji `taxonomy_exists()` można przeczytać w dokumentacji na stronie *http://codex.wordpress.org/Function_Reference/taxonomy_exists*.

---

*Formaty wpisów, o których była mowa wcześniej, również są w istocie taksonomiami, tylko takimi, których nie należy modyfikować. Jeśli potrzebujesz czegoś podobnego do formatu wpisów, ale nie chcesz ich używać, możesz zawsze utworzyć ich replikę w postaci własnej taksonomii.*

---

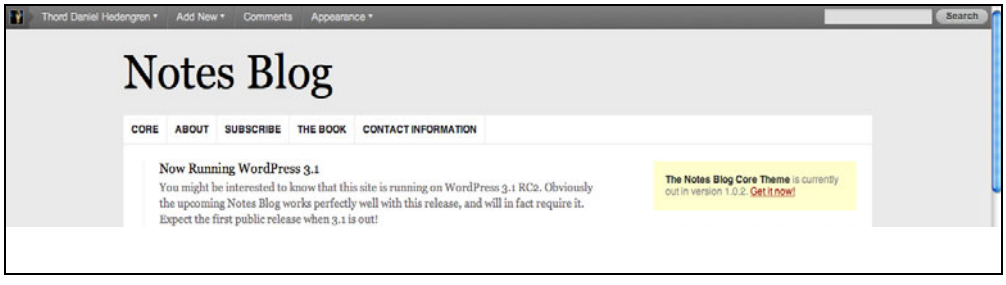
Możliwość kontroli sposobu prezentacji wpisów poprzez wykorzystanie tagów, własnych pól i taksonomii pozwala na wizualne wzbogacenie witryny i łatwe wyróżnienie ważnych elementów. Wyróżnianie wybranych części stron jest bardzo ważne, ale mimo to wszystko ostatecznie powinno do siebie pasować. W następnym podrozdziale zajmiemy się kolejną ważną częścią każdego projektu: menu.



# ULEPSZANIE MENU

Menu jest jedną z najważniejszych części projektu. Powinno zawierać wystarczająco dużo treści, aby posługujący się nim użytkownik mógł dotrzeć w głębsze partie witryny, a jednocześnie powinno być przejrzyste i łatwe w użyciu. Naprawdę trzeba się postarać, ponieważ dla użytkowników jest to bardzo ważna część witryny. Zresztą właśnie dlatego można znaleźć tak wiele różnych prób wizualnego uatrakcyjnienia tych elementów w witrynach. Niektórym udaje się to lepiej, a innym nieco gorzej, ponieważ niektórzy przedkładają wygląd nad walory użytkowe.

A przecież przyjemne w używaniu i intuicyjne menu (rysunek 11.1) również może być atrakcyjne wizualnie. Trzeba tylko wypróbować różne opcje, aby znaleźć najlepszą dla danego miejsca. Po prostu pamiętaj, że najważniejsze jest to, aby poruszanie się po witrynie było łatwe.




Rysunek 11.1. Proste menu

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Wyboru typu menu w WordPressie powinno się dokonywać na podstawie rodzaju witryny. Niektóre serwisy są zbudowane na bazie stron (zwłaszcza statyczne), podczas gdy inne mają o wiele luźniejszą strukturę i wystarczy na nich umieścić tylko chmurę tagów i pole wyszukiwarki. Między tymi dwoma ekstremami znajdują się witryny, w których sekcje oparte są na kategoriach (do których przypisane są wpisy), i zawierające strony, na których przedstawione są statyczne informacje typu „O nas". Tagi w takich witrynach zwykle są używane do precyzyjnego opisywania zawartości wpisów. Niezależnie od rodzaju witryny, jaką prowadzisz, menu powinieneś utworzyć przy użyciu specjalnie przeznaczonej do tego funkcji, której opis znajduje się w jednym z wcześniejszych rozdziałów. Dzięki temu budowa menu nie będzie nastręczać problemów, a gdy trzeba je będzie zmodyfikować, będzie to łatwe. Oczywiście nie ma rozwiązań uniwersalnych, ale w tej metodzie można mieszać kategorie ze stronami, wpisami, łączami itd.

Jakby tego było mało, trzeba jeszcze dodatkowo zdecydować, jak menu powinno wyglądać, jaką powinno mieć orientację oraz czy jakoś je urozmaicić, aby było bardziej interesujące. Jednym z popularnych rozwiązań są tzw. **przesuwane drzwi** (ang. *sliding doors*). W poniższym podrozdziale wyjaśniam, jak zrobić takie menu w WordPressie.

## PRZESUWANE DRZWI



Technika CSS przesuwanych drzwi jest prosta w implementacji, a jednocześnie umożliwia efektywne wykorzystanie grafik tła w poziomym menu bez konieczności ustawiania stałej szerokości. Ogólnie chodzi o to, aby wstawić obrazy w tle elementów menu, które są na tyle elastyczne, że mogą pomieścić dowolną ilość tekstu. Podstawę naszego menu będzie stanowił poniższy kod:

```
<ul id="navigation">
    <li><a href="menu-item-1">Pierwszy element</a></li>
    <li><a href="menu-item-2">Drugi element</a></li>
    <li><a href="menu-item-3">Trzeci element</a></li>
    <li><a href="menu-item-4">Czwarty element</a></li>
    <li><a href="menu-item-5">Piąty element!</a></li>
</ul>
```

Kod o takiej strukturze zwracają w zasadzie wszystkie tagi szablonowe WordPressa generujące listy kategorii i stron. Istotnie menu zawsze powinno się budować na bazie list nieuporządkowanych, gdy tylko jest to możliwe. Czym w końcu jest menu, jeśli nie listą?

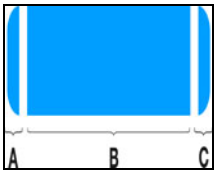
Wyobraźmy sobie, że do reprezentacji elementów menu chcemy użyć takiego przycisku o zaokrąglonych rogach, jaki widać na rysunku 11.2. Jest to typowy nowoczesny styl. Problem w tym, że gdybyśmy chcieli umieścić go w tle każdego elementu menu, elementy te musiałyby mieć stałą szerokość. A ponieważ nie chcemy tworzyć osobnego obrazu dla każdego z tych elementów, musimy wykazać się pomysłowością.

Rozwiązanie polega na podzieleniu obrazu na trzy części.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Rysunek 11.2. Obraz tła z zaokrąglonymi rogami

Pierwsza część reprezentuje zaokrąglone rogi z lewej strony, druga środek, cokolwiek na nim się znajduje, a trzecia — zaokrąglone rogi z prawej strony (rysunek 11.3).




Rysunek 11.3. Jak pociąć obraz przycisku

Środkowa część powinna zostać użyta jako tło odnośnika, a ponieważ jestem leniwy i nie chce mi się rozwijać tego tematu, użyłem jednokolorowego tła. Następnie lewą stronę obrazu należy umieścić po lewej stronie podnośników, a prawą — po prawej. Pośrodku natomiast będzie widoczne jednokolorowe tło.



Jak to zrealizować w HTML-u? To proste, wystarczy do każdego łącza dodać element `span`:

```
<ul id="navigation">
    <li><a href="menu-item-1"><span>Pierwszy element</span></a></li>
    <li><a href="menu-item-2"><span>Drugi element</span></a></li>
    <li><a href="menu-item-3"><span>Trzeci element</span></a></li>
    <li><a href="menu-item-4"><span>Czwarty element</span></a></li>
    <li><a href="menu-item-5"><span>Piąty element!</span></a></li>
</ul>
```

Teraz wystarczy dodać obrazy za pomocą CSS. Obraz reprezentujący lewą stronę należy umieścić przy lewej krawędzi łączy, które powinny mieć dodatkowo zdefiniowany odpowiedni kolor tła. Obraz reprezentujący prawą stronę należy ustawić w tle elementów `span`, tylko przysunąć go do prawej krawędzi tych elementów. W efekcie otrzymasz przyciski z zaokrąglonymi rogami, które mogą mieć dowolną szerokość.

Żeby ułatwić Ci zrozumienie, o co chodzi, poniżej przedstawiam też kod CSS. Zanim go użyjesz, będziesz pewnie musiał go dostosować do swoich potrzeb, ale chodzi tylko o to, aby pokazać, jak to działa:

```
ul#navigation li {
    float:left;
    padding: 5px;
    list-style:none; }

ul#navigation a:link, ul#navigation a:visited {
    display:block; }
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
ul#navigation a:hover, ul#navigation a:active {
    background: #888 url(corners-left.gif) no-repeat left;
    float:left; }

ul#navigation a span {
    float:left;
    display:block; }

ul#navigation a:hover span {
    float:left;
    display:block;
    background: url(corners-right.jpg) no-repeat right; }
```

Jak zmusić kod do działania z funkcją `wp_nav_menu()`, czyli tagiem szablonowym służącym do wyświetlania menu? To proste, wystarczy dodać znaczniki `<span>` i `</span>` za pomocą parametrów `before` i `after`:

```
<?php wp_nav_menu( array(
    'before' => '<span>',
    'after' => '</span>'
) ); ?>
```

## MENU ROZWIJANE



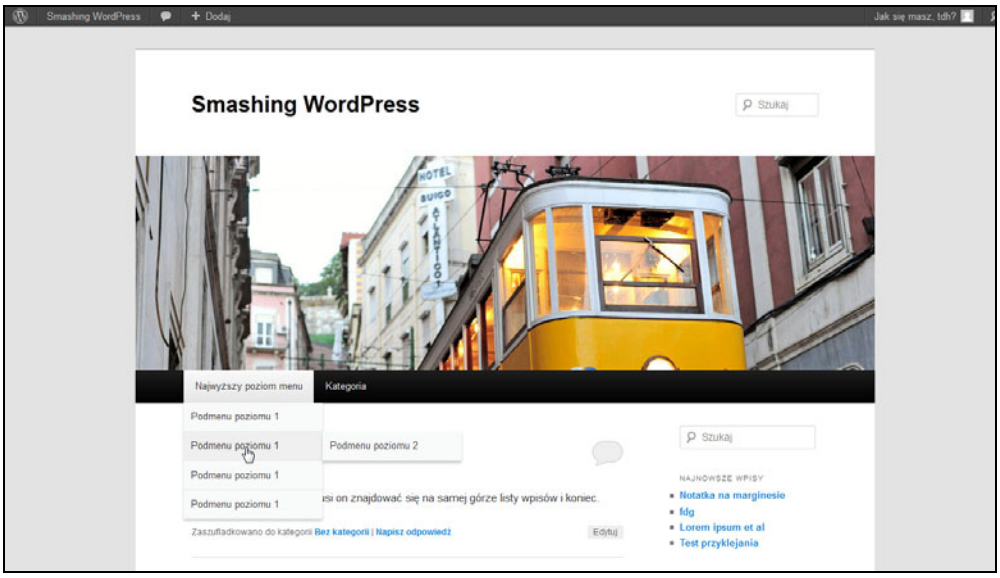
Im więcej treści zawiera witryna, tym więcej chciałoby się zmieścić także w jej menu. To oczywiste, bo menu ma przecież ułatwiać użytkownikom dotarcie do informacji, co sprawia, że kusi nas dodawanie tam coraz więcej i więcej. Popularnym rozwiązaniem są menu poziome, ale w nich można niewiele zmieścić, ponieważ ogranicza nas szerokość strony. Dlatego często stosowanym wyjściem jest użycie menu, które po najechaniu kursorem na dowolny element rozwija listę dodatkowych elementów, jak widać na rysunku 11.4. Dzięki temu użytkownicy mają łatwy dostęp do podsekcji menu.

Jednak od kiedy pojawiły się urządzenia dotykowe, takie jak smartfony i tablety, przy tworzeniu menu rozwijanych należy zachować ostrożność. Element ten jest bardzo ważną częścią strony i dlatego nie polecałbym uzależniania jego działania od efektu najechania kursorem. Użytkownik ekranu dotykowego nie ma możliwości używania kursora, a jedynie może coś nacisnąć (tzn. dotknąć na wyświetlaczu, co jest odpowiednikiem tradycyjnego kliknięcia) lub nie. Na ekranie dotykowym nie da się najechać na element kursorem, przez co menu wykorzystujące ten efekt w takich urządzeniach nie działa tak jak powinno. Obecnie w użytku znajdują się już miliony tabletów i liczba ta cały czas rośnie.

Czy w takim razie należy całkiem zrezygnować z menu rozwijanych? Zalecam ostrożność, ale moim zdaniem wciąż mogą być one przydatne. Jeśli zapewni się jakieś rozwiązania awaryjne, np. możliwość dotykowego używania elementu menu najwyższego poziomu, i umożliwi się użytkownikom dotarcie do głębszych partii witryny, to nie widzę przeszkód, aby zastosować menu rozwijane. Nawigacja po witrynie nie będzie tak wygodna i łatwa jak przy użyciu myszy, ale wszystko będzie się dało znaleźć. Pamiętaj o tych kwestiach, gdy będziesz projektować menu w swoich przyszłych projektach.

Teraz zmienimy temat i zajmiemy się wyświetlaniem reklam w pętli.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 11.4. Menu rozwijane w motywie Twenty Eleven



# WSTAWIANIE REKLAM W PĘTLI

Często jestem pytany o to, jak wstawić reklamy — zwłaszcza Google AdSense — do pętli. Najczęściej chodzi o to, aby wyświetlić reklamy za pierwszym, drugim albo trzecim wpisem, a następnie kontynuować wyświetlanie wpisów, jakby nigdy nic.

Jednym ze sposobów jest podzielenie pętli na dwie części. Pierwsza część wyświetla kilka wpisów (np. dwa), po nich wyświetlamy reklamę, a następnie druga pętla wyświetla kilka kolejnych wpisów. Dla użytkownika witryny wszystko wygląda normalnie.

Jednak lepszym sposobem jest policzenie wpisów i wyświetlenie reklamy po wyświetleniu określonej liczby wpisów. Pomysł ten można zrealizować w PHP poprzez zwiększanie zmiennej w każdej iteracji pętli wyświetlającej jeden wpis.

Poniżej znajduje się typowy kod wyświetlający reklamę za pierwszym wpisem (kod HTML jest nieistotny i został usunięty):

```
<?php
    // Pętla
    while (have_posts()) : the_post();
    // Dodaje 1 do $postcount w każdej iteracji
    $postcount++; ?>
        <!-- Wyświetla wpis -->
        <?php
            // Sprawdza wartość $postcount
            if ($postcount < 2) : ?>
            <!-- Kod reklamy -->
        <?php
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            // Koniec sprawdzania
            endif;
    // Koniec pętli
    endwhile;
?>
```

Jest to zwykła pętla z dodatkiem licznika. W każdym powtórzeniu tej pętli zmienna `$postcount` jest zwiększana o jeden. Działanie to jest wykonywane na samym początku pętli `while (have_posts()) : the_post()`. Do zwiększania wartości został użyty operator ++, który właśnie zwiększa wartość zmiennej o jeden.

Dalszy kod jest bardzo prosty. Jeśli zmienna `$postcount` ma wartość mniejszą od 2, zostanie wyświetlony zarówno wpis, jak i kod reklamy. W pozostałych przypadkach zostanie wyświetlony tylko wpis.

A jak wyświetlić reklamę pod pierwszym i drugim wpisem? Trzeba w instrukcji warunkowej `if` zmienić wartość 2 na 3. Dzięki temu kod reklamy zostanie wyświetlony pod pierwszym i drugim wpisem:

```
<?php if ( $postcount < 3 ) : ?>
```

## Uwaga na regulaminy sieci reklamowych



Wstawienie reklam między wpisami wydaje się bardzo dobrym pomysłem i rzeczywiście reklamy takie nie przeszkadzają czytelnikom, przynajmniej nie za bardzo. A jeśli nawet trochę denerwują, to i tak są o wiele lepsze od reklam wstawianych w treści, co również da się zrobić, stosując pewne sprytne sztuczki.

Jednak mimo że rozmieszczenie reklam między wpisami może wydawać się znakomitym pomysłem, trzeba dokładnie przeczytać reguły sieci reklamowej, z której się korzysta. Jedną z najpopularniejszych jest Google AdSense. W sieci tej istnieje limit trzech reklam na jednej stronie, a jeśli się go przekroczy, i tak zostaną wyświetlone tylko trzy jednostki reklamowe. Może to być problemem zwłaszcza dla osób, które żyją z wyświetlania reklam.

Dlatego mimo że polecam rozmieszczenie reklam w różnych miejscach stron (jest to o wiele lepsze od ściśnięcia ich wszystkich u góry strony), muszę jednak uczulić Cię, abyś zawsze dokładnie czytał regulaminy sieci reklamowych.

Ponadto stosując techniki automatycznego wyświetlania reklam, uważaj, aby przez przypadek nie wyświetlić ich w zbyt dużych ilościach na całej stronie. Ani Tobie, ani czytelnikom, ani tym bardziej sieci reklamowej takie coś się nie spodoba.

Ponadto dokładnie przemyśl kwestię wyświetlania reklam na stronach błędu 404. Oczywiście na tych stronach również da się wyświetlać reklamy, ale reklama kontekstowa w takim miejscu raczej się nie sprawdzi. Poza tym nie wiem, czy reklamodawcy będą chcieli płacić za wyświetlanie ich reklam na stronach błędów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# TWORZENIE POMOCNYCH STRON BŁĘDU 404

Strony błędu 404 są bardzo ważne, ponieważ są wyświetlane zawsze wtedy, gdy użytkownik próbuje wejść na stronę, która nie istnieje. Może się to zdarzyć niezależnie od Ciebie, gdy np. ktoś utworzy link do Twojej strony i zrobi błąd. Plik szablonowy obsługujący błędy 404 ma nazwę *404.php*. Można w nim zrobić, co tylko się chce, ponieważ jest to normalny szablon WordPressa.

W tym rozdziale nie opisuję szczegółowo, jak utworzyć plik szablonowy *404.php* (tworzenie plików szablonowych opisałem już w rozdziale 4.), podpowiadam natomiast, jakie warunki powinien spełniać dobry plik tego typu:

- Powinno być oczywiste, że żądana strona nie istnieje. Innymi słowy, należy wyraźnie poinformować użytkownika, że strona, którą próbuje otworzyć, nie istnieje i nie powinno być co do tego żadnych wątpliwości.
- Zaoferuj możliwość rozwiązania problemu. Ponieważ błędy 404 najczęściej zdarzają się po kliknięciu błędnego łącza, należy dostarczyć użytkownikowi jakiś inny sposób na dotarcie tam, gdzie chce. W szczególności można zaoferować wyszukiwarkę.
- Pochwal się zawartością witryny. Odwiedzający może nie być aż tak bardzo zainteresowany treścią, której poszukiwanie przywiodło go na stronę 404, aby skorzystać z wyszukiwarki. Możesz jednak spróbować go zatrzymać u siebie, wyświetlając łącza do kategorii i dodając zwięzły opis każdej z nich. Możesz zamienić stronę 404 w kącik reklamowy zawartości witryny.
- Wyświetl najnowszą treść, np. listę odnośników prowadzących do najnowszych publikacji. Może coś z tego spodoba się odwiedzającemu.
- Rozwesel odwiedzającego. Odrobina lekkiego humoru może sprawić, że użytkownik będzie mniej zły z powodu nieznalezienia tego, czego szukał.
- Zaoferuj możliwość zgłoszenia błędu. Pozwól użytkownikowi skontaktować się z Tobą w celu powiadomienia Cię o brakującej treści. Takie coś jest pozytywnie odbierane przez użytkowników, a dodatkowo pozwala wyłowić różne niedoskonałości serwisu.

Dobra strona błędów 404 dużo mówi o witrynie, dlatego nie żałuj czasu na jej dopracowanie.



# UŻYWANIE BIBLIOTEK JAVASCRIPT W WORDPRESSIE

Język JavaScript to nieocenione narzędzie, zwłaszcza gdy używa się jego licznych bibliotek, takich jak jQuery, MooTools czy Scriptaculous. Dzięki tym bibliotekom można bardzo łatwo uzyskać różne efekty, takie jak przejścia, inne wizualne bajery i wiele więcej. Większość najfajniejszych witryn internetowych działa obecnie w oparciu o JavaScript. Także panel administracyjny WordPressa nie jest zbyt przyjemny w użyciu, gdy wyłączy się obsługę JavaScriptu w przeglądarce.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

JavaScriptu można używać także na stronach WordPressa. Za pomocą tego języka można robić wszystko — od wizualnych dodatków mających uczynić strony milszymi dla oka po nowe, genialne, własnoręcznie napisane funkcje. Trzeba tylko wiedzieć, jak dołączać skrypty do WordPressa. Do tego służy funkcja `wp_enqueue_script()`. Za jej pomocą można dołączać zarówno biblioteki JavaScript standardowo dostępne z WordPressem, jak również własne skrypty.

Sposób użycia tej funkcji jest bardzo prosty. Skrypt, który chce się dołączyć, należy przypiąć do wybranego haka. Jeśli chcesz np. załadować skrypt jak najpóźniej, bo i tak będzie potrzebny, dopiero gdy użytkownik wykona jakąś czynność, można napisać funkcję używającą funkcji `wp_enqueue_script()` i podłączyć ją do haka `wp_enqueue_scripts` jak w poniższym przykładzie dołączania biblioteki jQuery:

```
function smashing_scripts() {
    wp_enqueue_script( 'jquery' );
}
add_action( 'wp_enqueue_scripts', 'smashing_scripts' );
```

Ten kod spowoduje załadowanie biblioteki jQuery w wersji dostępnej w WordPressie. Inne biblioteki, które można dołączyć w ten sposób, znajdują się w folderze *wp-includes/js/*. Dzięki funkcji `wp_enqueue_script()` ułatwione jest także ładowanie skryptów tylko wtedy, gdy są potrzebne. Trzeba tylko sprytnie posłużyć się tagami warunkowymi. Jeśli chcesz załadować skrypt tylko na stronie głównej, użyj tagu warunkowego `is_home()` itd.



Za pomocą JavaScriptu można zrobić wiele rzeczy. Wszystko zależy od rodzaju witryny i Twoich osobistych preferencji. Niektórzy wolą bibliotekę Prototype niż jQuery, inni z kolei wolą MooTools lub jeszcze coś innego. Dlatego właśnie za pomocą funkcji `wp_enqueue_script()` można załadować, co tylko się chce. Należy jej używać oraz powinno się uważać, aby nie spowalniać działania witryny dodawaniem skryptów na stronach, na których nie są potrzebne, lub wczytywaniem skryptów zbyt wcześnie. Po co w ogóle czekać na załadowanie skryptu, który nie jest wcale potrzebny?

Zachęcam Cię do przeczytania dokumentacji funkcji `wp_enqueue_script()` na stronie *codex.wordpress.org/Function_Reference/wp_enqueue_script*. Znajdziesz tam informacje na temat pozostałych bibliotek standardowo dostępnych z WordPressem, jak również dowiesz się, jak przekazywać do nich zależności oraz jak je wyrejestrowywać. Później warto też sięgnąć do źródeł na temat samych bibliotek.

## REJESTROWANIE SKRYPTÓW

W sieci dostępnych jest mnóstwo gotowych do użycia skryptów JavaScript, nie wspominając o rozszerzeniach popularnych bibliotek, o których była mowa powyżej. Poczytaj jednak, zanim użyjesz którejś z nich, nie tylko po to, aby nie dodać do swoich stron czegoś niepożądanego, ale również aby upewnić się, że są obsługiwane przez wszystkie przeglądarki. Gdy masz pewność, że użycie wybranej biblioteki nie zaszkodzi, dodatkowe funkcje lub efekty wizualne osiągnięte dzięki niej na stronach mogą znacznie uatrakcyjnić witrynę.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Kolejną bardzo przydatną funkcją jest `wp_register_script()`. Służy ona do rejestrowania skryptów, aby można było ich łatwo używać w podobny sposób do pokazanego wcześniej przykładu biblioteki jQuery. Dzięki tej funkcji można zarejestrować skrypt dostarczany z motywem, a następnie ładować ten skrypt za pomocą funkcji `wp_enqueue_script()` w taki sam sposób, jak ładuje się wszystkie inne skrypty w WordPressie. Można nawet ustawiać zależności, które zostaną załadowane przed właściwym skryptem, który rejestrujemy.

Poniższy kod rejestruje skrypt dostarczany wraz z motywem:

```
function smashing_scriptloader() {
    // Rejestracja skryptu
    wp_register_script(
         'smashing_script',
        get_template_directory_uri() . '/js/smashing.js'
    );
    // Zakolejkowanie skryptu
    wp_enqueue_script( 'smashing_script' );
}
add_action( 'wp_enqueue_scripts', 'smashing_scriptloader' );
```

Gdyby działanie skryptu zależało np. od biblioteki jQuery, ją również można by było dodać do funkcji `wp_register_script()`. Więcej na temat tej funkcji można przeczytać w dokumentacji na stronie *http://codex.wordpress.org/Function_Reference/wp_register_script*.



Można nawet wyrejestrować skrypt i zastąpić go własnym skryptem. Możliwość ta przydaje się na przykład wtedy, gdy chcemy użyć innej wersji jQuery niż dostarczona z WordPressem. Do tego celu służy funkcja `wp_deregister_script()`. Więcej na temat wyrejestrowywania skryptów można przeczytać na stronie *http://codex.wordpress.org/Function_Reference/wp_deregister_script*.

# DOSTOSOWYWANIE STYLU WORDPRESSA DO WŁASNEJ MARKI

Czasami chcielibyśmy, aby WordPress był wizualnie bardziej spójny ze stylem reprezentowanym przez naszą markę. Najczęściej dotyczy to witryn, w których używane są wtyczki lub inne elementy umożliwiające użytkownikom rejestrację i używanie panelu administracyjnego.

Zmienić trzeba dwie rzeczy. Po pierwsze, aby użytkownik mógł się zalogować, musi skorzystać z formularza logowania widocznego na rysunku 11.5. Jak widać, ani trochę nie przypomina on stylem witryny, którą tworzysz, a więc koniecznie trzeba go zmienić. Druga rzecz to panel administracyjny. Standardowo wygląda on jak typowy WordPress i stylem w najmniejszym stopniu nie przypomina naszej „supermarki”. Kto by tak chciał?

Niestety dla administracyjnych elementów nie ma takich szablonów jak dla stron wyświetlających treść, ale co nieco można jednak zmienić. Wszystko zależy od tego, jak bardzo zależy Ci na zmianie domyślnego wyglądu WordPressa i jak daleko chcesz się w tych modyfikacjach posunąć.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Rysunek 11.5. Domyślny formularz logowania

## WŁASNY FORMULARZ LOGOWANIA



Jedną z najczęściej zmienianych części WordPressa od strony administracyjnej jest formularz logowania. Logo i napis WordPress rzucają się w oczy, co jest irytujące w szczególności w przypadku tych witryn, które nie mają nic wspólnego z typowym blogowaniem.

Oczywiście formularz logowania można po prostu zmodyfikować bezpośrednio w plikach WordPressa, ale w ten sposób można coś popsuć, a poza tym zmiany trzeba by było wprowadzać od nowa po każdej aktualizacji systemu. O wiele lepszym rozwiązaniem jest użycie wtyczki. Poszukaj propozycji w dodatku A. Wymieniłem tam kilka dobrych rozwiązań i warto je wypróbować.

Często formularz logowania dostosowuje się wyglądem do motywu, bo przecież tak naprawdę jest to część motywu. W takim przypadku kod dostosowujący powinien być przenoszony wraz z plikami motywu. Na szczęście można łatwo ten problem rozwiązać, podłączając się do haka `login_head` i stosując dodatkowy arkusz stylów. W ten sposób formularz logowania można dostosować do swoich potrzeb w dowolny sposób, pod warunkiem że nie chce się zmienić układu elementów. Zmiana koloru tła i wstawienie własnego logo nie sprawią problemu. Wystarczy do tego napisać nieskomplikowane reguły CSS.

Najpierw jednak trzeba napisać funkcję, którą podłączy się do haka `login_head`. Techniki używania haków zostały opisane w rozdziale 6. i powinny już być Ci znane. Pamiętaj, że motyw powinien być tak skonstruowany, aby na jego bazie można było łatwo tworzyć motywy potomne, których właściciele również mogą chcieć ustawić własne tło i logo na stronie logowania. Dlatego właśnie w poniższym przykładzie użyłem funkcji `get_stylesheet_directory_uri()`, a nie `get_template_directory_uri()`:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
// Własny arkusz stylów dla strony logowania
function smashing_login() {
    $smashingLoginUrl = get_stylesheet_directory_uri() . '/login.css';
    wp_register_style( 'smashingLoginStyle', $smashingLoginUrl );
    wp_enqueue_style( 'smashingLoginStyle' );
}
add_action( 'login_head', 'smashing_login', 1 );
```

Ten prosty kod dodaje zawartość funkcji `smashing_login()` do haka `login_head`. W treści tej oczywiście jest użyta funkcja `wp_enqueue_style()` (działająca podobnie do funkcji `wp_enqueue_script()`) zawierająca informację o arkuszu stylów o nazwie *login.css*. Więcej na temat funkcji `wp_enqueue_style()` można przeczytać na stronie *http://codex.wordpress.org/Function_Reference/wp_enqueue_style*.

Tylko od Ciebie zależy, co zmienisz. Wystarczy, że wpiszesz odpowiednie reguły CSS w nowym arkuszu stylów (w pliku *login.css*). Możesz np. zmienić kolor tła, odwołując się do elementu `body`, ustawić własne logo, które znajduje się w elemencie `h1`, a także zmodyfikować sam formularz logowania, który znajduje się w kontenerze `div#login`. Przyjemnej zabawy!

---

*Jeśli chcesz zmienić tylko logo, to możesz użyć jednej z dostępnych wtyczek, zamiast pisać kod samodzielnie. Ale wtedy niczego nowego się nie nauczysz, prawda?*

---



## MOTYWY PANELU ADMINISTRACYJNEGO

Modyfikowanie motywu panelu administracyjnego jest raczej trudne. Przede wszystkim jedynym sposobem na zrobienie tego bez grzebania w rdzennych plikach jest przesłonięcie tych plików (czego nie należy robić, jeśli nie tworzy się całkiem osobnej gałęzi WordPressa) za pomocą wtyczek. Poniżej przedstawiam kod takiej wtyczki:

```
<?php
/*
Plugin Name: Smashing Admin Colors
Plugin URI: http://tdh.me/wordpress/smashing-admin/
Description: Dodaje własny schemat kolorów dla panelu administracyjnego.
Version: 1.0
Author: Thord Daniel Hedengren
Author URI: http://tdh.me/
*/
function smashing_admin_colors() {
    $file = dirname(__FILE__) . '/smashing-admin';
    wp_admin_css_color(
         'smashingadmin',
         'Smashing Admin',
        plugin_dir_url( $file ) . 'smashing-admin.css',
        array( '#ff4720', '#41b7d8', '#333333', '#a6a6a6' )
);
}
add_action( 'admin_init', 'smashing_admin_colors' );
?>
```

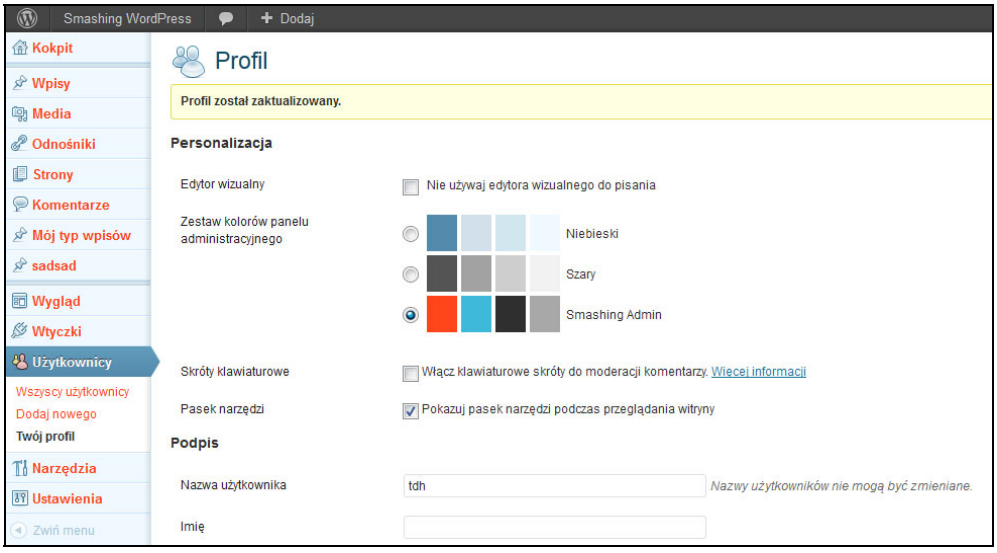
ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Do haka `admin_init` została dodana funkcja `smashing_admin_colors()`. Dzięki temu funkcja `wp_admin_css_color()` rejestruje nowy schemat kolorów o nazwie *Smashing Admin* i pobiera plik *smashing-admin.css* z folderu wtyczki. Tablica znajdująca się na końcu funkcji `wp_admin_css_color()` reprezentuje kolory schematu dostępnego w profilu użytkownika w panelu administracyjnym.

Oczywiście w pliku *smashing-admin.css* powinien znajdować się jakiś kod CSS. W pliku tym można wpisać całkiem nowe reguły, jak również użyć zmodyfikowanego istniejącego arkusza. Poniższy przykład importuje arkusz stylów klasycznego schematu kolorów i zmienia definicje kolorów niektórych łączy. Nie jest to nic oszałamiającego, ale chodzi tylko o to, aby pokazać, jakie są możliwości. Rezultat można obejrzeć po wybraniu tego schematu kolorów w panelu administracyjnym (rysunek 11.6).

```
@import url(../../../wp-admin/css/colors-classic.css);
.form-table td, .form-table th {
    border-bottom-color: #ffffff; }
#wphead, #footer, #footer-upgrade {
    color: #333; }
#adminmenu a:link, #adminmenu a:visited {
    color:#ff4720; }
#adminmenu a:hover {
    color: #41b7d8; }
```






Rysunek 11.6. Kolorystyka Smashing Admin

Można znaleźć kilka naprawdę robiących duże wrażenie motywów panelu administracyjnego, ale nie miej złudzeń — dostosowanie tego panelu do własnych potrzeb praktycznie zawsze wymaga dużo pracy. Chociaż czasami potrzebujemy jedynie dokonać drobnych modyfikacji, jak zmiana kolorów itp. W takich przypadkach nie powinno być żadnych problemów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# DOPIESZCZANIE WITRYNY

Znajomość sekretów WordPressa pozwala na utworzenie witryny będącej czymś więcej niż kolejnym zwykłym (choćby nawet najświetniejszym) motywem. Stosując opisane w tym rozdziale techniki, można udoskonalić każdą witrynę, a więc warto je wypróbować i trochę się nimi pobawić.

Podczas gdy precyzyjnie dostosowany styl wpisów i poprawnie skonstruowane menu to bardzo ważne czynniki wpływające na jakość witryny, możliwości udoskonalania jest o wiele więcej. Wiele porad na ten temat znajdziesz w doskonałych serwisach Smashing Magazine (*www.smashingmagazine.com*) i A List Apart (*www.alistapart.com*). Niektóre z opisanych rozwiązań można zastosować od razu, nad innymi trzeba trochę popracować. Popuść wodze fantazji!



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 12 ZABAWA Z MEDIAMI

**MÓWI SIĘ, ŻE JEDEN OBRAZ** wart jest tysiąca słów. Za pomocą obrazu można przekazać wiadomość o wiele szybciej niż przy użyciu tekstu. To samo dotyczy innych mediów wizualnych, takich jak filmy czy animacje. Są to środki pozwalające zilustrować przekazywaną myśl i mogą występować jako uzupełnienie tekstu lub nawet być samodzielnymi przekaźnikami informacji.

Jeśli używasz elementów multimedialnych w swojej witrynie, to musisz zastanowić się nad sposobem publikowania takich elementów jak galerie czy ilustracje. Tematem tego rozdziału jest publikowanie elementów multimedialnych w WordPressie, ale w szerszym rozumieniu niż tylko dodawanie ilustracji do wpisów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# TWORZENIE GALERII OBRAZÓW

W WordPressie już od dawna dostępny jest skrót kodowy `[gallery]` do tworzenia galerii.

---

*Jeśli używasz edytora wizualnego, to nie zobaczysz przycisku galerii. Zamiast tego będzie widoczne duże pole z informacją, że jest to obszar obrazu. Możesz przełączyć się na tryb HTML, aby zobaczyć kod.*

---

Skrót `[gallery]` powoduje wyświetlenie wysłanych na serwer obrazów w postaci siatki miniatur, które można klikać. Gdy użytkownik kliknie miniaturę, może zostać wyświetlony obraz w większym rozmiarze w obrębie strony motywu albo całkiem osobno. Pierwsza z wymienionych opcji to tzw. **strona załącznika**, ponieważ obrazy są załącznikami do wpisów. Jeśli prowadzisz witrynę, w której publikujesz głównie tekst i od czasu do czasu jakąś grafikę, to ten skrót kodowy powinien Ci w zupełności wystarczyć.

Jeśli chcesz dostosować sposób wyświetlania tych galerii, to pierwsze, co powinieneś zrobić po zainstalowaniu WordPressa i wybraniu motywu, to skierować się na stronę ustawień mediów w panelu administracyjnym. Na stronie tej można określić warunki skalowania obrazów. Każdy obraz wysłany do WordPressa jest zapisywany w maksymalnie czterech wersjach, z których każda służy do czego innego.



Na rysunku 12.1 pokazana jest strona ustawień mediów WordPressa, na której zdefiniowane są rozmiary obrazów:

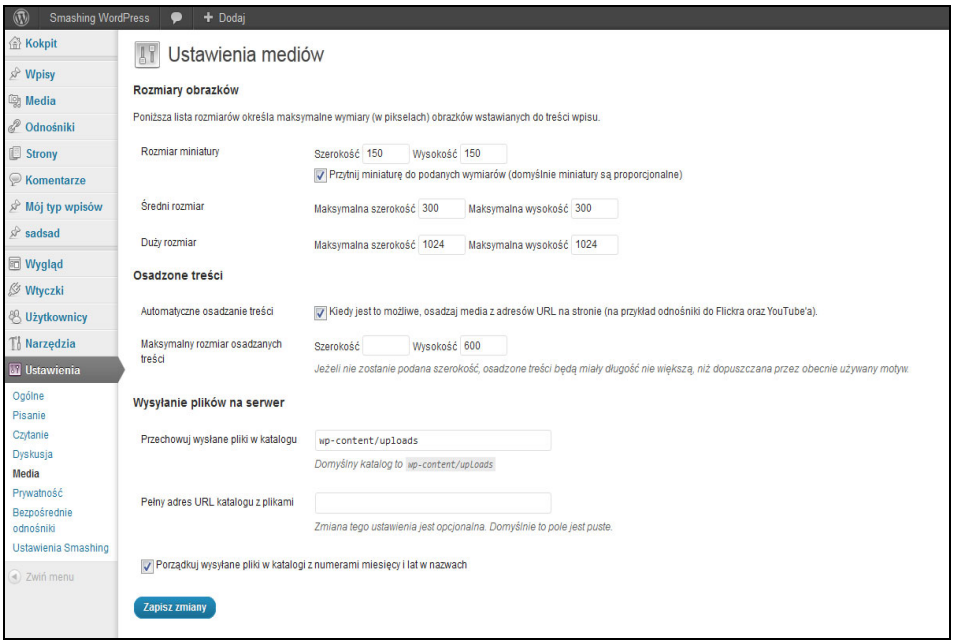
- Miniatura — obcięta wersja obrazu mająca pełnić rolę symbolu oryginału. Można wybrać obcinanie do konkretnych wymiarów, przez co widoczny będzie tylko fragment oryginalnej grafiki. Jest to niewielki obraz mający z założenia służyć jako łącze. Domyślne wymiary to 150×150, ale można je zmienić.
- Średni rozmiar — pomniejszona wersja oryginału z zachowaniem proporcji. Można określić maksymalną szerokość i wysokość, aby mieć kontrolę nad skalowaniem obrazu. Ten rozmiar jest także używany na stronach załączników.
- Duży rozmiar — także proporcjonalnie zmniejszona wersja oryginalnego obrazu.
- Pełny rozmiar — ta opcja dostępna jest w oknie wysyłania pliku na serwer i oznacza obraz w oryginalnym rozmiarze.

Drobna uwaga: WordPress nie tworzy wersji, które są większe od oryginału. Jeśli więc w ustawieniach masz zapisane, że duży obraz ma wysokość i szerokość 800 pikseli i wyślesz plik o mniejszych wymiarach, wersja o tych wymiarach nie zostania utworzona i nie będzie dostępna do wyboru w WordPressie. Po prostu nie ma sensu jej tworzyć.

Po co w takim razie ustawiać te opcje?

To proste. Szerokość miniatur powinna odpowiadać szerokości i liczbie kolumn, jakie chcesz wyświetlać w galeriach tworzonych za pomocą skrótu `[gallery]`. Ustaw taką wartość, która zapewni atrakcyjny efekt w Twojej witrynie.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974



Rysunek 12.1. Strona ustawień mediów WordPressa



Ponadto uważam, że średni rozmiar obrazu powinien być równy maksymalnemu dopuszczalnemu w projekcie. Innymi słowy, jeśli maksymalna szerokość wynosi 580 pikseli, to maksymalną szerokość medium należy ustawić również na 580 pikseli — aby mieć pewność, że każdy obraz można wstawić w dowolnym miejscu w każdym wpisie. Jako że średnie rozmiary obrazów są domyślnie używane na stronach załączników, warto zadbać o to, aby dobrze tam pasowały. Oczywiście równie dobrze możesz na tych stronach wyświetlać duże obrazy, ale mogą być Ci one potrzebne do innych celów — np. do prezentowania w sposób nieograniczony szerokości strony. Niezależnie od tego, jak ustawisz rozmiary obrazów i jaki rozmiar będziesz wyświetlać na stronach załączników, ważne jest, aby obrazy te dobrze wyglądały i spełniały swoje zadanie.

Jak wspomniałem wcześniej, duży rozmiar można wyświetlać w wyniku kliknięcia miniatury. W zasadzie duże obrazy można wyświetlać też na stronach załączników, wykorzystując do tego celu np. szablon *attachment.php*, ale domyślnie tak nie jest. Duży obraz jest interesujący tylko wówczas, gdy wysyłamy na serwer zdjęcie o wysokiej rozdzielczości i nie chcemy tworzyć odnośnika do oryginalnej wersji zajmującej 15 MB, która jest o wiele za duża do przeglądania w internecie. Duży obraz jest mniejszy od oryginału i lepiej dopasowany do ekranu.

# STYLIZOWANIE GALERII

Sprawienie, aby galeria tworzona przy użyciu skrótu `[gallery]` dobrze wyglądała, nie jest trudne, w szczególności jeśli ustawi się szerokości miniatur odpowiednie do liczby planowanych kolumn. Wartość tę można ustawić podczas dodawania galerii, tylko trzeba o tym pamiętać.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Stylizowanie galerii również jest bardzo łatwe. Cała galeria mieści się w elemencie `div` z przypisanymi kilkoma pozwalającymi go zidentyfikować klasami, z których jedna to `gallery`. Jak to zwykle bywa w WordPressie, do dyspozycji mamy też kilka identyfikatorów i klas, ale najczęściej w arkuszu stylów wystarcza tylko selektor `div.gallery`.

Każdy element galerii znajduje się w elemencie `dl.gallery-item`, który z kolei zawiera element `dt.gallery-icon` zawierający miniaturę w odnośniku, i ewentualnie element `dd.gallery-caption`, jeśli dla danego obrazu został zdefiniowany podpis. Elementy są układane w rzędzie, po określonej liczbie obrazów następuje złamanie wiersza za pomocą elementu `br` z atrybutem `style="clear: both"` i rozpoczyna się nowy wiersz itd.

Mówiąc krótko, dla listy miniatur potrzebne są następujące style:

```
div.gallery {}
dl.gallery-item {}
dt.gallery-icon {}
dd.gallery-caption {}
```

Poniżej znajduje się kod z motywu Notes Blog. Zdefiniowano w nim delikatny kolor i niewielki rozmiar pisma dla podpisów oraz utworzono niewielkie odstępy między elementami, aby uatrakcyjnić wygląd galerii. Oczywiście jeśli chcesz, możesz zaprojektować coś bardziej wyszukanego, dodać tła, obramowanie itd.:



```
div.gallery {
    margin-bottom: 14px; }

img.attachment-thumbnail {
    border:0; }

dd.gallery-caption {
    margin-top: 8px;
    font-size: 12px;
    color: #777;
    font-style: italic; }
```

To jednak dopiero połowa pracy. Wprawdzie można sprawić, aby miniatury były odnośnikami bezpośrednio do obrazów, ale często stosowanym rozwiązaniem jest wyświetlanie grafik na stronach załączników, czyli w obrębie stron projektu. Najczęściej stronami tymi są *single.php* albo *index.php*, ponieważ w większości motywów brakuje szablonu *attachment.php*, a już w ogóle mało który ma pliki *video.php*, *image.php* itd.

Aby motyw był przyjazny dla galerii, na stronie pojedynczego obrazu powinno znajdować się łącze prowadzące z powrotem do galerii oraz powinny znajdować się przyciski *Dalej* i *Wstecz* pozwalające przeglądać obrazy po kolei. Do wyświetlenia prostych łączy nawigacyjnych ze strzałkami oraz tytułami poprzedniego i następnego wpisu w tytułach można użyć funkcji `previous_post_link()` i `next_post_link()`. W tagach tych jest możliwość ograniczenia przeglądania do jednej kategorii, jak również zmodyfikowania odnośnika:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<?php previous_post_link(); ?>
<?php next_post_link(); ?>
```

Więcej na ich temat można przeczytać w dokumentacji na stronach *http://codex.wordpress.org/Function_Reference/previous_post_link* i *http://codex.wordpress.org/Template_Tags/next_post_link*.

Jeśli chcesz mieć trochę większą kontrolę nad nawigacją, możesz użyć poniższego kodu, który pobiera wpis nadrzędny (wpis, z którym związany jest dany załącznik) i tworzy do niego odnośnik. Nie ma tu nic nadzwyczajnego. Oczywiście nie możemy użyć funkcji `the_permalink()`, ponieważ wskazywałaby ona sam załącznik, natomiast funkcja `the_title()` zwraca tytuł obrazu. Tytuł jednak również zazwyczaj powinien być wyświetlany:

```
<p class="attachmentnav">
    &larr; Powrót do
    <a href="<?php echo get_permalink( $post->post_parent ) ?>" title="<?php echo
    get_the_title( $post->post_parent ) ?>" rev="attachment">
        <?php echo get_the_title( $post->post_parent ) ?>
    </a>
</p>
```

A skoro mowa o zwracaniu danych przez funkcje — opis, który można wpisać przy wysyłaniu obrazu na serwer, jest zwracany przez funkcję `the_content()`, co oznacza, że obsługę wyświetlania załączników można łatwo zdefiniować w pliku *single.php* lub *index.php*. Wystarczy tylko użyć tagu warunkowego `is_attachment()` do sprawdzenia, jaki rodzaj strony jest wyświetlany. Jeśli nie chcesz używać tagu warunkowego, możesz zawsze użyć specjalnego szablonu, np. *image.php* albo *attachment.php*.



Wracając do tematu, teraz utworzymy odnośniki pozwalające wyświetlić poprzedni i następny obraz w galerii. Do tego celu posłużą nam funkcje `previous_image_link()` i `next_image_link()`. Obie domyślnie wyświetlają miniaturę będącą łączem do docelowego obrazu. Poniższe łącza zostały rozmieszczone przy lewej i prawej krawędzi za pomocą własności CSS `float`:

```
<div class="alignleft">
    <?php next_image_link(); ?>
</div>
<div class="alignright">
    <?php previous_image_link(); ?>
</div>
```

Mimo że miniatura, jeśli zostanie dobrze wystylizowana, może być bardzo atrakcyjna, czasami wolimy jednak łącza tekstowe. Jeśli tak jest w Twoim przypadku, to w parametrach rozmiaru tych funkcji przekaż puste łańcuchy (tak, funkcje te pobierają także inne rozmiary). Wówczas zostanie wyświetlony tytuł wpisu obrazu, taki jaki mu nadałeś. Można też przekazać po prostu dodatkowy parametr zawierający tekst odnośnika:

```
<div class="alignleft">
    <?php next_image_link( '', 'Następny obraz' ); ?>
</div>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

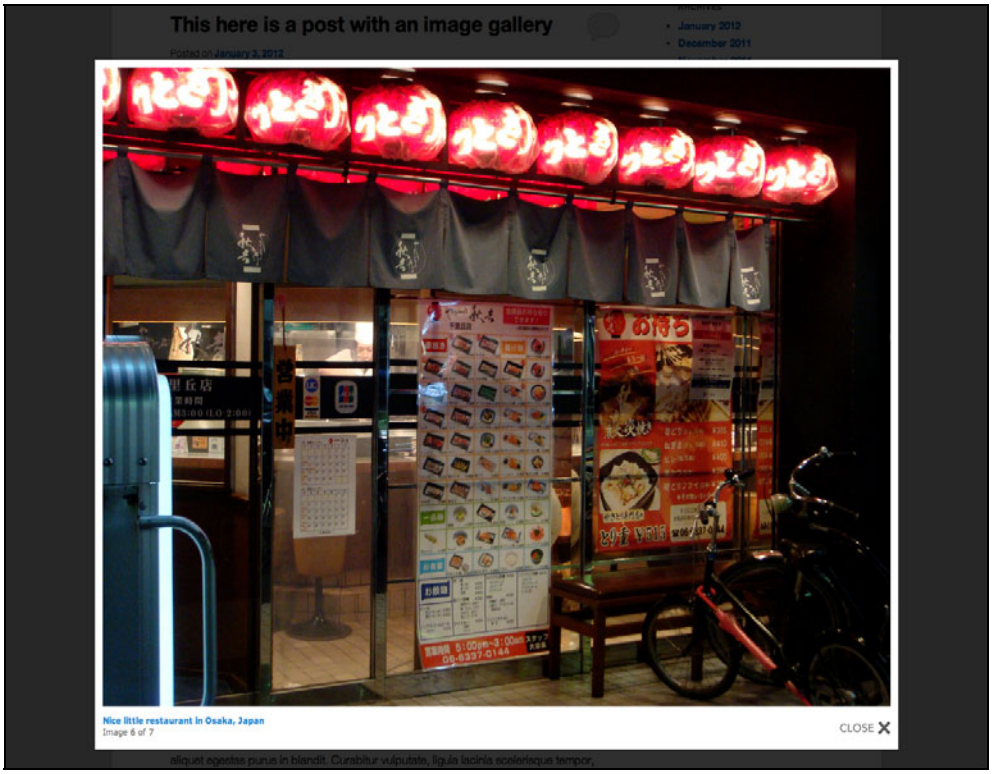
```
<div class="alignright">
    <?php previous_image_link( '', 'Poprzedni obraz' ); ?>
</div>
```

To w zasadzie wszystko, co trzeba wiedzieć na temat galerii w WordPressie. Kolejnym krokiem po dostosowaniu galerii do własnego motywu jest jej podrasowanie za pomocą wtyczek.

## LEPSZE PRZEGLĄDANIE W LEKKICH OKIENKACH

Popularnym sposobem prezentacji obrazów w witrynach internetowych jest wyświetlanie ich na nakładkach na stronę zwanych po angielsku *lightbox* (rysunek 12.2). Aby wrócić do przeglądania treści witryny, trzeba zamknąć nakładkę z obrazem, co może się wydawać niezbyt dobrym rozwiązaniem, ale kiedy porówna się to z koniecznością otwierania nowej strony, aby obejrzeć obraz w pełnych wymiarach, można zmienić zdanie. Większość dobrze zaprojektowanych tego typu rozwiązań dodaje także przyjazne dla użytkownika przyciski do przeglądania grafik.






Rysunek 12.2. Obraz wyświetlony na lekkiej nakładce (lightbox)

Do implementacji tego rozwiązania używa się JavaScriptu i stosuje pewne sztuczki projektowe, ale sposobów realizacji jest naprawdę dużo. Wybór jednego z nich zależy od tego, czego potrzebujesz w przypadku konkretnej witryny. Moim zdaniem najlepiej jest wybrać jakąś wtyczkę do WordPressa, ponieważ dzięki temu nie trzeba będzie dodawać własnoręcznie

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

żadnych klas do obrazów, aby umożliwić ich rozpoznanie przez skrypt. Opis kilku znakomitych wtyczek znajduje się w dodatku A, więc jeśli nie masz jeszcze ulubionej, możesz tam poszukać czegoś dla siebie. Wtyczka wszystko za Ciebie zrobi i nie będzie trzeba otwierać żadnych stron załączników. Użytkownicy będą mogli wygodnie przeglądać Twoje galerie fotografii.

Oczywiście żadne rozwiązanie nie jest idealne. Bo co się stanie, gdy użytkownik będzie miał wyłączoną obsługę JavaScriptu w swojej przeglądarce albo kliknie miniaturę, zanim skrypt zostanie w całości załadowany? Wówczas zostanie wyświetlony cały obraz poza projektem, tak jakby łącze prowadziło bezpośrednio do niego (co zazwyczaj jest prawdą, ponieważ skrypt uruchamia różne efekty dopiero wtedy, gdy zostanie wczytany). Nie jest to zbyt przyjemne, zwłaszcza dla użytkownika, który spodziewa się ładnej nakładki i wygodnych przycisków do przeglądania następnych i poprzednich obrazów. Ale przecież przeglądarki mają przycisk *Wstecz*, więc nic strasznego się nie stanie.

Kiedy nie powinno się używać nakładek? Jednym z przeciwwskazań są małe urządzenia. Czy taka galeria dobrze prezentowałaby się na 7-calowym ekranie? Czy byłaby użyteczna? To samo można powiedzieć o stronach załączników, ale je łatwiej jest dostosować do potrzeb różnych urządzeń. Należy zadbać o to, aby skrypt nakładki nie powodował problemów w takich sytuacjach.

W końcu jeśli zarabiasz na wyświetleniach stron, nie używaj nakładek, chyba że wiesz, że dzięki nim zdobędziesz więcej czytelników. W każdym razie jeśli użytkownicy chcący obejrzeć każdy kolejny obraz będą musieli załadować nową stronę i przy okazji nowy zestaw reklam, to nie można chcieć więcej.



## GALERIE SPOZA WORDPRESSA

Na koniec jeszcze kilka słów o rozwiązaniach do tworzenia galerii niezależnych od WordPressa. Do wyboru jest kilka skryptów. Niektóre z nich to w istocie kompletne systemy same w sobie, a inne to tylko programy skalujące obrazy i zapisujące wyniki tego skalowania w folderach w postaci plików HTML.

Zalecam zastanowić się dwa razy, zanim użyje się zewnętrznej galerii, takiej jak np. Gallery2. Jedną z największych wad takiego rozwiązania jest brak elastyczności. WordPressa można rozszerzać za pomocą rozmaitych wtyczek i jeśli grafiki należą do systemu, można z nimi robić różne rzeczy. Jeśli jednak obrazy umieścisz w zewnętrznym systemie i tylko wyświetlisz je w WordPressie w jakiś sposób, to stracisz wiele możliwości. Ponadto co zrobisz, gdy ten zewnętrzny skrypt nagle przestanie działać albo zacznie kolidować z WordPressem?

To samo tak naprawdę dotyczy wtyczek przenoszących funkcjonalność galerii z rdzenia WordPressa do własnych skryptów. Dzięki temu możliwe jest zaimplementowanie lepszych funkcji, lepszego sortowania itd., ale jeśli wtyczka taka zostanie porzucona albo z jakiegoś powodu przestanie działać, będziesz zdany wyłącznie na siebie. Poza tym czemu by nie poszukać wtyczki, która rozszerza funkcjonalność rdzenia WordPressa, zamiast działać prawie jak samodzielne rozwiązanie?

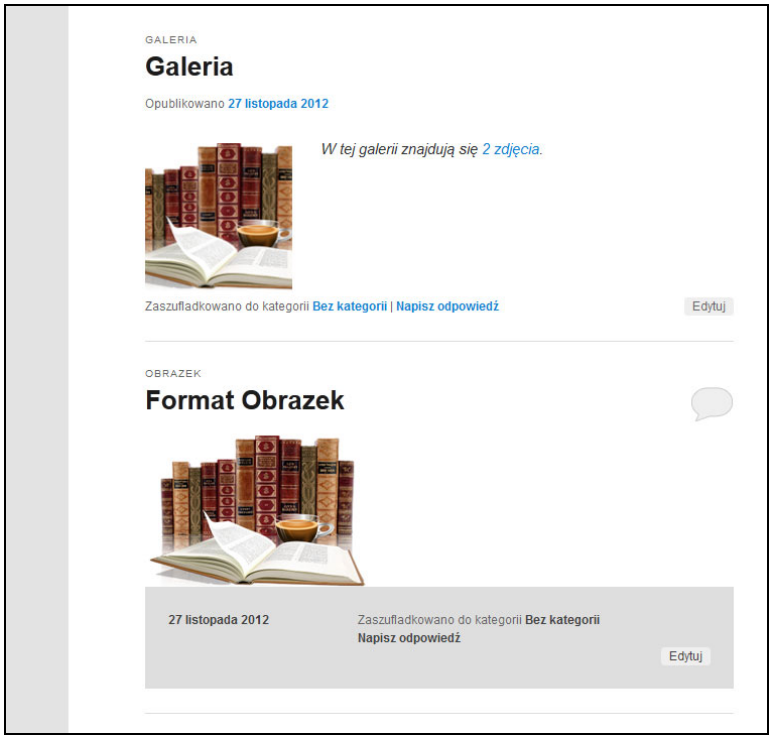
ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Mimo wszystko czasami po prostu potrzebne jest bardziej rozbudowane rozwiązanie i nie da się uniknąć wyjścia poza rdzeń WordPressa czy to za pomocą wtyczki, czy korzystając z zewnętrznego systemu. W takim przypadku upewnij się tylko, czy wiesz, co robisz i czy będziesz wiedział, jak rozwiązać ewentualne trudności. Twoje samopoczucie będzie na pewno lepsze, jeśli będziesz wiedział, że w każdej chwili możesz użyć jakichś narzędzi do konwersji lub importu danych.

# FORMATY WPISÓW

Formaty wpisów dają nam większą kontrolę nad treścią wpisów, o czym była już mowa wcześniej. Dzięki nim można dostosować wpisy do rodzaju zapisanej w nich treści, co jest szczególnie przydatne w przypadku treści multimedialnych. Gdy jest mowa o mediach, na myśl przychodzą cztery formaty wpisów: *Obrazek*, *Galeria*, *Film* i *Plik dźwiękowy*. Na rysunku 12.3 pokazano, w jaki sposób można wizualnie odróżnić od siebie pojedyncze obrazy i galerie.






Rysunek 12.3. Formaty wpisów Obrazek i Galeria w motywie Twenty Eleven

Formaty wpisów zostały opisane wcześniej, a więc już wiesz, że możesz je formatować i nimi zarządzać w dowolny sposób. Dlatego też myślę, że możemy przejść od razu do osadzania treści. Jeśli chcesz sobie odświeżyć wiadomości na temat formatów wpisów, przeczytaj jeszcze raz rozdział 6.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

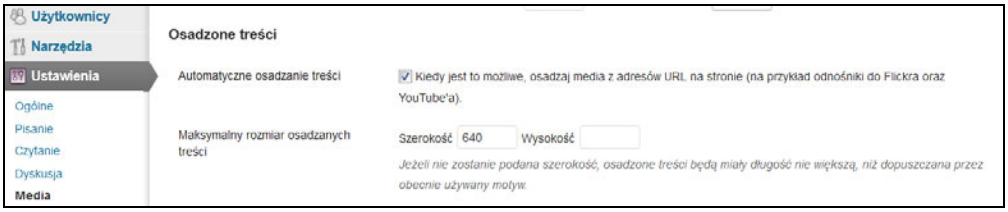
# OSADZANIE TREŚCI MULTIMEDIALNEJ NA STRONACH

Jeśli na swoich stronach wykorzystujesz filmy, to najprawdopodobniej korzystasz z usług takich serwisów jak YouTube albo Vimeo. Publikowanie treści z tych portali jest bardzo proste. Wystarczy skopiować specjalny kod i wkleić go we wpisie w swojej witrynie. Ale czy wiesz, że wcale nie musisz bawić się z wewnętrznymi ramkami, aby osadzać te treści? WordPress może te treści osadzić za Ciebie i automatycznie dostosować ich wymiary do Twoich ustawień dla mediów.

## KONFIGUROWANIE USTAWIEŃ

Zacznij od sprawdzenia, czy wszystkie ustawienia mediów masz poprawnie skonfigurowane. Wejdź na stronę ustawień mediów i znajdź sekcję *Osadzone treści*. Pole wyboru *Kiedy jest to możliwe, osadzaj media z adresów URL na stronie (na przykład odnośniki do Flickra oraz YouTube'a).* powinno być zaznaczone. Dzięki temu WordPress będzie próbował osadzać treść bezpośrednio. Dodatkowo najprawdopodobniej zechcesz zmienić ustawienia rozmiaru mediów osadzonych. Ustawienia te działają podobnie do ustawień obrazów, tzn. WordPress proporcjonalnie dostosuje szerokość do wysokości lub odwrotnie. W większości przypadków ważna jest tylko szerokość, a więc ustaw ją na taką wartość, jaka Ci odpowiada (np. 640 pikseli), a pole wysokości pozostaw puste, jak pokazano na rysunku 12.4.






Rysunek 12.4. Ustawienia treści osadzonych

Dzięki tym ustawieniom osadzone treści będą miały odpowiednie rozmiary.

Samo osadzanie treści jest łatwe. Wystarczy wkleić adres URL wybranego zasobu, a WordPress spróbuje go pobrać. Jeśli np. chcesz osadzić film Nyan Cat dostępny w serwisie YouTub pod adresem *http://www.youtube.com/watch?v=QH2-TGUlwu4*, po prostu wklej ten adres w osobnym wierszu tekstu i zapisz wpis. WordPress pobierze film i osadzi go na stronie. W ten sposób można też używać skróconych adresów YouTube (w tym przypadku *http://youtu.be/QH2-TGUlwu4*).

Istnieje też skrót kodowy `[embed]` służący do osadzania treści. Używanie go polega na wpisaniu adresu URL zasobu między znacznikami `[embed]` i `[/embed]`:

```
[embed]http://www.flickr.com/photos/tdhedengren/4333551150/sizes/o/in/
➥set-72157623327077126/[/embed]
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Za pomocą skrótu `[embed]` można też określać wymiary osadzanej treści. Na przykład poniższy kod zmienia szerokość obrazu na 250 pikseli. Pamiętaj jednak, że obraz jest ładowany w całości i skalowany przez HTML. W rzeczywistości jego rozmiar pozostaje oryginalny.

```
[embed width="250"]http://www.flickr.com/
➥photos/tdhedengren/4333551150/sizes/o/in/set-72157623327077126/[/embed]
```

### MAGICZNA TECHNIKA OEMBED

Automatyczne osadzanie treści w WordPressie działa dzięki technice zwanej **oEmbed**. Umożliwia ona pobieranie treści z witryn, które również ją obsługują, a więc np. YouTube i Vimeo. Niestety nie oznacza to, że można osadzać w ten sposób dowolną treść z dowolnej witryny ani nawet z dowolnej witryny obsługującej oEmbed. WordPress musi dodatkowo obsługiwać określonego dostawcę. Co jakiś czas dodawana jest obsługa kolejnych serwisów, ale jeśli potrzebujesz czegoś od zaraz, to możesz dodać taką obsługę własnoręcznie. Są nawet specjalne wtyczki, które to robią, np. Twitter Embed (*http://wordpress.org/extend/plugins/twitter-embed*) pozwala na osadzanie wpisów z Twittera.

Nie będę tutaj zgłębiał tego tematu, ale jeśli chcesz samodzielnie dodać do swojego WordPressa obsługę jakiejś witryny obsługującej oEmbed, to informacje, jak to zrobić, znajdziesz na stronie *http://oembed.com*. Gdy już dokładniej zorientujesz się, czym jest format oEmbed, poczytaj o dodawaniu własnych dostawców przy użyciu funkcji `wp_oembed_add_provider()` (*http://codex.wordpress.org/Function_Reference/wp_oembed_add_provider*).



## WYŚWIETLANIE LOSOWYCH OBRAZÓW

Wyświetlanie losowych obrazów i innych elementów na stronach podnosi atrakcyjność tych stron i sprawia, że stają się one bardziej żywe. Wśród blogerów popularne jest stosowanie zmiennego obrazu w nagłówku i promowanie losowych wpisów. Ogólnie rzecz biorąc, elementy, które między wizytami trochę się zmieniają (ale nie na tyle, żeby użytkownik nie mógł się połapać), są uważane za bardziej interesujące niż statyczne prezentacje.

Najprostszą techniką losowania jest użycie jednej z funkcji losowych PHP, np. `rand()` albo `mt_rand()`. Istnieją też rozwiązania tego typu w JavaScripcie. W internecie można znaleźć wiele gotowych skryptów, a więc jeśli nie musisz koniecznie pisać własnego programu od podstaw, lepiej jest skorzystać z gotowego rozwiązania.

To jednak nie jest jedyna możliwość. Istnieją też specjalne wtyczki pozwalające wyświetlać losowe elementy. Kilka z nich opisałem w dodatku A. Losowe wyświetlanie treści, a zwłaszcza obrazów, to tak często używana funkcja, że jej zastosowanie w kolejnej witrynie jest wręcz śmiesznie łatwe. Chyba że chcesz wyświetlać losowe obrazy z wpisów. Wtedy jest już trochę trudniej.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## WYŚWIETLANIE LOSOWYCH OBRAZÓW Z GALERII

Bardzo fajny efekt można uzyskać, wyświetlając losowe obrazy z galerii, czyli zbiorów grafik wysłanych do WordPressa. Obrazy te są załącznikami i można się do nich dostać przy użyciu funkcji `get_post()` w dodatkowej pętli. Chodzi o to, aby wyświetlić miniaturę dowolnego załącznika i wykonać łącze do tego obrazu. Ponieważ Twoja galeria będzie właściwie sformatowana (zgodnie z tym, czego nauczyłeś się w podrozdziale „Stylizowanie galerii"), użytkownik na pewno zechce obejrzeć więcej zdjęć, a więc wydaje się, że jest to całkiem dobry sposób na zatrzymanie odwiedzających w witrynie.

Poniżej znajduje się przykład realizacji tego pomysłu. Cała struktura jest oparta na nieuporządkowanej liście, ponieważ to prawdopodobnie najlepszy wybór w tym przypadku:

```
<ul class="random-attachments">
    <?php $new_query = new WP_Query( '&showposts=4' ); ?>
    <?php while ( $new_query->have_posts() ) : $new_query->the_post(); ?>
    $args = array(
         'post_type' => 'attachment',
         'numberposts' => 1,
         'orderby' => rand,
         'status' => 'publish',
         'post_mime_type' => 'image',
         'parent' => $post->ID
    );
    $attachments = get_posts( $args );
    if ( $attachments ) {
        foreach ( $attachments as $attachment ) {
    echo '<li>';
    echo wp_get_attachment_link( $attachment->ID, 'thumbnail', true, '' );
    echo '</li>';
        }
    }
    ?>
<?php endwhile; ?>
</ul>
```

Na początku tworzona jest nowa pętla z ograniczeniem liczby wpisów do wyświetlenia. Dalej znajduje się definicja tablicy o nazwie `$args`. W WordPressie często tak się robi. Tworzy się tablicę z argumentami, którą następnie przekazuje się do odpowiedniej funkcji. W tym przypadku parametry w tablicy należą do funkcji `get_posts()`, za pomocą której decydujemy, co ma zostać ostatecznie wyświetlone. Niedługo do tego dojdziemy.

Znaczenie parametrów zapisanych w tablicy `$args` jest oczywiste. Wybierane mają być tylko wpisy typu załącznikowego, pokazywany ma być tylko jeden załącznik na wpis (w przeciwnym razie otrzymamy kilka obrazów zamiast jednego), kolejność ma być losowa, używane mają być tylko opublikowane wpisy oraz pod uwagę brane mają być tylko obrazy (typ MIME `image`). Na koniec dodajemy identyfikator wpisu nadrzędnego. Nie jest to konieczne w tym przypadku, ale może się przydać w przyszłości.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Wszystkie te informacje są zapisane w tablicy o nazwie `$args`. Teraz wystarczy tylko przekazać ją do funkcji `get_posts()`:

```
$attachments = get_posts( $args );
```

Teraz wynik działania funkcji `get_posts()` z parametrami z tablicy `$args` jest zapisany w zmiennej `$attachments`, której użyjemy w pętli `foreach`. W pętli tej znajduje się następujący kod sterujący sposobem wyświetlania załączników:

```
echo wp_get_attachment_link( $attachment->ID, 'thumbnail', true, '' );
```

Tag szablonowy `wp_get_attachment_link()` domyślnie wyświetla odpowiedni załącznik. Tutaj podany mu został identyfikator z pętli `foreach` oraz ustawiono, aby wyświetlał miniatury, a nie obrazy w oryginalnych wymiarach. Parametr `true` oznacza, że wynik ma zostać podlinkowany do strony załącznika. Ostatni parametr określa, czy ma być wyświetlona ikona medium. Domyślna wartość to `false` i jeśli się nic nie wpisze, to właśnie ona jest stosowana.

Ten krótki fragment kodu umieszczony w pasku bocznym spowoduje wyświetlenie nieuporządkowanej listy losowych miniatur, których kliknięcie będzie powodować przejście na odpowiednie strony.



## DODATKOWE OPCJE LOSOWANIA OBRAZÓW

Nadal Ci mało? Jeśli tak, to powinieneś zainteresować się fantastycznym światem serwisów do publikowania zdjęć i udostępnianymi przez nie widżetami, kodami do osadzania treści oraz wtyczkami (więcej szczegółów znajduje się w podrozdziale „Optymalne wykorzystanie serwisów do publikowania zdjęć"). Dobrym pomysłem jest np. losowe wyświetlenie swoich najnowszych prac w gustownej ramce. Poszukaj różnych rozwiązań, a na pewno znajdziesz coś odpowiedniego dla siebie. Ponadto jeśli jeszcze nie przeglądałeś dodatku A, to poszukaj w nim opisów ciekawych wtyczek i sugestii.

Drobne ostrzeżenie: nie przesadzaj z ilością treści i kontroluj czas ładowania stron.

Zewnętrzne usługi zawsze wydłużają czas ładowania stron, a jeśli z usług tych pobierane są obrazy graficzne, które mogą nie być właściwie skompresowane, to czas ten może być znacznie wydłużony. Wiem, że może Cię kusić, aby użyć fajnego widżetu we Flashu udostępnianego przez jakiś serwis do publikowania zdjęć albo wtyczki wyświetlającej Twoje najlepsze zdjęcia, ale bądź ostrożny.

Ponadto należy się zastanowić, czy w ogóle potrzebuje się bloku z obrazami. To, że masz dostęp do strumienia grafik, wcale nie oznacza, że musisz go używać. Czy odwiedzających to zainteresuje? Jeśli nie, to zapomnij o tym i wykorzystaj cenne miejsce na ekranie do wyświetlenia czegoś innego.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### Kreatywne sposoby wykorzystania obrazów promocyjnych i nagłówkowych

Dwa rodzaje obrazów można w projektach wykorzystać w jeszcze bardziej pomysłowy sposób: są to obrazy nagłówkowe i ikony. Jak wiesz, obraz nagłówkowy jest często używany jako tradycyjny nagłówek bloga, ale to nie oznacza, że musi tak być. Natomiast ikona to obraz przypisany do wpisu i najczęściej używany jako jego symbol lub w podobny sposób.

Ale te rodzaje obrazów można także wykorzystać na inne sposoby! Oto kilka pomysłów:

- Obrazu nagłówka można używać do wyświetlania wiadomości dla użytkowników, np. powitań itp.
- Cele marketingowe — obrazu nagłówkowego można użyć do promowania produktu, wydarzenia, usługi, osoby itd.
- Ikony można użyć jako tła. Kto powiedział, że ikona musi znajdować się w tytule wpisu? Równie dobrze możesz ją wyświetlić w tle wpisu.
- Ikona jako nagłówek — czy wiesz, że motyw Twenty Ten w nagłówkach wpisów wstawia ikony tych wpisów, jeśli są zdefiniowane? Dzięki temu każdy wpis może mieć własny nagłówek.

## OPTYMALNE WYKORZYSTANIE SERWISÓW DO PUBLIKOWANIA ZDJĘĆ



Dla właścicieli witryn działających na serwerach nie najwyższej klasy lub dzielonych hostingach bardzo ważnym czynnikiem jest przestrzeń dyskowa i przepustowość. Trudno wyobrazić sobie lepsze rozwiązanie dla takich osób niż przechowywanie filmów i obrazów w takich serwisach jak YouTube i Flickr. To samo dotyczy także większych portali nie generujących przychodu, chociaż ich właściciele zwykle są w stanie zapewnić specjalne rozwiązania, jak przechowywanie danych w chmurze lub statycznych plikach na serwerze.

Przechowywanie zdjęć w jednym z serwisów do publikowania fotografii na pewno pozwala zaoszczędzić miejsce dyskowe i transfer na koncie hostingowym, zwłaszcza jeśli w witrynie publikowanych jest dużo grafik. Dobrym przykładem może być portal z grami, w którym wyświetlane są zrzuty ekranu o rozmiarach sięgających nawet do 700 kB. Dodatkowo dla dużych ekranów i monitorów o rozdzielczości HD również trzeba dostarczać wysokiej jakości obrazy. Wyobraźmy sobie, że codziennie publikujemy dwie galerie po 10 zdjęć. To oznacza 140 zdjęć tygodniowo i 600 miesięcznie. Jeśli każde z nich zajmuje około 700 kB, to miesięcznie wychodzi 420 MB. Nie trzeba być matematykiem, aby zrozumieć, że utrzymanie takiego portalu wymaga dużych ilości transferu i przestrzeni dyskowej.

Dlatego bardzo przydatne są portale do publikowania zdjęć. Wysyłając swoje zdjęcia do serwisu Flickr, nie musisz przejmować się tymi kwestiami, ponieważ cały ciężar utrzymania danych spada na ten serwis. Może przechowywanie zrzutów ekranu dla serwisu z grami komputerowymi w portalu Flickr byłoby przesadą, ale np. dla fotografa może to być znakomite rozwiązanie.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

W ten sposób sporo oszczędza się na filmach. W dzisiejszych czasach mało kto przechowuje filmy na własnych serwerach. Zamiast tego korzysta się z usług osadzania treści z serwisów typu YouTube i Vimeo. Dlaczego by nie zastosować takiego samego rozwiązania dla grafik?

Oczywiście rozwiązania te mają wady. Przede wszystkim jeśli portal, w którym przechowujesz pliki, zniknie, to wraz z nim pójdą w niepamięć Twoje dane. Oczywiście jeśli będziesz przechowywał kopię zapasową wszystkich plików, to będziesz mógł je odzyskać, tylko trochę się napracujesz. Z drugiej strony, jeżeli skorzystasz z usług największych portali, to istnieje niewielkie ryzyko, że któryś z nich zniknie z sieci. Jeśli uznasz, że YouTube jest niezawodny, to tak samo możesz traktować Flickra i podobne.

Kolejną kwestią jest czas ładowania. Jeśli serwis, w którym przechowujesz obrazy, będzie miał problemy z dostarczaniem treści, Twoja strona również na tym ucierpi. Osobiście uważam, że obrazy są bardziej, jak je nazywam, „bezpośrednią treścią" niż filmy i dlatego wolę przechowywać je lokalnie, gdy tylko mam taką możliwość. Jeśli muszę je gdzieś przenieść, to najpierw szukam serwerów plików i rozwiązań chmurowych, a dopiero na końcu decyduję się na skorzystanie z czysto zewnętrznych usług.

To oczywiście nie oznacza, że w ogóle nie biorę pod uwagę możliwości użycia serwisów zewnętrznych, zwłaszcza że jestem aktywnym użytkownikiem Flickra, którego czasami używam także do przechowywania zdjęć. Jest to wygodne i szybkie rozwiązanie, które pozwala zaoszczędzić zarówno przestrzeń dyskową, jak i transfer. Niezależnie od tego, co napisałem wcześniej, są to twarde fakty, którym trudno zaprzeczyć.

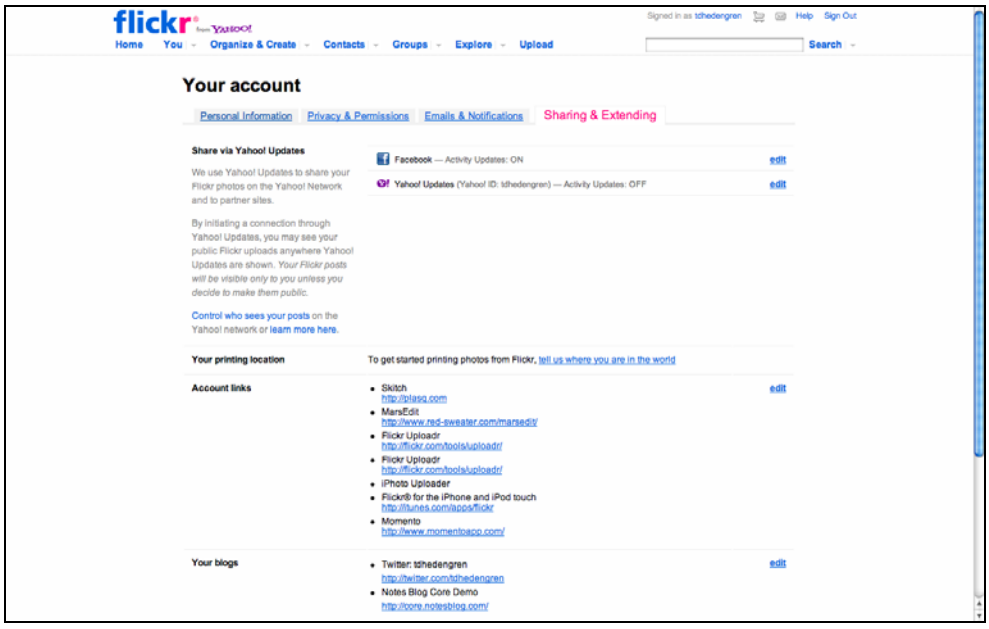


## PRZECHOWYWANIE OBRAZÓW W SERWISIE FLICKR

Flickr (*http://flickr.com*) to prawdopodobnie najpopularniejszy na świecie serwis do publikowania zdjęć. Pewną liczbę zdjęć można opublikować za darmo, a jeśli chce się więcej, można wykupić lepsze konto. Ponadto serwis ten bardzo dobrze współpracuje z blogami i umożliwia nawet publikowanie własnych (jak również cudzych) zdjęć bezpośrednio na blogach. Trzeba tylko odpowiednio zdefiniować ustawienia konta. WordPress jest tylko jedną z wielu platform blogowych, w których można publikować zdjęcia z Flickra. Poniżej znajduje się opis, jak to zrobić.

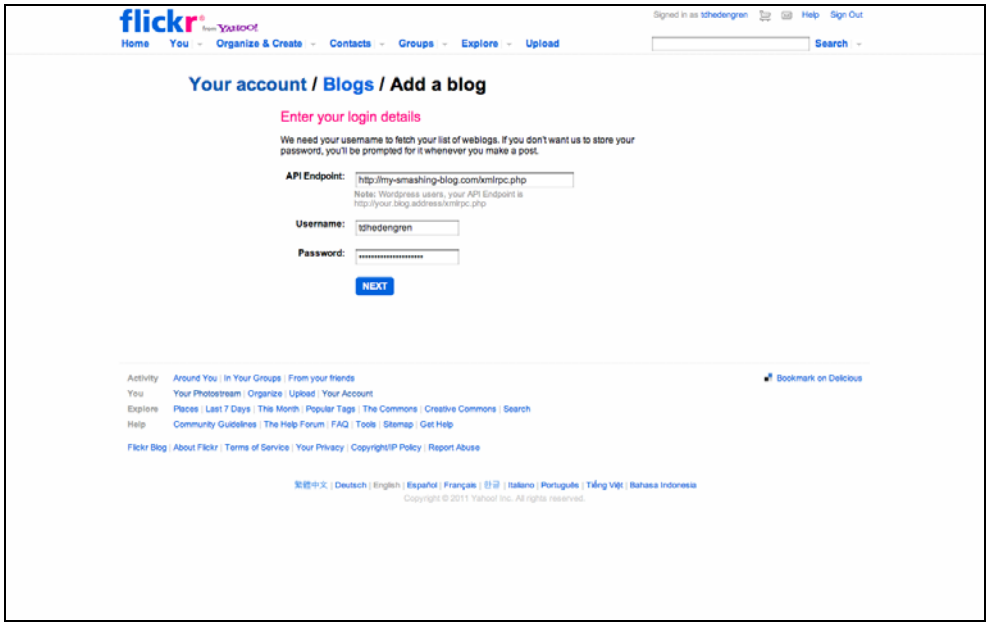
1. Zaloguj się na swoim koncie w serwisie Flickr i przejdź na stronę *Your Account* (twoje konto).
2. Kliknij kartę *Sharing & Extending*, a następnie kliknij łącze *Edit*, które znajduje się po prawej stronie sekcji *Your Blogs* (twoje blogi) — rysunek 12.5.
3. Dodaj swojego bloga, wypełniając formularz (rysunek 12.6). Musisz też w swoim blogu włączyć funkcję publikowania poprzez XML-RPC (*Ustawienia/Pisanie*). W serwisie Flickr należy podać adres pliku *xmlrpc.php*, który znajduje się w katalogu głównym WordPressa.
4. Przejrzyj poradnik i wybierz (oraz ewentualnie dostosuj) domyślny układ wpisów publikowanych z Flickra.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 12.5. Łącze Edit znajduje się po prawej stronie listy blogów






Rysunek 12.6. Formularz na dane bloga

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

To wszystko! Od tej pory możesz wysyłać zdjęcia prosto z serwisu Flickr do swojego bloga. Znajdź zdjęcie, które chciałbyś opublikować, kliknij łącze *Share This* (opublikuj to) znajdujące się w prawym górnym rogu, a następnie wybierz bloga na karcie *Blog It* (opublikuj w blogu). Niestety nie wszystkie zdjęcia z Flickra można tak udostępniać. Niektóre są udostępniane na licencjach wykluczających wykorzystanie ich w ten sposób.

Flickr jest powszechnie wykorzystywany nie tylko do publikowania zdjęć, ale również do oszczędzania ruchu. Serwując zdjęcia z tego portalu, nie obciążasz własnego serwera. Oczywiście jest to broń obosieczna, ponieważ jeśli Flickr będzie miał kłopoty z dostarczaniem treści (albo całkiem wyleci z biznesu), Twoje obrazy również będą w opałach. Dlatego zawsze trzymaj kopię zapasową wszystkich plików.

## UŻYWANIE POKAZÓW SLAJDÓW Z SERWISU FLICKR

Jeśli jesteś aktywnym użytkownikiem serwisu Flickr, to może zechcesz osadzić w swojej witrynie pokaz slajdów z tego portalu. Jest to niewielki widżet we Flashu, który można umieścić w dowolnym miejscu na stronie, a nawet można w razie potrzeby zmienić jego wymiary. Pokaz slajdów można utworzyć dla dowolnej grupy obrazów, nawet z całego strumienia zdjęć wybranego użytkownika. Oczywiście pokaz taki może zawierać wyłącznie publicznie udostępniane zdjęcia.



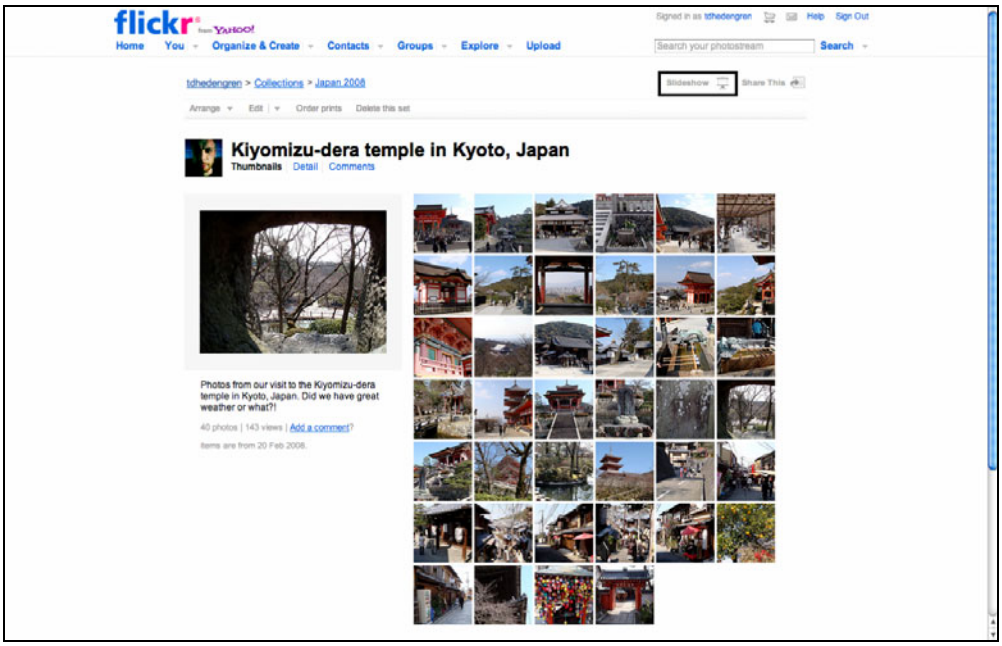
Dodanie pokazu slajdów do strony jest łatwe. Należy znaleźć strumień zdjęć i kliknąć łącze *Slideshow* (pokaz slajdów) znajdujące się w prawym górnym rogu (rysunek 12.7) obok łącza *Share This*. Jeśli łącze *Share This* jest widoczne, to znaczy, że dane zdjęcie można dodać do pokazu slajdów. Jeśli go nie ma — to zdjęcia nie można dodać. Proste.

Pokaz slajdów rozpoczyna się (rysunek 12.8). Kliknij łącze *Share* w prawym górnym rogu, a następnie kliknij łącze *Customize This HTML* (dostosuj ten kod HTML) znajdujące się pod kodem osadzania. Zostanie wyświetlone nowe okno, w którym można ustawić własną szerokość i wysokość lub wybrać wartości standardowe. Skopiuj kod osadzania i wklej go w wybranym miejscu na stronie.

Kiedy pokazy slajdów z Flickra są przydatne? Gdy chce się wykorzystać Flickra do przekazywania treści i oszczędzania ruchu. W pozostałych sytuacjach preferuję jednak tworzenie atrakcyjnych galerii. Pokaz slajdów może być ciekawym uzupełnieniem, można go np. wyświetlić w pasku bocznym, aby chwalić się swoimi najnowszymi osiągnięciami.

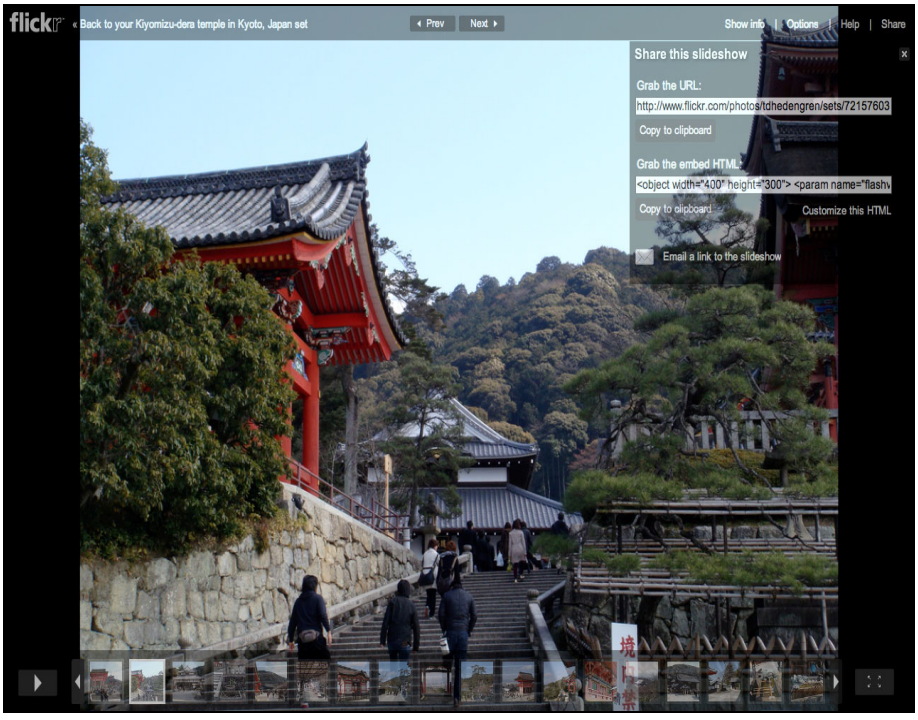
Poza tym pokazy slajdów w serwisie Flickr są nowością. Jeśli chcesz wykorzystać ten portal do czegoś więcej niż przechowywanie statycznych obrazów, musisz sięgnąć do API (*www.flickr.com/services/api*) i licznych wtyczek, za pomocą których można zdziałać cuda.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 12.7. Łącze Slideshow łatwo przeoczyć, ale najczęściej jest ono widoczne






Rysunek 12.8. Kod osadzania lepiej jest pobrać ze strony niż bezpośrednio z pokazu slajdów, dzięki czemu można ustawić więcej opcji

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# STRZEŻ SIĘ BAŁAGANU

Obrazy i inne media wizualne doskonale uatrakcyjniają strony wizualnie, zwłaszcza gdy użyje się serwisów typu Flickr i ich widżetów pozwalających chwalić się najnowszymi zdjęciami. Użytkownicy lubią, gdy strona wygląda na żywą, a więc warto o to dbać. Tak przynajmniej jest w idealnej sytuacji. Jednak w rzeczywistości na stronach można czasami znaleźć nadmiar grafik powodujący bałagan.

Krótko mówiąc, używaj tych narzędzi z głową, niezależnie od tego, czy wyświetlasz losowy obraz w nagłówku, czy najnowsze zdjęcia z Flickra. Wszystko musi trzymać się kupy i współgrać z resztą projektu.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 13 DODATKOWE FUNKCJE

**W WORDPRESSIE MOŻNA** zrobić wiele różnych rzeczy. W tym rozdziale pokażę Ci kilka fajnych funkcji, które możesz zechcieć wykorzystać w swoim projekcie lub w jakiejś innej witrynie. Dowiesz się, jak wyświetlać różne treści na kartach, tworzyć formularze logowania, korzystać z kanałów RSS, przystosowywać strony do druku i wiele więcej. Większość opisanych komponentów można wykorzystać na wiele różnych sposobów, a ponieważ każda witryna jest inna, także sposoby użycia tych elementów mogą być różne.

W opisie każdego elementu dodatkowo wyjaśniam, kiedy dana technika może być najbardziej przydatna, a kiedy można o niej zapomnieć. Oprócz wtyczek i różnych funkcji, jakie można dodać do WordPressa, istnieje jeszcze ogromna ilość opcji, przez co bardzo łatwo można zapełnić witrynę rzeczami, które nie są w niej wcale potrzebne. Każdy dodatek należy dokładnie przemyśleć, nawet jeśli trudno jest się oprzeć niektórym fajerwerkom.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# WYŚWIETLANIE TREŚCI NA KARTACH

Karty pozwalają oszczędzić sporo cennego miejsca na ekranie. W blogach i innych dynamicznych rodzajach witryn na kartach wyświetla się popularne wpisy, ostatnio komentowane artykuły i inne tego typu treści, które można logicznie zgrupować w jednym miejscu. Należy tylko uważać, aby na karcie, która domyślnie nie jest widoczna, nie ukryć jakichś ważnych informacji, które powinny być widoczne cały czas. Kluczowe w tej technice jest to, aby nie ukryć czegoś ważnego na niewyświetlonej karcie.

Zasadniczo utworzenie kart i zarządzanie nimi nie jest trudne. Czasami nawet tworzony komponent nie musi wyglądać jak zestaw kart. Jest to raczej rodzaj funkcji niż element projektu.

## INTELIGENTNE ZASTOSOWANIE

Nie dość że utworzenie własnego komponentu z kartami jest łatwe, to na dodatek w internecie można znaleźć wiele gotowych do użycia skryptów, jeśli komuś nie chce się pisać własnego rozwiązania od podstaw. Kod przedstawiony poniżej jest na tyle prosty, że można go wykorzystać w większości przypadków, a jednocześnie w razie potrzeby da się go uatrakcyjnić wizualnie. W sumie gdy już użyjesz komponentu z kartami, to czemu miałbyś go nie dostosować wyglądem do projektu witryny?



Kod ten do działania wymaga biblioteki jQuery, którą należy załadować za pomocą funkcji `wp_enqueue_script()` omówionej wcześniej. Przyjmuję założenie, że biblioteka ta jest już załadowana. Poniższa część skryptu powinna zostać wczytana w sekcji nagłówkowej motywu, najlepiej z osobnego pliku za pomocą funkcji `wp_enqueue_script()` w pliku *functions.php*:

```
<script>
// Funkcja wyświetlająca kartę
function viewTab(tabId) {
    // Pobiera wszystkie elementy potomne elementu contents-container
    var elements = jQuery('#contents-container').children();
    // Iteracja przez te elementy
    jQuery.each(elements, function(index, value) {
        // Sprawdzenie, czy karta została kliknięta
        if (jQuery(this).attr('id') == tabId) {
            // Wyświetlenie elementu
            jQuery(this).css({ 'display':'block' });
            // Dostosowanie CSS
            jQuery('#tab-'+ jQuery(this).attr('id')).addClass('active-tab');
        }
        // Jeśli karta nie jest kliknięta
        else {
            // Ukrycie karty
            jQuery(this).css({ 'display':'none' });
            // Dostosowanie CSS
            jQuery('#tab-'+ jQuery(this).attr('id')).removeClass('active-tab');
        }
    });
}
</script>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Jeśli chcesz tylko wypróbować ten kod, możesz go umieścić bezpośrednio w nagłówku strony w pliku *header.php*. Nie zostawiaj jednak takiego ustawienia na zawsze, ponieważ skrypty JavaScript powinno się ładować z osobnych plików.

Poniżej znajduje się kod HTML, z którego zostaną utworzone karty:

```
<ul id="tabs">
    <li id="tab-content-recent" class="active-tab">
        <a href="javascript:viewTab('content-recent');">Najnowsze</a>
    </li>
    <li id="tab-content-popular">
        <a href="javascript:viewTab('content-popular');">Popularne</a>
    </li>
    <li id="tab-content-comments">
        <a href="javascript:viewTab('content-comments');">Komentarze</a>
    </li>
</ul>
<div id="contents-container">
    <div id="content-recent">
        Treść karty Najnowsze.
    </div>
    <div id="content-popular" style="display: none;">
       Treść karty Popularne.
    </div>
    <div id="content-comments" style="display: none;">
        Treść karty Komentarze.
    </div>
</div>
```



Trzeba napisać arkusze stylów, aby dostosować wygląd tego komponentu do własnych potrzeb, i wcale nie musi on wyglądać jak zestaw kart. Podstawą działania tego kodu jest to, że łącza znajdujące się w elementach listy powodują wyświetlenie kontenerów `div` zawierających treść kart.

Wystarczy tylko dodać kod wyświetlający najnowsze i popularne wpisy oraz komentarze i praca będzie skończona. Nic wielkiego.

Jeśli jednak chciałbyś ułatwić obsługę tego komponentu, mógłbyś dla każdego kontenera `div` utworzyć obszar widżetów, dzięki czemu w razie potrzeby mógłbyś łatwo zamienić dowolną funkcję na wtyczkę. Aby zrealizować ten pomysł, trzeba trochę popracować nad widżetami. Należy dla każdego kontenera `div` za pomocą funkcji `register_sidebar()` utworzyć obszar widżetów zgodnie z opisem w rozdziale 4.

W poniższym kodzie dodano fikcyjny pasek boczny (który trzeba by było utworzyć w pliku *functions.php*) do każdej karty. Paski te są umieszczone w elementach `div`:

```
<ul id="tabs">
    <li id="tab-content-recent" class="active-tab">
        <a href="javascript:viewTab('content-recent');">Najnowsze</a>
    </li>
```

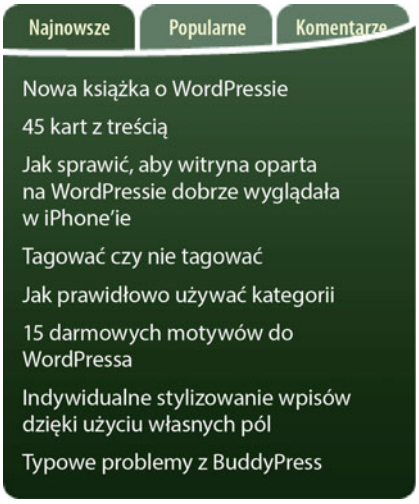
ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    <li id="tab-content-popular">
        <a href="javascript:viewTab('content-popular');">Popularne</a>
    </li>
    <li id="tab-content-comments">
        <a href="javascript:viewTab('content-comments');">Komentarze</a>
    </li>
</ul>
<div id="contents-container">
    <div id="content-recent">
        <!-- Obszar widżetów karty Najnowsze -->
        <?php dynamic_sidebar( 'Najnowsze wpisy' ); ?>
    </div>
    <div id="content-popular" style="display: none;">
        <!-- Obszar widżetów karty Popularne -->
        <?php dynamic_sidebar( 'Popularne wpisy' ); ?>
    </div>
    <div id="content-comments" style="display: none;">
        <!-- Obszar widżetów karty Komentarze -->
        <?php dynamic_sidebar( 'Najnowsze komentarze' ); ?>
    </div>
</div>
```



Teraz treść na każdej karcie możesz umieszczać w obszarze widżetów w panelu administracyjnym WordPressa, podobnie jak to robisz w przypadku widżetów stopki i paska bocznego. Oczywiście komponentu tego można używać także do wielu innych celów.

Na rysunku 13.1 pokazano, jak może wyglądać opisywany komponent po lekkim poprawieniu przedstawionego wcześniej kodu.




Rysunek 13.1. Karty ułatwiają zorganizowanie treści i nawigacji na stronie

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### UŻYWAĆ KART CZY NIE

Czy umieszczanie treści na kartach to dobry, czy zły pomysł? Wszystko sprowadza się do tego, jakiego rodzaju treść publikujesz i w jakim typie witryny to robisz. Przede wszystkim obsługa kart musi być intuicyjna i prosta dla każdego użytkownika. Treść znajdująca się na wstępnie ukrytych kartach nie może być najważniejsza, ponieważ niektórzy użytkownicy nie lubią klikać. Ponadto taka treść może zostać w ogóle przeoczona przez przypadkowych odwiedzających, którzy tylko szybko przejrzą stronę i na podstawie tego, czy coś ich na niej zainteresowało, zdecydują, czy na niej pozostać. Nie możesz ukrywać tego, co masz najlepsze na kartach, podobnie jak nie powinieneś obrzydzać użytkownikom korzystania z menu i wyszukiwarki.

Niemniej jednak karty w wielu witrynach mogą być wartościowym elementem, zwłaszcza gdy chcemy zaoszczędzić miejsce na ekranie, ukrywając różne listy treści itp. Czy np. odnośniki do polecanych witryn muszą być cały czas widoczne? Można by je umieścić na jednej karcie, na drugiej można by umieścić profile portali społecznościowych, na jeszcze innej odnośniki do zaprzyjaźnionych witryn itd. W ten sposób zaoszczędzisz sporo miejsca, a użytkownik, który będzie chciał dowiedzieć się np. tego, na jakich portalach społecznościowych można Cię znaleźć, kliknie odpowiednią kartę. Użytkownicy tacy lubią, gdy treść jest wyświetlana na kartach, ponieważ dzięki nim przeglądanie zawartości witryny jest często łatwiejsze.

Ogólnie rzecz biorąc, kart do wyświetlania treści i budowy menu należy używać z rozwagą. To dodatkowe kliknięcie może nie podobać się niektórym użytkownikom. Ale wypchanie stron rzadko używanymi elementami również nie jest dobre. Trzeba znaleźć złoty środek.



## WYŚWIETLANIE ZAWARTOŚCI KANAŁÓW RSS

Kanały RSS to bardzo przydatne narzędzie nie tylko do subskrybowania wiadomości w specjalnym czytniku, ale również do wyświetlania na stronach treści partnerów lub z innych własnych projektów. Zapewne znasz już standardowy widżet RSS WordPressa. Za jego pomocą można wyświetlać wiadomości z dowolnego kanału RSS.

Ale nie musisz ograniczać się do używania tylko tego widżetu, a nawet jakiegokolwiek widżetu. Możesz podłączyć się do parsera kanałów i sterować udostępnianą w nich treścią na wiele różnych sposobów. Jeśli chcesz wysłać opis, to nie ma problemu, podobnie jak nie ma problemu, jeśli chcesz ująć w kod HTML wybrane elementy, aby mieć nad nimi większą kontrolę. Jeżeli chcesz wykonać jakieś bardziej zaawansowane działania na kanałach RSS, możesz użyć wtyczki albo napisać własny kod przy użyciu wbudowanych funkcji (z których wspomniana wtyczka prawdopodobnie również by korzystała!).

Należy jednak pamiętać, że wyświetlanie kanałów RSS może spowolnić działanie witryny. Jeśli będziesz pobierać nowości z kilku portali (a nawet tylko jednego), zauważysz, że czasami strony będą otwierać się z opóźnieniem. Jest to spowodowane tym, że najpierw musi zostać pobrana treść z kanału RSS, która następnie jest poddawana obróbce przez PHP, a dopiero potem może zostać wyświetlona. Dotyczy to w równym stopniu skryptów wpisanych w plikach motywu bezpośrednio, jak i widżetów RSS. Dlatego jeśli w swojej witrynie wyświetlasz

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

zawartość wielu kanałów RSS, rozważ możliwość użycia jakiegoś rozwiązania buforującego, aby zmaksymalizować szybkość działania stron. W zasadzie najlepszym wyjściem jest uruchomienie zadania crona, które co określony czas pobierałoby treść i zapisywało ją na serwerze. Możesz też użyć jakiejś wtyczki buforującej (opis kilku znajduje się w dodatku A) albo skorzystać z API Transients, do którego wkrótce dojdziemy.

Tak, kanały RSS są bardzo przydatne, ale jeśli rozejrzysz się po różnych blogach i przyjrzysz się, co najbardziej spowalnia ładowanie stron, to dowiesz się, że kanały te należą do czołówki pod tym względem, obok innych funkcji pobierających treści z zewnątrz (np. reklamy). Krótko mówiąc, uważaj, aby Twoja witryna nie została spowolniona przez kanały RSS.

## WBUDOWANY PARSER

WordPress zawiera wbudowany mechanizm publikowania treści w kanałach RSS. Można używać widżetu RSS, o którym wspominałem wcześniej, ale najczęściej potrzebna jest większa kontrola nad tą funkcją. Na szczęście osiągnięcie jej nie jest trudne, a to dzięki dodaniu SimplePie do rdzenia WordPressa. To prawda, w WordPressie 2.8 dodano do rdzenia nowy parser kanałów RSS. Stare funkcje, takie jak `wp_rss()`, nadal działają, ale są przestarzałe i powinno się je zmienić na nowsze, np. `fetch_feed()`.



Dokumentacja SimplePie może się wydawać skomplikowana przy pierwszym zetknięciu, ale w rzeczywistości system ten jest bardzo łatwy w użyciu. Na przykład poniższy kod dodaje najnowsze nagłówki z łączami z witryny *http://tdh.me/*:

```
<ul>
<li><h2>TDH.me</h2></li>
<?php $feed = fetch_feed( 'http://tdh.me' );
    foreach ( $feed->get_items() as $item ){
        printf( '<li><a href="%s">%s</a></li>', $item->get_permalink(),
            $item->get_title() );
    }
?>
</ul>
```

Dołączając kanał w ten sposób, otwieram go dla funkcji SimplePie. W kodzie tym znajduje się pętla przeglądająca i wyświetlająca zawartość kanału. W tym przykładzie pobierane są tylko łącze i tytuł każdego wpisu, ale można łatwo dodać datę i opis:

```
<ul>
<li><h2>TDH.me</h2></li>
<?php $feed = fetch_feed( 'http://tdh.me' );
    printf( $feed->get_title() );
    foreach ( $feed->get_items() as $item ){
        printf( '<li><a href="%s">%s</a></li>', $item->get_permalink(),
            $item->get_title() );
        printf( '<p>%s</p>', $item->get_description() );
        printf( '<p><small>%s</small></p>', $item->get_date('j F Y at g:i a') );
}
?>
</ul>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

System SimplePie jest ogromny i żeby go opisać, trzeba by było napisać nową książkę. Wszystkie potrzebne informacje znajdziesz w dokumentacji, która jednak nie jest niestety w pełni zgodna z WordPressem, ponieważ SimplePie to mimo wszystko samodzielny parser RSS. W każdym bądź razie znajdziesz ją pod adresem *http://simplepie.org/wiki/*.

Jeszcze wrócimy do SimplePie, ale najpierw kilka słów na temat API Transients.

## BUFOROWANIE PRZY UŻYCIU API TRANSIENTS

Oczekiwanie, aż system pobierze treść z kanałów RSS, może spowodować spowolnienie wyświetlania się stron. Dlatego warto zastanowić się nad zastosowaniem buforowania treści. Są wtyczki, które do tego służą, ale jest to też dobry moment na omówienie API Transients. Jest to bardzo przydatne narzędzie do tymczasowego przechowywania danych w bazie danych. Bardziej szczegółowe informacje na jego temat można znaleźć w dokumentacji pod adresem *http://codex.wordpress.org/Transients_API*.

Przedstawiony wcześniej przykład z kanałem RSS jest doskonałym kandydatem do zastosowania rozwiązania polegającego na tymczasowym przechowywaniu danych. Najpierw za pomocą funkcji `get_transient()` sprawdza się, czy zapisana jest jakaś treść, którą można wysłać na wyjście, a następnie za pomocą funkcji `set_transient()` zapisuje się treść, jeśli jest to potrzebne. W końcu bufor służy do przechowywania danych tymczasowo, dlatego dane w nim zapisane powinno się co jakiś czas zastępować nowymi. Na tym polega najważniejsza różnica między API Transients a API Options — w tym pierwszym jest możliwość ustawienia daty wygaśnięcia zapisanej treści.



Spójrzmy, jak moglibyśmy wykorzystać API Transients w poprzednio prezentowanym kodzie:

```
<ul>
<li><h2>TDH.me</h2></li>

<?php
// Próba pobrania treści z bufora
if ( ! ( $feed = get_transient( 'smashing_feedcache' ) ) ) {
    // Brak treści w buforze, a więc pobieramy treść z kanału i ją tam zapisujemy
    $feed_obj = fetch_feed( 'http://tdh.me' );
    $feed = array();
    $feed['title'] = $feed_obj->get_title();
    $feed['items'] = array();
    foreach ( $feed_obj->get_items() as $item ){
        // Zapisanie treści kanału w postaci tablicy
        $feed['items'][] = array(
            'permalink' => $item->get_permalink(),
            'title' => $item->get_title(),
            'description' => $item->get_description(),
            'date' => $item->get_date('j F Y at g:i a')
        );
    }
    // Zapisanie na godzinę
    set_transient( 'smashing_feedcache', $feed, ( 60*60 ) );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
}
// Zwracanie treści
foreach ( $feed['items'] as $item ) {
    printf( '<li><a href="%s">%s</a></li>', $item['permalink'], $item['title'] );
    printf( '<p>%s</p>', $item['description'] );
    printf( '<p><small>%s</small></p>', $item['date'] );
} ?>
</ul>
```

Najpierw sprawdzamy, czy w bazie danych jest zapisane coś, co chcielibyśmy zwrócić. W tym celu użyliśmy funkcji `get_transient()` do szukania danych związanych z `smashing_feedcache`. Jeśli nie ma nic do zwrócenia, pobieramy dane z kanału RSS za pomocą funkcji `fetch_feed()` i przechowujemy je w postaci tablicy. Następnie zapisujemy dane za pomocą funkcji `set_transient()` i ustawiamy czas wygaśnięcia na godzinę (60 minut × 60 sekund). Instrukcja zwracająca dane pobiera zawartość tablicy, którą zapisaliśmy w bazie danych.

## MIESZANIE KANAŁÓW ZA POMOCĄ SIMPLEPIE

Dzięki SimplePie możemy korzystać z wielu kanałów RSS jednocześnie i wcale nie mam na myśli możliwości wyświetlenia kilku bloków z treścią RSS na jednej stronie. O nie, chodzi o coś całkiem innego, a mianowicie o mieszanie zawartości wielu kanałów RSS i tworzenie z nich jednego kanału. W SimplePie nazywa się to **multikanałem** (ang. *multifeed*) — warto znać tę nazwę, ponieważ może się przydać przy przeszukiwaniu dokumentacji.



Poniżej znajduje się prosty kod, który pozwoli nam przetestować omawianą funkcję. Skrypt ten pobiera dwa kanały i wyświetla ich zawartość według dat. Liczba elementów jest ograniczona do 10:

```
<ul>
<li><h2>Ciekawe nagłówki</h2></li>
<?php $feed = fetch_feed( array( 'http://rss1.smashingmagazine.com/feed/',
    'http://feeds.digg.com/digg/topic/apple/popular.rss' ) );
        $feed->enable_order_by_date( true );
        foreach ( $feed->get_items(0, 9) as $item ){
            printf( '<li><a href="%s">%s</a></li>', $item->get_permalink(),
                $item->get_title() );
            printf( '<p><small>%s</small></p>', $item->get_date('j F Y at g:i a')
);
        }
?>
</ul>
```

W tym przypadku zamiast jednego adresu URL kanału RSS zdefiniowane zostały dwa adresy, które przekazano funkcji `fetch_feed()` w postaci tablicy. W razie potrzeby można w ten sam sposób dodać kolejne adresy. Następnie użyte zostało ustawienie sortowania według dat, którego opis udało się znaleźć w dokumentacji SimplePie:

```
$feed->enable_order_by_date( true );
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Ustawienia tego można by było użyć również w przykładzie z pojedynczym kanałem, chociaż w takim przypadku i tak wiadomo, że najnowszy wpis zostanie wyświetlony na początku listy.

Dalej znajduje się pętla `foreach` rozpoczynająca działanie od pierwszego elementu (0), a kończąca na dziesiątym (9). W ten sposób otrzymujemy 10 elementów.

System SimplePie jest bardzo rozbudowany, ale zdobyta tutaj wiedza pozwoli Ci rozpocząć korzystanie z niego. W czasie gdy piszę tę książkę, sekcja pomocy na ten temat w dokumentacji WordPressa jest skąpa, ponieważ funkcja `fetch_feed()` została dodana dopiero w WordPressie 2.8. Myślę jednak, że z czasem to się zmieni. Informacji na temat parsera RSS możesz zawsze szukać w dokumentacji SimplePie. Zainteresuj się też rozwiązaniami buforowania.

# WŁASNE SKRÓTY KODOWE

**Skrót kodowy** to specjalny znacznik, który można umieścić w treści wpisu, aby w tym miejscu coś wyświetlić. Przykładem takiego skrótu jest znacznik `[gallery]`. Skróty kodowe można tworzyć samodzielnie, co jest bardzo przydatne, zwłaszcza w witrynach, które nie są blogami.

## DODAWANIE SKRÓTÓW KODOWYCH



Skrót kodowy można dodać za pomocą wtyczki albo w pliku *functions.php*:

1. Najpierw utwórz prosty skrót kodowy wyświetlający fragment tekstu. Później użyjesz go do robienia czegoś bardziej interesującego.
2. Utwórz funkcję robiącą to, co chcesz. Funkcję tę oczywiście należy wpisać w bloku PHP w pliku *functions.php*:

   ```
   function smashing_text_example() {
       return 'Ten tekst został wyświetlony przez nasz skrót kodowy!
       ↪Fantastycznie!';
   }
   ```

3. Przypisz funkcję `smashing_text_example()` do skrótu kodowego. Do tego służy funkcja `add_shortcode()`:

   ```
   add_shortcode( 'textftw', 'smashing_text_example' );
   ```

   Ten skrypt tworzy skrót `[textftw]` i zastępuje go zawartością funkcji `smashing_text_example()`. Innymi słowy, znacznik `[textftw]` we wpisach będzie zastępowany tekstem *Ten tekst został wyświetlony przez nasz skrót kodowy! Fantastycznie!*.

   Jak widać, tworzenie skrótów kodowych jest bardzo proste. Teraz zrobimy coś bardziej przydatnego, np. skrót kodowy do promowania kanału RSS:

4. Zaczynamy od funkcji. Tym razem będzie zwracać tekst z kodem HTML, za pomocą którego możemy kontrolować wygląd tekstu:

   ```
   function smashing_rss_promotion() {
       return '<div class="rsspromo">Bądź na bieżąco - <a href="http://
   ```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            tdh.me/feed/">subskrybuj mój kanał RSS!</a></div>';
    }
```

5. Następnie tworzymy skrót kodowy i wiążemy go z funkcją `smashing_rss_promotion()`:

```
add_shortcode( 'rsspromo', 'smashing_rss_promotion' );
```

6. W ten sposób utworzyliśmy skrót `[rsspromo]` zwracający element `div` z funkcji `smashing_rss_promotion()`. Pozostało jeszcze tylko napisać parę arkuszy stylów, aby wizualnie uatrakcyjnić zwracany tekst:

```
div.rsspromo {
    background: #eee;
    border: 1px solid #bbb;
    color: #444;
    font-size: 16px;
    font-style: italic; }

div.rsspromo a {
    font-weight:bold;
    font-style:normal; }
```

Możesz napisać dowolne inne reguły, które będą pasować do Twojego projektu.

Teraz za pomocą skrótu `[rsspromo]` możesz wstawiać, gdzie tylko chcesz, komponent promocyjny swojego kanału RSS. W podobny sposób można wstawiać też formularze subskrypcji wiadomości przez e-mail albo reklamy. Dzięki temu gdy znajdziesz reklamodawcę oferującego korzystniejsze warunki, będziesz mógł łatwo zmienić kod.



## CIEKAWOSTKI DOTYCZĄCE SKRÓTÓW KODOWYCH

Może zastanawiasz się, czy skróty kodowe można zagnieżdżać. Można, tylko trzeba pamiętać o poprawnym otwieraniu i zamykaniu znaczników, tak jak w HTML-u. Na przykład poniższy kod jest poprawny:

```
[skrot-1]
    [skrot-2]
    [/skrot-2]
[/skrot-1]
```

Natomiast ten jest niepoprawny i mógłby narobić na stronie bigosu:

```
[skrot-1]
    [skrot-2]
[/skrot-1]
    [/skrot-2]
```

Jednak aby pierwszy z przytoczonych przykładów działał, należy zezwolić na zagnieżdżanie skrótów za pomocą funkcji `do_shortcode()`. Gdybyśmy np. mieli skrót `[pull]` tworzący proste cytaty blokowe, to jego parametr `$content` powinniśmy umieścić w funkcji `do_shortcode()`:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
function pullquote( $atts, $content = null ) {
    extract(shortcode_atts(array(
         'float' => '$align',
    ), $atts));
    return '<blockquote class="pullquote' . $float. '">' .
        do_shortcode($content) . '</blockquote>';
}
add_shortcode( 'pull', 'pullquote' );
```

W przeciwnym razie skróty kodowe umieszczone między znacznikami `[pull]` i `[/pull]` nie będą poprawnie interpretowane.

Pamiętaj, że skróty kodowe działają tylko w treści wpisów. Nie można ich używać w nagłówkach ani nawet wypisach. Są one powiązane z tagiem `the_content()`. Można to jednak zmienić za pomocą funkcji `do_shortcode()` i dodatkowego filtra. Dobrym pomysłem może być dodanie obsługi skrótów kodowych przynajmniej w obszarach widżetów. Wówczas widżet tekstowy może być jeszcze bardziej przydatny.

Poniższy kod włącza obsługę skrótów kodowych w domyślnym widżecie tekstowym. Należy go wpisać do pliku *functions.php* lub wtyczki:

```
add_filter( 'widget_text', 'do_shortcode' );
```

Ta instrukcja włącza obsługę funkcji `do_shortcode()` dla funkcji `widget_text()` jako filtra (oczywiście za pomocą funkcji `add_filter()`). Proste, czyli tak jak lubię!



API skrótów kodowych jest bardzo obszerne. Można o nim poczytać w dokumentacji WordPressa na stronie *http://codex.wordpress.org/Shortcode_API*.

Skróty kodowe są doskonałym sposobem na dodanie do wpisów zarówno nowych funkcji, jak i treści. Czasami jednak trzeba zastosować niestandardowe rozwiązania projektowe, zwłaszcza gdy witryna ma promować inne serwisy.

---

*Nie zapomnij, że skróty kodowe powinny działać także po zmianie motywu. Kilka propozycji rozwiązania tej kwestii znajdziesz w rozdziale 8.*

---

# WYSYŁANIE E-MAILI Z WORDPRESSA

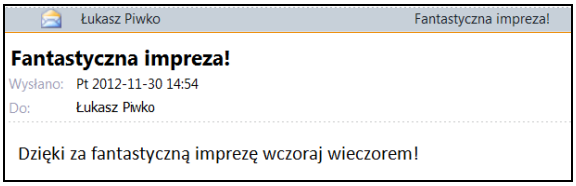
Jeśli chcesz wysyłać za pośrednictwem WordPressa wiadomości e-mail, możesz użyć funkcji `wp_mail()`. Przy jej użyciu możesz wysyłać wszystko — od weryfikacji wykonania jakichś działań po powiadomienia użytkowników o nowościach w serwisie.

Funkcja `wp_mail()` jest łatwa w użyciu. Poniższy fragment kodu, gdy się go aktywuje jakąś funkcją, wysyła na adres *smashing@domain.com* wiadomość e-mail o temacie *Fantastyczna*

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*impreza!* i treści *Dzięki za fantastyczną imprezę wczoraj wieczorem!*. Wynik działania tego skryptu pokazano na rysunku 13.2.

```
wp_mail( 'smashing@domain.com', 'Fantastyczna impreza!', 'Dzięki za fantastyczną
↪imprezę wczoraj wieczorem!' )
```




Rysunek 13.2. Przykładowy e-mail wysłany z WordPressa

Można nawet załączać pliki i używać formatu HTML zamiast zwykłego tekstu. Więcej na ten temat napisano w dokumentacji na stronie *http://codex.wordpress.org/Function_Reference/wp_mail*.

Funkcji `wp_mail()` używa się do różnych celów, np. do wysyłania powitań do nowo zarejestrowanych użytkowników przy użyciu haka `user_register`.

Wysyłając wiadomości e-mail ze stron internetowych, należy pamiętać o kilku ważnych kwestiach:

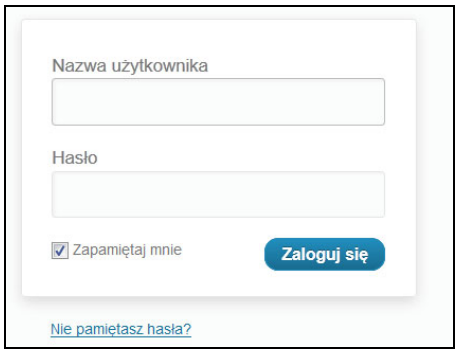


- Nie spamuj. Jest to jedna z najgorszych rzeczy, jakie można zrobić.
- Nie wysyłaj wiadomości e-mail zbyt często, ponieważ możesz znaleźć się na czarnej liście.
- Upewnij się, że Twój kod działa. Drobny błąd może spowodować przypadkowe wysłanie setek zduplikowanych wiadomości, co może przeciążyć serwer i napytać Ci różnych innych problemów.
- Nie pozostawiaj luk w zabezpieczeniach. Upewnij się, że żaden nieprzyjazny użytkownik nie będzie mógł wykorzystać Twojego mechanizmu do rozsyłania spamu. Działania takie mogą być różne, od wysyłania losowych maili na prawo i lewo po wielokrotne naciskanie przycisku zatwierdzenia.
- Niczego nie ukrywaj. Użytkownicy lubią wiedzieć, co się będzie działo z ich adresami e-mail. Dlatego poinformuj ich, w jaki sposób zamierzasz wykorzystywać ich adresy i czego mogą się spodziewać w swoich skrzynkach pocztowych. Jeśli chodzi o pocztę elektroniczną, to nikt nie lubi niespodzianek.

# DODAWANIE FORMULARZA LOGOWANIA

Czasami dobrze jest umieścić formularz logowania (rysunek 13.3) nie tylko na jednej specjalnej stronie (standardowo reprezentowanej przez plik *wp-login.php*). Jeśli prowadzimy witrynę wymagającą logowania w celu skorzystania z różnych funkcji, to warto pokusić się o umieszczenie takiego formularza np. w pasku bocznym albo nagłówku.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 13.3. Prosty formularz logowania

Poniżej przedstawiony jest przykład formularza logowania przeznaczonego do wyświetlenia w pasku bocznym. Pasek ten zazwyczaj jest elementem `ul`, a więc kod ten powinien znajdować się w elemencie `li`. Niemniej jednak możesz go umieścić także np. w nagłówku albo jeszcze gdzie indziej i wówczas najlepiej byłoby go zapisać w kontenerze `div`, aby ułatwić sobie stylizowanie formularza. W budowie tego komponentu najważniejsza jest funkcja `wp_login_form()`:

```
<?php if ( !( is_user_logged_in() ) ){ ?>
<h2>Logowanie</h2>
<?php wp_login_form(); ?>
<p>
    Nie pamiętasz hasła? <a href="<?php echo home_url( '/wp-login.php?
    ➥action=lostpassword' ); ?>">Przypomnij hasło!</a>
</p>
<?php } else { ?>
<h2>Admin</h2>
<ul>
    <li>
        <a href="<?php echo admin_url(); ?>">Kokpit</a>
    </li>
    <li>
        <a href="<?php echo admin_url( 'post-new.php' ); ?>">Dodaj wpis</a>
    </li>
    <li>
        <a href="<?php echo wp_logout_url( urlencode( $_SERVER['REQUEST_URI'] ) );
        ➥?>">Wyloguj</a>
    </li>
</ul>
<?php }?>
```

Najpierw za pomocą instrukcji `if` sprawdzamy, czy użytkownik jest zalogowany, a mówiąc dokładniej: czy jest zalogowany użytkownik z rolą współpracownika lub wyższą. Funkcja `admin_url()` zwraca odpowiednie łącze dla roli użytkownika. W jej drugim wywołaniu został jej dodatkowo przekazany argument `post-new.php`. W podobny sposób używana jest funkcja `home_url()`, za pomocą której wyświetlamy łącze do formularza odzyskiwania hasła.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*Opis ról użytkowników znajduje się w dokumentacji na stronie http://codex.wordpress.org/Roles_and_Capabilities.*

Zatem najpierw sprawdzamy, czy użytkownik jest zalogowany. Jeśli nie jest, zostanie wyświetlony formularz logowania za pomocą funkcji `wp_login_form()`. Funkcja ta przyjmuje kilka parametrów pozwalających zdefiniować różne etykiety określające, gdzie powinien zostać skierowany użytkownik po zalogowaniu itp. Więcej informacji o nich można znaleźć na stronie *http://codex.wordpress.org/Function_Reference/wp_login_form*.

Dalej znajduje się klauzula `else` zawierająca kod dla zalogowanych użytkowników. W tym przypadku wyświetlamy łącza do kokpitu i dodawania nowego wpisu oraz łącze wylogowywania. Łącza te są jednak niepotrzebne, chyba że wyłączono górny pasek WordPressa.

Pamiętaj, aby nie dodawać formularzy logowania do projektu, jeśli nie są one niezbędne. Po co kusić różnych złoczyńców, aby spróbowali złamać Twoje hasło, jeśli nie jest to konieczne? Ponadto nie ma też sensu wyświetlać formularza logowania, jeśli użytkownik nie może zrobić z niego użytku. Tylko zajmie się cenne miejsce na ekranie. Jeżeli wyświetlasz formularz logowania, to niech on będzie przydatny.



# DRUKOWANIE TREŚCI

Niektórzy użytkownicy lubią wydrukować sobie artykuły z bloga, aby móc je przeczytać na papierze. Jeśli prowadzisz witrynę, w której udostępnienie takiej opcji mogłoby być korzystne, powinieneś umożliwić łatwe drukowanie wpisów.

Najpierw dodaj łącze *Wydrukuj*. Można to zrobić za pomocą prostego skryptu JavaScript. Do jego działania nie są potrzebne żadne dodatki ani kolejkowanie jakichkolwiek bibliotek:

```
<a href="javascript:window.print()" rel="nofollow">Wydrukuj</a>
```

To wszystko. Kliknięcie tego łącza spowoduje wydrukowanie strony. Podczas gdy rozwiązanie to jest bardzo proste, to jednak strona może nie być właściwie przygotowana do druku, tzn. może zawierać duże nagłówki, wiele kolumn itd. Dlatego dodatkowo powinno się przygotować arkusz stylów przeznaczony dla druku. Zasadniczo nie musisz tego arkusza umieszczać w osobnym pliku, ale czasami dobrze jest oddzielić reguły przeznaczone dla ekranu od reguł dla druku.

Utwórz plik o nazwie *print.css* i rozejrzyj się po stronach, aby określić, co powinno być drukowane, a co nie. Pasek boczny i stopka najprawdopodobniej nie będą potrzebne, a więc można je usunąć:

```
#sidebar, #footer { display: none; }
```

Nie ma ich! A przynajmniej nie powinno ich być, jeśli mają standardowe identyfikatory `#sidebar` i `#footer`.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Teraz zadbamy o to, aby właściwa treść po wydrukowaniu dobrze wyglądała:

```
#content {
    width:100%;
    margin:0;
    padding:0;
    float:none;
    background: #fff;
    color: #222; }

a:link, a:visited { color: #000; }
```

Prawie czarny tekst, białe tło, pełna szerokość oraz brak pływania, dziwnych marginesów i dopełnień. Dodałem też regułę zamieniającą kolor łączy na czarny (pseudoklasy `:active` i `:hover` są tu oczywiście w ogóle niepotrzebne).

Na temat tego, co jeszcze powinno się znajdować w arkuszu stylów dla druku, można by dużo napisać. Można np. zmienić wygląd nagłówków oraz usunąć komentarze, których może być bardzo dużo. Krótko mówiąc, elementy, których nie chcemy drukować, usuwamy, a pozostałe odpowiednio formatujemy. Ponadto można zmienić jednostkę rozmiaru pisma na punkty (`pt`) albo `em`, ponieważ są to typowe jednostki, jakimi posługują się drukarki. Dodatkowo warto zdefiniować sposoby łamania stron oraz style takich elementów jak cytaty blokowe i listy. Sam decydujesz, jak Twoja strona ma wyglądać po wydrukowaniu.



Pozostaje jeszcze tylko dołączyć nowy arkusz stylów do stron. Najlepiej użyć do tego celu funkcji `wp_ enqueue_style()` i haka `wp_enqueue_scripts` w pliku *functions.php*:

```
add_action( 'wp_enqueue_scripts', 'smashing_printstyle' );
function smashing_printstyle() {
    // Rejestracja arkusza stylów
    wp_register_style( 'printstyle', get_template_directory_uri() .
     '/print.css', array(), false, 'print' );
    // Dodanie arkusza stylów do kolejki
    wp_enqueue_style( 'printstyle' );
}
```

Najpierw rejestrujemy arkusz stylów za pomocą funkcji `wp_register_style()`, aby mieć do niego łatwy dostęp. Następnie dodajemy go do kolejki za pomocą funkcji `wp_enqueue_style()`, używając zarejestrowanego uchwytu. Dzięki użyciu haka `wp_enqueue_scripts` nie trzeba dodawać arkusza w pliku *header.php*.

---

*Jest kilka rozwiązań, które jeszcze bardziej ułatwiają drukowanie stron. Jedno z tych, które mi się podobają, ma nazwę PrintFriendly (*www.printfriendly.com*). Jest to usługa umożliwiająca dodanie przycisków drukowania na stronę. Jest nawet wtyczka ułatwiająca jej integrację z WordPressem. Jeśli chcesz zaoszczędzić na czasie, wypróbuj PrintFriendly, a zwłaszcza zajrzyj na stronę* www.printfriendly.com/button.

---

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# WIĘCEJ…

Rozszerzanie funkcjonalności witryny o niezbędne dodatki jest czymś naturalnym. W przypadku WordPressa takie rozszerzanie często oznacza promowanie artykułów i dodawanie elementów interaktywnych, takich jak formularz logowania.

Dla uzupełnienia w następnym rozdziale pokażę Ci kilka nietypowych rzeczy, jakie można robić w WordPressie. Bądź co bądź WordPressa można używać nie tylko do tworzenia zwykłych blogów i witryn. Rozdział 14. w całości poświęciłem na pokazanie fantastycznych możliwości, jakie stwarza ta znakomita platforma wydawnicza.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# 14 NIETYPOWE ZASTOSOWANIA WORDPRESSA

**WIESZ JUŻ, ŻE WORDPRESSA** można wykorzystać do budowy zarówno zwykłych blogów, jak i witryn z artykułami, takich jak statyczne serwisy, portale czasopism i gazet, oraz wszelkich innych witryn z publikacjami tekstu, grafiki i innych mediów.

Ale po co się ograniczać? WordPressa można używać do budowy jeszcze innych projektów — takich, których twórcy tej platformy na pewno nie mieli na myśli, rozpoczynając kiedyś swoją pracę. Na podstawie adaptacji przedstawionych w tym rozdziale chciałbym Ci pokazać pełny potencjał systemu. Dowiesz się, jak użyć go do budowy tablicy ogłoszeń o pracę i baz wiedzy, jak wykorzystać go do handlu elektronicznego i wiele więcej. Możliwości jest wiele i chciałbym, aby ten rozdział stał się dla Ciebie inspiracją do poszukiwania nowych.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PUBLIKOWANIE TREŚCI DOSTARCZANEJ PRZEZ UŻYTKOWNIKÓW

To, że WordPress jest znakomitym systemem CMS do budowy większych i mniejszych witryn, wcale nie oznacza, że nie można go użyć do innych celów. Można na przykład pozwolić użytkownikom na dodawanie nie tylko komentarzy, ale i innych rodzajów treści. Publikacje takie można wyraźnie oddzielić od własnych artykułów albo z nimi pomieszać.

Aby użytkownik mógł wysłać treść za pomocą standardowych funkcji WordPressa, musi być zarejestrowany w systemie. Popularnym motywem, w którym taką możliwość mamy, jest P2 dostępny pod adresem *http://wordpress.org/extend/themes/p2* i pokazany na rysunku 14.1. Inspiracją do jego powstania był Twitter, a wiele firm, m.in. Automattic, używa go do wewnętrznej komunikacji.

Rysunek 14.1. Motyw P2 był do niedawna używany w oficjalnym blogu zespołu programistów WordPressa WpDevel

Sposoby realizacji opisywanego pomysłu opiszę na podstawie analizy sytuacji, w których jego wdrożenie może być korzystne. Najpierw jednak rozważymy kwestię walorów użytkowych. Mimo że panel administracyjny WordPressa jest, ogólnie rzecz biorąc, prosty, niektórym użytkownikom może sprawiać kłopoty. Problem ten można rozwiązać, przenosząc funkcję dodawania wpisów na strony frontowe. Rozwiązanie takie można zastosować przynajmniej dla prostych treści.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## PRZYJMOWANIE WPISÓW OD UŻYTKOWNIKÓW

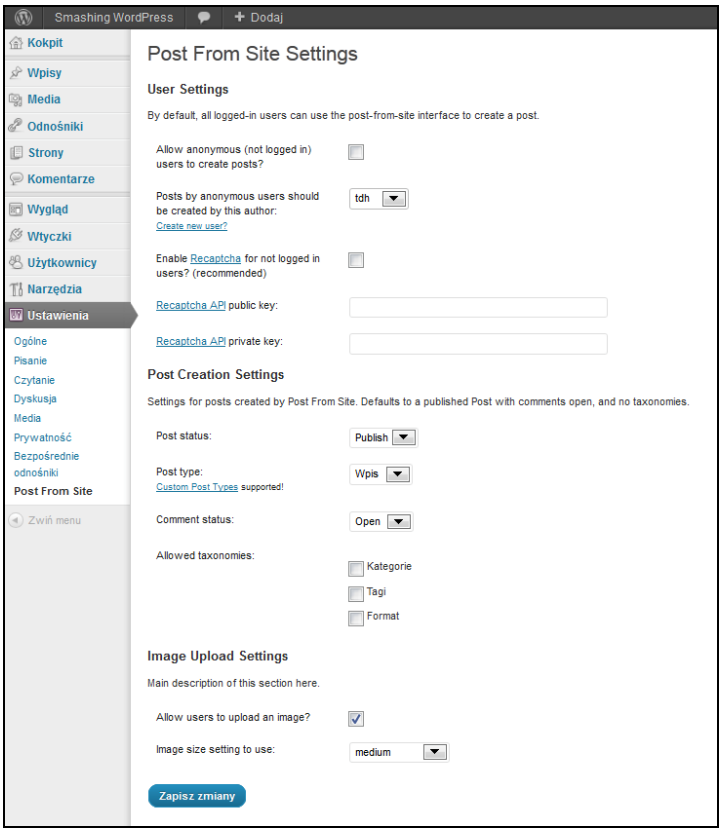
Jeśli chcesz zaprosić odwiedzających do publikowania wpisów na Twoim blogu i planujesz taką możliwość udostępniać przez dłuższy czas, to dobrym pomysłem będzie utworzenie specjalnej strony z formularzem do wysyłania wpisów dla gości. Dzięki wtyczkom jest to bardzo proste. Na przykład wtyczka Post From Site (*http://wordpress.org/extend/plugins/post-from-site*) umożliwia dodawanie prostych formularzy zamieszczania wpisów przy użyciu widżetu i skrótów kodowych. Na rysunku 14.2 pokazano formularz tej wtyczki utworzony na stronie za pomocą skrótu kodowego `[post-from-site]`.

Rysunek 14.2. Wtyczka Post From Site umożliwia dodawanie wpisów użytkownikom witryny

Wtyczka Post From Site jest doskonałym przykładem narzędzia umożliwiającego publikowanie gościnnych wpisów w WordPressie. Można jej używać od razu po zainstalowaniu, ale w razie potrzeby dostępnych jest też wiele ustawień, za pomocą których można ją dostosować do indywidualnych potrzeb. Możesz na przykład wybrać, aby wpisy były zapisywane jako szkice określonego typu z włączonymi komentarzami, albo zezwolić użytkownikowi na dodawanie obrazów. Najczęściej na publikację wpisów zezwala się zarejestrowanym użytkownikom, ale można też gościnne wpisy przypisywać do wybranego konta i publikować jako anonimowe. Na rysunku 14.3 pokazano stronę ustawień wtyczki, aby było wiadomo, o czym jest mowa. Jak widać, nie jest to nic skomplikowanego.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 14.3. Strona ustawień wtyczki Post From Site



Ta prosta wtyczka ma sporo opcji konfiguracyjnych, z których niektóre są dość zaawansowane. Mimo że czasami może być konieczne opracowanie własnego rozwiązania, wtyczki tego typu mogą być bardzo pomocne i to nie tylko w przyjmowaniu wpisów od użytkowników.

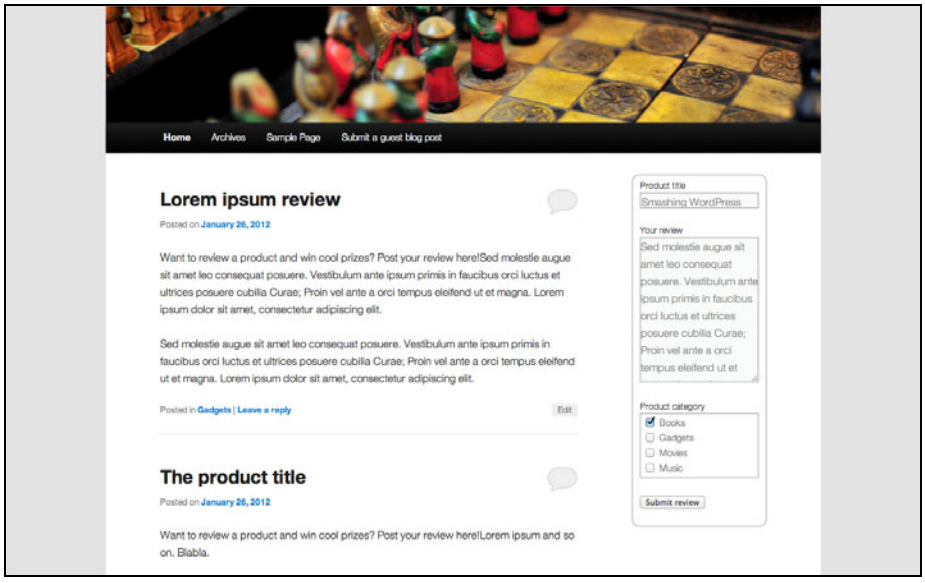
## OBSŁUGA WIADOMOŚCI I RECENZJI PUBLIKOWANYCH PRZEZ UŻYTKOWNIKÓW

Jednym z najczęstszych zastosowań funkcji wpisu gościnnego jest umożliwienie użytkownikom publikowania wiadomości i recenzji. Umożliwiając użytkownikom dodawanie tych treści, otrzymujemy dodatkowe artykuły do publikacji i wzbogacamy witrynę. W przypadku niektórych witryn może to być bardzo dobry pomysł. Na przykład w serwisie, w którym prezentowany jest jakiś produkt, korzystne może być umożliwienie publikowania recenzji, natomiast witryna poświęcona jakiemuś celebrycie powinna rozkwitnąć, gdy umożliwi się użytkownikom wysyłanie plotek o tej osobie. Możliwości jest wiele.

W zasadzie w wielu przypadkach może wystarczyć opisana wcześniej wtyczka Post From Site, ale czasami potrzebne jest coś bardziej rozbudowanego. Jeśli np. planujesz publikować recenzje czytelników, to zapewne zechcesz je w jakiś sposób zintegrować z systemem oceniania. Na poważnych stronach z wiadomościami potrzebny jest jakiś sposób oznaczania źródeł publikowanych wiadomości itd.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Gdyby nie brak wsparcia w nowszych wersjach WordPressa, dobrym rozwiązaniem mogłaby być wtyczka One Quick Post (*http://wordpress.org/extend/plugins/one-quick-post/*), dzięki której można utworzyć widżet pozwalający na dodawanie wpisów przez użytkowników. Jeśli zdecydujesz się na jej użycie, powinieneś w motywie zdefiniować obszar na widżety o odpowiedniej szerokości, aby można było wygodnie edytować wpis. Wszystko zależy od Twoich konkretnych potrzeb. Na rysunku 14.4 pokazano ten widżet w motywie Twenty Eleven. Jest on umieszczony w pasku bocznym i wygląda na trochę za bardzo ściśnięty.




Rysunek 14.4. Widżet wtyczki One Quick Post w motywie Twenty Eleven

Wtyczka One Quick Post ma o wiele więcej opcji niż Post From Site. Jest ich tak dużo, że aż trudno skonfigurować widżet (rysunek 14.5).

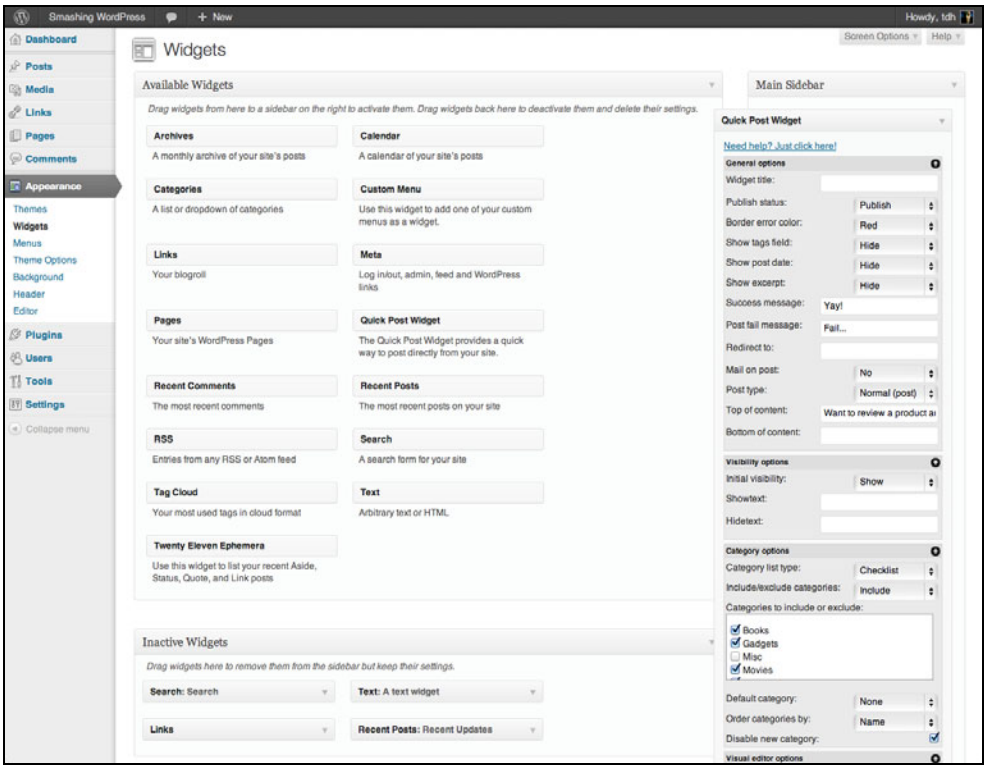
## TWORZENIE TABLICY OGŁOSZEŃ O PRACĘ

Do tej pory była mowa o umożliwianiu użytkownikom dodawania treści do czytania dla innych użytkowników. A co byś powiedział na coś całkiem innego, np. tablicę ogłoszeń o pracę? Każdy na pewno widział taki portal. Bez względu na rozmiar i kształt podstawowe funkcje w każdym tego typu serwisie są takie same. Użytkownicy mogą publikować ogłoszenia, czasami za drobną opłatą. Następnie inni użytkownicy szukający pracy lub pracownika czytają takie ogłoszenia i mogą na nie odpowiadać, również ewentualnie za drobną opłatą.

Taki portal można zbudować przy użyciu WordPressa i wcale nie jest to bardzo skomplikowane zadanie. Teoretycznie można by było użyć do tego celu zwykłych wpisów należących do kategorii **Szukam pracy** i **Dam pracę** oraz kilku tagów do precyzyjnej kontroli wyszukiwarki. Później wpisuje się tylko tytuł i treść oraz ewentualnie wypis i gotowe. Takie ogłoszenia można publikować na listach razem ze zwykłymi archiwami. Jedynym problemem jest wysyłanie ogłoszeń.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 14.5. Konfiguracja pojedynczego widżetu

Pewnie już się domyślasz, do czego zmierzam. Opisywaną funkcję można zrealizować za pomocą dowolnej z dwóch opisanych wcześniej wtyczek (chociaż One Quick Post byłaby bardziej odpowiednia). Twoim jedynym zadaniem jest zbudowanie witryny wokół tego centralnego punktu, tak aby wyświetlać treść w odpowiedni sposób.

Są jednak lepsze sposoby. Jeśli chcesz pobierać opłaty za publikowanie ogłoszeń, to musisz dodać oddzielne rozwiązanie realizacji płatności. Istnieją płatne wtyczki, które są do tego przeznaczone. Także wtyczki do publikowania ogłoszeń o pracę są dostępne — np. Your Classified Ads (*http://wordpress.org/extend/plugins/your-classified-ads*), której twórcą jest G. Breant (także autor wtyczki One Quick Post). Trudno jest udzielać dalszych rad, ponieważ wszystko zależy od Twojego pomysłu na działalność, ale na dobry początek możesz przynajmniej zadbać o to, aby użytkownicy mogli pisać ogłoszenia i zapisywać je jako szkice wpisów. Później możesz zażądać opłaty za publikację ogłoszenia, np. poprzez system PayPal, i po otrzymaniu należnej kwoty ręcznie zatwierdzić dany wpis. Nie jest to idealne rozwiązanie, ale tak też się da, a poza tym dzięki temu ma się możliwość kontrolowania jakości ogłoszeń publikowanych w portalu. Jeśli jednak ruch w witrynie będzie duży, musisz zautomatyzować proces, a to oznacza, że albo znajdziesz odpowiednią wtyczkę — istnieją dobre komercyjne wtyczki tego typu — albo napiszesz ją samodzielnie. Wszystko zależy od tego, czego dokładnie potrzebujesz i jakiego rodzaju portal prowadzisz. Może będziesz chciał pobierać opłaty tylko za wyróżnianie ogłoszeń. Wówczas sprzedawałbyś ogłoszenia w tradycyjny sposób, zamiast pobierać opłaty za każde ogłoszenie. Jest wiele możliwości.



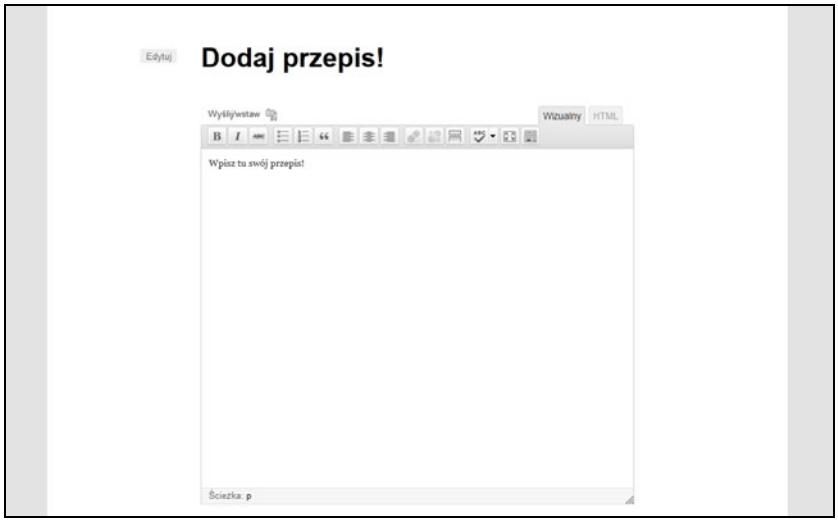
ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## FUNKCJA WP_EDITOR()

Dzięki funkcji `wp_editor()` możesz sprawić, że edytor wpisów WordPressa będzie widoczny na stronach witryny. Nie powinno się tego robić w każdym rodzaju serwisu, ale można tę opcję wykorzystać we wtyczce. Użycie funkcji `wp_editor()` jest bardzo łatwe. Wystarczy wywołać ją w dowolnym szablonie strony i już. Poniżej znajduje się przykładowy kod takiego wywołania:

```
wp_editor( 'Wpisz tu swój przepis!', 'submitrecipe' );
```

To spowoduje wyświetlenie edytora z domyślnymi ustawieniami. Pierwszy parametr określa tekst, który będzie domyślnie widoczny w polu edytora. Natomiast drugi parametr to identyfikator tego pola. Musi być napisany małymi literami oraz nie może zawierać łączników ani znaków podkreślenia, jeśli chcesz, aby TinyMCE (edytor wizualny wykorzystywany w WordPressie) dobrze działał. Efekt działania powyższego wiersza kodu pokazano na rysunku 14.6. Powinien wyglądać znajomo, ponieważ jest to zrzut ekranu z szablonu strony motywu Twenty Eleven.




Rysunek 14.6. Efekt działania funkcji wp_editor()

Niewiele można tu jednak zdziałać, ponieważ nie ma przycisku zatwierdzania. Aby rzeczywiście móc skorzystać z funkcji `wp_editor()`, należy dodać formularz i w jakiś sposób odebrać dane. Dlatego jeśli chcesz utworzyć naprawdę dobry formularz dodawania wpisów na stronę, lepiej utwórz wtyczkę z wykorzystaniem funkcji `wp_editor()` i pól `nonce` (*http://codex.wordpress.org/Wordpress_Nonce_Implementation*), aby dobrze zabezpieczyć formularz przed złośliwymi działaniami.

Zapewne najlepiej będzie, jeśli skorzystasz z jednej z opisanych wcześniej wtyczek, ale wiedząc, że istnieje taka funkcja jak `wp_editor()`, można zrobić wiele fajnych rzeczy. Więcej na temat tej funkcji i jej parametrów można przeczytać w dokumentacji na stronie *http://codex.wordpress.org/Function_Reference/wp_editor*.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### OSTATNIE SŁOWO NA TEMAT TREŚCI DODAWANEJ PRZEZ UŻYTKOWNIKÓW

Pozwolenie użytkownikom na dodawanie treści ma nie tylko zalety. Jeśli dopisze Ci szczęście, otrzymasz mnóstwo wartościowej treści, ale nie zawsze musi tak być. Równie dobrze może się okazać, że dostaniesz tylko spam albo bezwartościowe wpisy. Jeśli będziesz musiał moderować wpisy przed ich opublikowaniem, będziesz musiał poświęcić sporo czasu, podobnie jak jest z komentarzami.

Możliwość publikowania treści przez użytkowników może być bardzo przydatną funkcją witryny, trzeba tylko wiedzieć, jak nad nią zapanować i co robić, gdy pojawi się lawina wpisów.

## WORDPRESS JAKO BAZA WIEDZY

Z WordPressa mogą też korzystać firmy chcące publikować własne bazy wiedzy w stylu odpowiedzi na najczęściej zadawane pytania. Koncepcja jest bardzo prosta i opiera się na dodawaniu wpisów przez użytkowników oraz użyciu tagów i kategorii. Dzięki funkcji komentarzy każdy zgłoszony przez użytkownika temat może być przedyskutowany. A gdy uda się osiągnąć porozumienie, administrator może taki wpis przenieść do bazy wiedzy.



Oto podstawowe funkcje opisywanego systemu:

- Użytkownicy mogą zgłaszać problemy dzięki specjalnej wtyczce. Każdy zgłoszony problem jest wpisem.
- Są dwie główne kategorie: FAQ i Baza wiedzy.
- Użytkownicy mogą oznaczać wpisy (które są zgłaszanymi problemami) za pomocą tagów, aby ułatwić wyszukiwanie według słów kluczowych.
- Dostępna jest niestandardowa taksonomia dla bazy wiedzy.

Sposób działania witryny byłby następujący:

1. Użytkownik zgłasza problem za pomocą dostępnego na stronie formularza. Wpis dzięki specjalnej wtyczce zostaje zapisany jako szkic.
2. Administrator publikuje problem w kategorii FAQ.
3. Od tej pory inni mogą komentować ten wpis. Komentować może każdy, ale odpowiedzi administratora są wyróżniane. Jeśli wyłączy się komentowanie, należy umożliwić użytkownikowi, który dodał wpis, zażądanie otrzymania odpowiedzi bez konieczności zgłaszania kolejnych problemów.
4. Gdy problem zostanie rozwiązany, administrator dodaje wpis do kategorii Baza wiedzy.
5. Wszystkie wpisy w kategorii Baza wiedzy otrzymują odpowiednie tagi w niestandardowej taksonomii Tagi bazy wiedzy. Dzięki temu będzie można wyświetlać chmury tagów (i inne komponenty) skomponowane tylko z tagów bazy wiedzy.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Celem utworzenia taksonomii Tagi bazy wiedzy jest umożliwienie przeszukiwania rozwiązanych problemów. Taksonomia ta istnieje obok standardowych tagów, do których zaliczają się także wpisy FAQ, a więc nierozwiązane problemy. To również może być przydatne.

Dodatkowo można też przeglądać kategorie, dzięki czemu utworzenie dwóch sekcji — FAQ i Baza wiedzy — jest bardzo łatwe.

Jeśli masz już witrynę opartą na WordPressie i chcesz dodać taką funkcję, nie powinieneś do tego używać standardowych wpisów, tylko utworzyć nowy typ wpisów. Techniki tworzenia własnych typów wpisów są opisane w rozdziale 6., a także w dalszej części tego rozdziału, a więc nie powinieneś mieć problemu z dostosowaniem tego przykładu do własnych potrzeb.

## DODAWANIE FUNKCJI

Projekt motywu przedstawiony w tym podrozdziale będzie prosty i dlatego skupię się wyłącznie na opisie najważniejszych części. Techniki, które tu opisuję, można zastosować w dowolnym innym motywie. Najpierw musimy utworzyć nową taksonomię. Więcej na ten temat napisałem w rozdziale 6. Poniżej natomiast przedstawiam kod, który zostanie użyty. Można go umieścić w pliku *functions.php* motywu (nie ma znaczenia, czy jest to motyw nadrzędny, czy potomny), ale lepiej byłoby utworzyć wtyczkę, aby zachować ustawienia także po zmianie motywu. Więcej na temat przenośności danych napisałem w rozdziale 8.

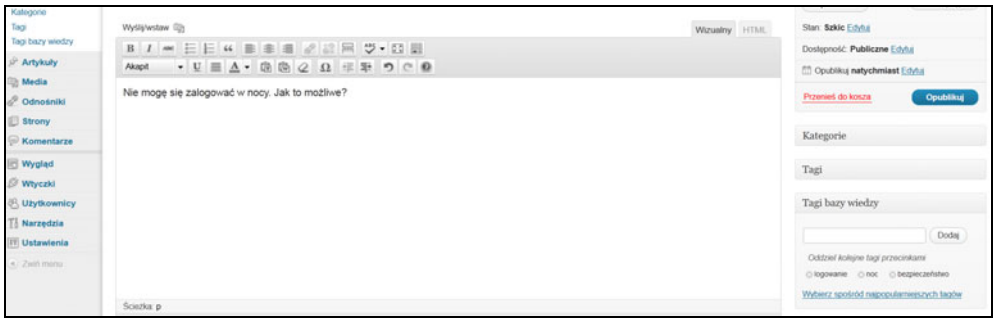


```
add_action( 'init', 'kbase', 0 );

function kbase() {
        register_taxonomy( 'knowledge_base', 'post',
            array(
                 'hierarchical' => false,
                 'labels' => array(
                 'name' => 'Tagi bazy wiedzy',
                 'singular_name' => 'Tagi bazy wiedzy',
                 'search_items' => 'Przeszukuj tagi Bazy wiedzy',
                 'popular_items' => 'Popularne tagi Bazy wiedzy',
                 'add_new_item' => 'Dodaj nowe tagi dla Bazy wiedzy'
            ),
              'query_var' => true,
              'rewrite' => true
        )
    );
}
```

To wystarczy. W panelu administracyjnym pojawi się nowy moduł tagów dla wpisów (rysunek 14.7). Tagi te będą przeznaczone tylko dla wpisów z bazy wiedzy.

Mając zdefiniowaną taksonomię, możemy zająć się odbieraniem zgłoszeń problemów. Kilka sposobów realizacji tego zadania zostało już opisanych, a więc nie będę tego już tu powtarzał. Możesz utworzyć plik szablonowy *home.php* do prezentowania najnowszych zgłoszeń i dodatków do bazy wiedzy (byłaby to po prostu lista najnowszych wpisów w kategorii) oraz wyświetlania formularza zgłaszania problemów. Osobiście dodałbym jeszcze wielkie

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 14.7. Moduł tagów dla bazy wiedzy

pole wyszukiwarki i chmurę tagów ukazującą najpopularniejsze tagi w taksonomii bazy wiedzy. Drugie z tych zadań można zrealizować, przekazując nazwę taksonomii do funkcji `wp_tag_cloud()`:

```
<?php
    $args = array(
         'taxonomy' => 'knowledge_base'
    )
    wp_tag_cloud( $args );
?>
```



Parametr `taxonomy` zawiera nazwę taksonomii, nie jej etykietę. Zdefiniuj odpowiedni styl dla chmury i już możesz wyświetlać na stronie listę najpopularniejszych słów kluczowych należących do bazy wiedzy, czyli dotyczących już rozwiązanych kwestii.

To w zasadzie wszystko. Po zaimplementowaniu części związanej ze zgłaszaniem problemów i taksonomią pozostaje jeszcze wyświetlenie najnowszych wpisów z odpowiednich kategorii oraz zachęcenie użytkowników do przeszukania bazy przed wysłaniem kolejnego zgłoszenia.

## DODATKOWE ULEPSZENIA

Utworzenie prostej bazy wiedzy w WordPressie jest bardzo łatwe, a w razie potrzeby można ją jeszcze rozbudować. Przede wszystkim trzeba sprawić, aby użytkownicy wiedzieli, że zanim wyślą zgłoszenie problemu, powinni najpierw przeszukać dostępne już informacje. W przeciwnym razie będziesz otrzymywać mnóstwo duplikatów zgłoszeń. Chcemy zachęcić użytkowników do przeszukiwania naszej bazy wiedzy, a więc musimy udostępnić im dobrą wyszukiwarkę. Niestety, jeśli WordPress ma jakiś słaby punkt, to jest nim właśnie funkcja wyszukiwania. Dlatego warto rozważyć możliwość skorzystania z jakiejś zewnętrznej usługi, np. Twoja wyszukiwarka Google (*www.google.com/cse/*) albo innej o podobnym zastosowaniu, którą można łatwo zintegrować z własną witryną. Jest też wiele wtyczek poprawiających działanie wyszukiwarki WordPressa, które również warto wypróbować. Jeśli jeszcze żadnej nie używasz, to zajrzyj do dodatku A, gdzie znajdziesz opis kilku, które mogą Ci się spodobać.

Oto kilka innych pomysłów na usprawnienie bazy wiedzy:

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Rejestracja i edytowanie — pozwól użytkownikom rejestrować się, aby mogli edytować wysyłane przez siebie pytania, tworzyć archiwa swoich wpisów itd. Możesz nawet w ten sposób trochę utrudnić wysyłanie spamu.
- Własny typ wpisów — jeśli bazę wiedzy chcesz wbudować w już istniejącą witrynę, to lepiej by było, gdybyś użył własnych typów wpisów zamiast standardowych. Ta metoda byłaby równie łatwa do zastosowania, o czym przekonasz się w podrozdziale „Tworzenie katalogu produktów".
- Ocenianie — pozwól użytkownikom oceniać Twoje odpowiedzi. W trybie FAT wpisy mogą być oceniane w komentarzach, a po przeniesieniu wpisu do bazy wiedzy użytkownicy mogą oceniać, w jakim stopniu rozwiązanie jest przydatne. W tym drugim przypadku potrzebna będzie wtyczka do oceniania wpisów. Kilka propozycji takich wtyczek znajduje się w dodatku A.
- Powiązane tematy — obok każdego wpisu wyświetl za pomocą specjalnej wtyczki inne, powiązane z nim wpisy. Użytkownik dzięki temu może odnaleźć szukane rozwiązanie.
- Usprawnienie wyszukiwarki — za pomocą JavaScriptu można ułatwić użytkownikom znajdowanie odpowiedzi na często zadawane pytania. Użyj pól wyszukiwania wyświetlających propozycje typu „Czy miałeś na myśli XYZ", gdy użytkownik wpisze np. „XPT".
- Usprawnienia strony głównej — im lepiej wyróżnisz rozwiązania problemów, tym szybciej użytkownicy je znajdą, a ponieważ cały serwis właśnie do tego służy, strona główna powinna być jak najlepiej dostosowana do potrzeb użytkowników.
- Subskrypcje i powiadomienia — zaoferuj możliwość subskrybowania informacji o nowościach i rozwiązanych problemach. Pomysł ten można zrealizować na wiele sposobów, a ponieważ WordPress tworzy kanał RSS praktycznie dla wszystkiego, jestem pewien, że to jest dobra droga. Upewnij się, że użytkownicy dowiedzą się, jakie są możliwości subskrypcji.



Baza wiedzy to jeszcze jeden z wielu pomysłów wykorzystania WordPressa. Innym jest prowadzenie sklepu internetowego.

## WORDPRESS I HANDEL ELEKTRONICZNY

Nie ma wątpliwości, że WordPressa można używać jako platformy sprzedażowej. W najprostszym przypadku można prowadzić tradycyjnego bloga i za jego pośrednictwem dokonywać sprzedaży produktów. Można publikować **linki partnerskie**, czyli specjalne łącza, np. do sklepu Amazon prowadzące do wybranych produktów. Gdy ktoś dokona zakupu, korzystając z takiego odnośnika, my otrzymujemy prowizję. Jeśli kupiłeś tę książkę, klikając łącze na jednej z moich stron internetowych, to zarobiłem dzięki temu parę dodatkowych groszy. Dzięki!

Jednak mówiąc o handlu elektronicznym, zwykle mamy na myśli coś „większego" niż linki programów partnerskich — koszyki na zakupy, cyfrową dystrybucję, płatności PayPal itd.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Wszystko to można mieć w WordPressie. W dodatku A opisałem kilka wtyczek ułatwiających implementację narzędzi handlowych, od prostych dodatków po całe koszyki na zakupy.

## PROWADZENIE SKLEPU OPARTEGO NA WORDPRESSIE

Zastanawiasz się, czy da się prowadzić sklep oparty na WordPressie? Tak, **jest to możliwe**, zwłaszcza gdy ma się niezbyt dużo produktów.

Uruchomienie sklepu opartego na WordPressie nie jest łatwe, ale to nie znaczy, że od razu musisz się poddać. Nie ma powodu, dla którego miałoby to nie działać, jeśli tylko jesteś gotów zainstalować trochę wtyczek, wprowadzić do użytku własne pola oraz znaleźć sposób na połączenie swojego koszyka z systemem PayPal lub innym narzędziem obsługi płatności.

Nie brzmi zachęcająco? Spokojnie, nie jest tak źle, jak się wydaje. Są wtyczki, które większość pracy wykonają za Ciebie. Należy jednak pamiętać, że w porównaniu z typowymi systemami e-commerce większość tych wtyczek ma ubogie możliwości. Zanim zdecydujesz się na poprowadzenie takiej działalności przy użyciu WordPressa, dobrze wszystko przemyśl. Możliwe, że lepszym rozwiązaniem będzie jakiś specjalistyczny darmowy system do handlu elektronicznego. Przemyśl wszystkie za i przeciw i dopiero wtedy podejmij decyzję.



Zaletą WordPressa jest to, że najprawdopodobniej masz już działającą witrynę opartą na tym systemie i wystarczyłoby tylko dodać do niej jeden z dostępnych gotowych skryptów koszyków na zakupy. Większość z nich powinna działać bez żadnych problemów, a Tobie pozostałoby tylko wymyślenie sposobu na pobieranie opłat. Na szczęście firmy takie jak PayPal zadbały, aby było to łatwe, dzięki czemu możesz bez problemu zacząć zarabiać, sprzedając towary w swojej witrynie. Jeśli zatem szukasz lekkiego rozwiązania i nie chcesz używać skomplikowanych specjalistycznych platform handlowych, to WordPress może być dla Ciebie dobrym rozwiązaniem.

I jeszcze jedna uwaga. Decydując się na uruchomienie sklepu, dobrze się zastanów, w co zaczynasz się bawić. Handel to całkiem inna działka niż publikowanie treści.

## SPRZEDAWANIE PRODUKTÓW CYFROWYCH

Produkty cyfrowe, np. książki elektroniczne, to całkiem inna bajka. Nic nie stoi na przeszkodzie, abyś zaimplementował usługę płatności internetowych za wybrany plik i udostępniał go po otrzymaniu zapłaty. W istocie ma to niewiele wspólnego z WordPressem, ponieważ cała transakcja polega na sprawdzeniu, czy użytkownik dokonał płatności, a następnie przekazaniu mu pliku. Integracja takiego rozwiązania z blogiem jest bardzo łatwa, jeśli skorzysta się z jakiejś zewnętrznej usługi, np. E-junkie (*www.e-junkie.com*) — system ten pobiera część Twoich zarobków na obsługę płatności i dostarczenie pliku. Wystarczy tylko podać odnośnik, a dostawca załatwia pozostałe sprawy — coś podobnego do programów partnerskich, tylko otrzymujesz większą część zapłaty, ponieważ sprzedawany jest Twój produkt.

Rozwiązanie takie można też zaimplementować samodzielnie przy użyciu wtyczek lub skryptów. Nie da się jednak pominąć drobnego szczegółu, jakim jest pobieranie zapłaty za produkty,

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

a to oznacza, że musisz skorzystać z jakiejś usługi obsługi płatności. Teoretycznie tym również można zająć się samodzielnie, ale nie polecam takiego rozwiązania, choćby dlatego, że klienci czują się o wiele bardziej bezpiecznie, jeśli ich transakcję obsługuje znana im organizacja.

Produkty cyfrowe można sprzedawać w zasadzie w każdym rodzaju witryny. Może WordPress nie jest najdoskonalszą platformą handlową, ale do sprzedawania książek elektronicznych, plików MP3, projektów itp. nadaje się znakomicie.

### BUDOWA SKLEPU

Zdecydowałeś się użyć WordPressa jako platformy sprzedażowej. Nieważne czy ma to być duży, czy mały sklep, teraz musisz znaleźć sposób na takie jego skonfigurowanie, aby jak najlepiej działał i był jak najłatwiejszy w obsłudze.

Podczas gdy możesz użyć wpisów lub stron, myślę, że powinieneś zastanowić się nad użyciem własnych typów wpisów. Oddzielenie produktów przeznaczonych na sprzedaż od reszty infrastruktury witryny może nie być konieczne, ale jeśli jest korzystne lub może być takie w przyszłości, to najlepszym rozwiązaniem jest użycie własnych typów wpisów. Tematem następnego podrozdziału jest używanie własnych typów wpisów do prezentowania produktów.



## TWORZENIE KATALOGU PRODUKTÓW

Dzięki własnym typom wpisów utworzenie katalogu produktów (a tak naprawdę dowolnego katalogu, np. nieruchomości, osób itd.) jest banalnie proste. Należy utworzyć nowy typ wpisów, aby oddzielić produkty na sprzedaż od reszty treści i to niezależnie od tego, czy tych produktów jest 10, czy 500.

Oczywiście każdy produkt wymaga zapisania innych informacji, dlatego poniższą listę należy traktować tylko jako prosty przykład:

- Utwórz własny typ wpisów o nazwie **Książki**.
- Każdy produkt będzie książką, a informacje o nim będą przechowywane w postaci wpisu typu książkowego.
- W tym przykładzie obsługiwane są kategorie, które będą używane do sortowania książek.
- Nie włączaj obsługi tagów.

Tagi zostały wykluczone dla uproszczenia. W większości przypadków najprawdopodobniej ich włączenie byłoby pożądane, ponieważ dzięki nim można by było sortować wpisy według wydawców itp.

A tak przy okazji: dlaczego książki? Pomyślałem sobie, że mógłbym przedstawione tu rozwiązanie wykorzystać na swojej stronie *http://tdh.me*. Posługiwanie się realnym

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

przykładem zawsze jest dobrym pomysłem. Przedstawiony kod, podobnie jak większość innych moich treści, możesz wykorzystać do własnych celów.

## TWORZENIE TYPU WPISÓW DLA KSIĄŻEK

Szczegółowy opis własnych typów wpisów znajduje się w rozdziałach 6. i 9., a więc jeśli potrzebujesz odświeżenia pamięci, możesz do nich powtórnie zajrzeć.

Aby utworzyć typ wpisów o nazwie Książki, dodaj poniższy kod do pliku *functions.php* swojego motywu albo zapisz go w postaci wtyczki. Drugie rozwiązanie jest lepsze, ponieważ nie powoduje problemów z przenoszeniem danych. Jeśli potrzebujesz więcej informacji, przeczytaj rozdziały 6. i 7.

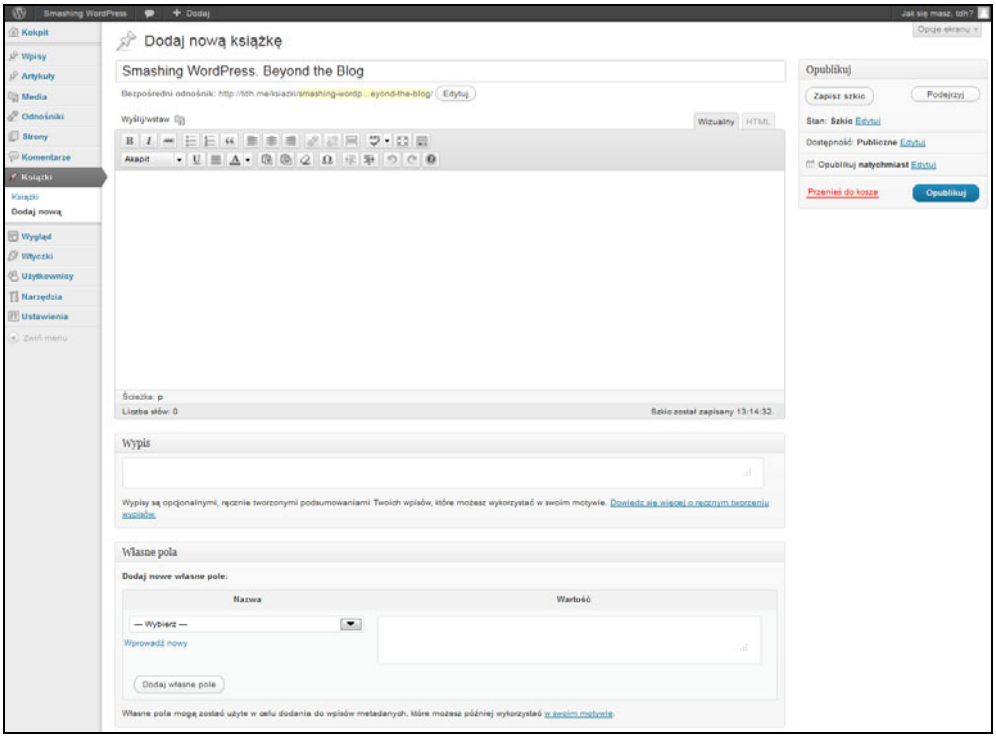
```
register_post_type( 'books', array(
    'label' => 'Książki',
    'labels' => array(
        'singular_name' => 'Książka',
        'add_new_item' => 'Dodaj nową książkę',
        'edit_item' => 'Edytuj książkę',
        'new_item' => 'Nowa książka',
        'view_item' => 'Zobacz książkę',
        'search_items' => 'Przeszukuj książki',
        'not_found' => 'Brak książek',
        'not_found_in_trash' => 'Nie znaleziono żadnej książki w koszu',
    ),
    'public' => true,
    'show_ui' => true,
    'capability_type' => 'post',
    'taxonomies' => array( 'category' ),
    'has_archive' => true,
    'rewrite' => array(
        'slug' => 'ksiazki'
    ),
    'query_var' => false,
    'supports' => array(
        'title',
        'editor',
        'author',
        'thumbnail',
        'excerpt',
        'custom-fields',
        'comments'
    )
) );
```

Ten kod jest bardzo prosty. Rejestruje typ wpisów `books` i definiuje dla niego różne etykiety, które będą widoczne w panelu administracyjnym, jak pokazano na rysunku 14.8. Tablica w elemencie `labels` służy do określania napisów, które mają być wyświetlane w różnych miejscach panelu administracyjnego WordPressa. Niektóre ustawienia mają wartości domyślne, ale lepiej jest wszystko zdefiniować samodzielnie, aby całość dobrze wyglądała.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Parametr `public` sprawia, że typ wpisów jest publiczny. Parametr `capability_type` ustawiony na `post` sprawia, że tworzony typ wpisów ma podobne działanie do zwykłych wpisów. Dodatkowo nasz typ obsługuje kategorie, strony archiwalne oraz przepisywanie URL-i pozwalające tworzyć ładne adresy (ostatnie ustawienie jest domyślne, ale i tak zazwyczaj je dodaję). Na końcu znajduje się parametr `supports` zawierający tablicę funkcji obsługiwanych przez typ wpisów w panelu administracyjnym (w większości są to różne moduły). Resztę ustawień pozostawiamy z wartościami domyślnymi (jeśli potrzebujesz odświeżenia pamięci, to zajrzyj do rozdziału 9.).




Rysunek 14.8. Tworzenie nowego wpisu typu Książka

Teraz musisz już tylko dodać wpisy dla wszystkich swoich książek i gotowe!

## TWORZENIE STRONY DLA KSIĄŻEK

Są dwa sposoby na utworzenie strony zawierającej listę wszystkich wpisów typu `books`. Jednym z nich jest utworzenie zwykłej strony w WordPressie i wybranie dla niej pliku szablonowego zawierającego odpowiednią pętlę. Drugie rozwiązanie polega na użyciu pliku szablonowego dla typu wpisów. Pokażę, jak zrealizować drugi z wymienionych pomysłów, czyli utworzę plik *archive-books.php* (plik *archive-[typ wpisu].php* ma pierwszeństwo przed plikiem *archive.php*). Równie dobrze mógłbym też użyć kilku tagów warunkowych w pliku *archive.php* albo *index.php*, ale tak będzie prościej.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Najpierw jednak jeszcze na chwilę zatrzymam się przy pewnym często spotykanym problemie związanym z własnymi typami wpisów, zwłaszcza tworzonymi przy użyciu wtyczek (opis kilku znajduje się w dodatku A). Wpisy własnych typów mogą nie wyświetlać się wraz ze zwykłymi wpisami. Jest tak, ponieważ WordPress domyślnie nie uwzględnia niestandardowych typów wpisów w pętli. Jeśli chcesz wyświetlić wpisy określonego typu, np. `books`, w szablonie, musisz użyć instrukcji `query_posts( 'post_type=books' )`, która nakazuje WordPressowi ich pobranie. Oczywiście można mieszać typy wpisów. W tym celu należy je wymienić w tablicy, np. poniższa instrukcja powoduje pobranie zwykłych wpisów i książek:

```
query_posts( 'post_type => array( 'post', 'books' ) ' )
```

Można też używać klasy `WP_Query` (pokazanej w następnym przykładzie) do tego samego celu co funkcji `query_posts()`. Więcej informacji na temat tej funkcji znajduje się w dokumentacji na stronie *http://codex.wordpress.org/Function_Reference/query_posts*.

Wracamy do archiwum książek. Wyobraźmy sobie, że nie mamy ogromnej ilości książek (ja naprawdę nie mam), i dlatego każdej książce (będącej wpisem zdefiniowanego przez nas typu) możemy poświęcić sporo miejsca na stronie. Po lewej stronie będzie znajdować się okładka, a po prawej krótki tekst zwięźle opisujący książkę. Okładkę będziemy pobierać dzięki funkcji dodawania ikon wpisów. Aby to było możliwe, musimy tę funkcję włączyć, dodając do pliku *functions.php* następujący kod:



```
// Dodanie obsługi ikon wpisów
add_theme_support( 'post-thumbnails' );
// Dodanie rozmiarów obrazów
add_image_size( 'books-small', 150, 0 );
```

Pierwsza instrukcja zawiera wywołanie funkcji `add_theme_support()` włączające obsługę ikon wpisów. Można to zrobić tylko dla wybranego typu wpisów, ale zazwyczaj ikon używa się w różnych typach wpisów na różne sposoby, a więc najlepiej jest je włączyć globalnie. W drugim wierszu zdefiniowany został nowy rozmiar obrazów za pomocą funkcji `add_image_size()`. Jego nazwa to `books-small`, a szerokość została ustawiona na 150 pikseli. Nie definiujemy wysokości ani nie włączamy obcinania.

Teraz możemy utworzyć plik *archive-books.php*. Zwróć uwagę, że kod tego pliku jest przygotowany do internacjonalizacji, o czym świadczy obecność konstrukcji `_e()` z domeną tekstową `tdh7`. Więcej na temat internacjonalizacji napisałem w rozdziale 6.

```
<?php get_header(); ?>
    <div id="inner-site">
        <div id="content" class="entry-list">
            <ul id="above-listings">
            <?php dynamic_sidebar('listings'); ?>
            </ul>
            <div id="section-head">
                <p><?php _e('I\'m a published author, and these are my books. Just
                ↪the English ones though, no Swedish works here I\'m afraid.',
                ↪'tdh7');?></p>
            </div>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
    <?php
        // Podstawowa pętla
        while ( have_posts() ) : the_post(); ?>

        <div id="post-<?php the_ID(); ?>"
        ↪<?php post_class('books-archive'); ?>>
            <div class="books-cover">
                <a href="<?php the_permalink(); ?>"
                ↪class="books-cover-link">
                <?php
                    // Czy wpis ma ikonę?
                    if ( has_post_thumbnail() ) {
                        // Tak, więc jej użyjemy
                        the_post_thumbnail( 'books-small' );
                    }
                    // Nie, więc użyjemy domyślnej
                    else { ?>
                        <img src="<?php echo
                        get_template_directory_uri(); ?>/
                        ↪img/books-nocoveryet.png" width="150"
                        ↪height="190" alt="Brak okładki" />
                    <?php }
                    // Wszystko zrobione
                ?>
                </a>
            </div>
            <div class="books-content">
                <h2 class="entry-title books-title">
                    <a href="<?php the_permalink(); ?>" title="<?php
                     ↪the_title_attribute(); ?>" rel="bookmark">
                        <?php the_title(); ?>
                    </a>
                </h2>
                <div class="entry-summary entry-summary-books">
                    <?php the_excerpt(); ?>
                </div>
            </div>
        </div>

    <?php
        // Koniec pętli
        endwhile; ?>
</div>
<?php
    // Prawa kolumna
    get_sidebar();

    // Jeśli to możliwe, wyświetlamy nawigację na dole
    if ( $wp_query->max_num_pages > 1 ) :
?>
    <div class="navigate">
        <div class="nav-left">
            <?php
               next_posts_link(__ ('&larr; Browse older content', 'tdh7') );
```



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
                    ?>
                </div>
                <div class="nav-right">
                    <?php
                        previous_posts_link(__ ('Browse more recent content
                        ↪&rarr;', 'tdh7') );
                    ?>
                </div>
            </div>
        <?php endif; ?>
    </div>

<?php get_footer(); ?>
```

Jest to zwyczajna pętla, której kod powinieneś znać już na pamięć (ale gdybyś jednak potrzebował odświeżenia pamięci, cofnij się do rozdziałów 3. i 4.). W stosunku do wcześniej prezentowanych pętli są tylko dwie różnice, o których warto wspomnieć. Pierwsza z nich to funkcja `post_class()` z parametrem `books-archive`. Parametr ten to nazwa klasy, która ma zostać dodana do danych wyjściowych funkcji.

Drugą różnicę stanowi instrukcja sprawdzająca, czy dodana jest okładka książki. Sprawdzamy, czy wpis ma ikonę, a jeśli tak, to ją wyświetlamy. Jeśli natomiast ikony nie ma, wyświetlimy domyślny obraz, ponieważ wpis bez grafiki dziwnie by wyglądał. Mimo że okładka książki jest często opracowywana dopiero po powstaniu tekstu, chcielibyśmy, aby wszystko dobrze wyglądało.



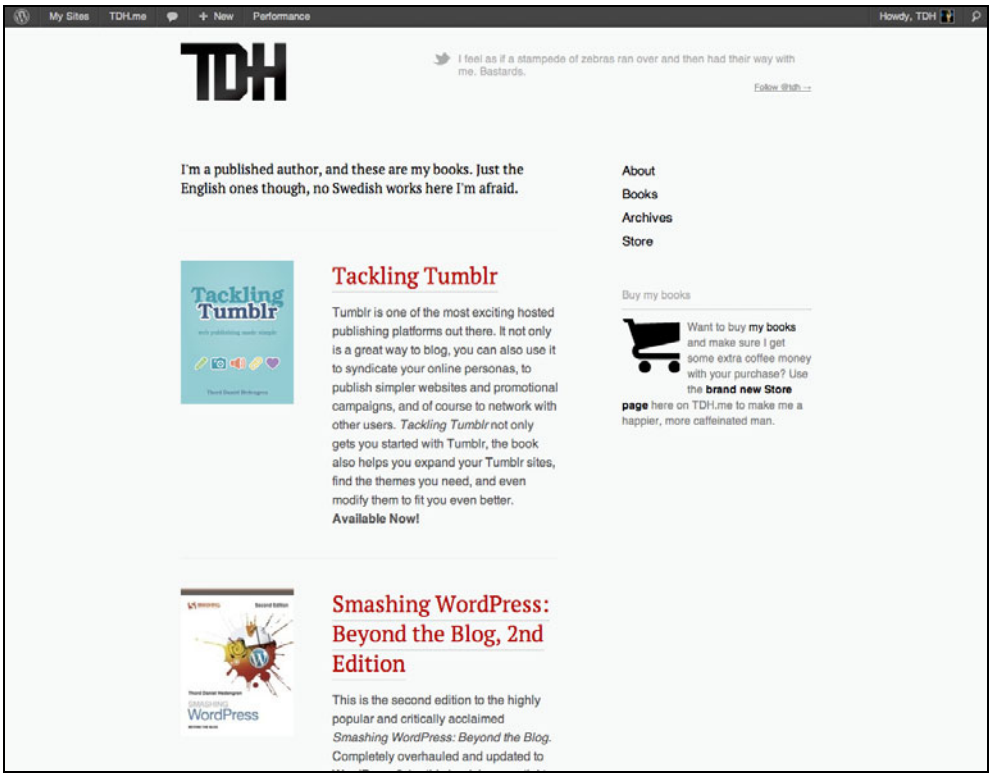
Napisanie arkuszy stylów będzie proste. Element `div.books` pełni dla książek tę samą rolę co `div.post` dla zwykłych wpisów. Dodatkowo okładkę i opis książki również umieszczono w elementach `div`.Warto przy tej okazji zaznaczyć, że dla własnych typów wpisów funkcja `post_class()` nie zwraca zwykłego zestawu klas, a jedynie jedną klasę o nazwie odpowiadającej nazwie typu wpisów. Dlatego w tym przypadku funkcja ta zwróci tylko klasę `books`. Jeśli chcesz, aby wpisy książek wyglądały tak samo jak zwykłe wpisy, to przypisz tej klasie te same style co zwykłym wpisom. W tym przypadku wystarczy tylko określić marginesy itp.

To w zasadzie wszystko, jeśli chodzi o nowy typ wpisów. Strona książek jest dodana, jak widać na rysunku 14.9, i zawiera listę książek reprezentowanych jako własny typ wpisów. Teraz stronę tę można łatwo aktualizować bez konieczności kłopotliwego edytowania stron — wystarczy po prostu dodać nowy wpis typu książkowego.

## PROMOWANIE PRODUKTÓW

Wpisy niestandardowego typu domyślnie nie są uwzględniane w zwykłej pętli, ale łatwo można to zmienić przy użyciu funkcji `query_posts()` lub klasy `WP_Query` (jak może pamiętasz, mają one takie same zastosowanie). Chociaż nie wydaje mi się, aby to miało sens. Powinniśmy natomiast jakoś promować nasze produkty. W następnych podrozdziałach przedstawiłem kilka pomysłów, jak to robić.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 14.9. Strona książek, prosta i gotowa do użytku



## Dodanie pętli

Najprostszym sposobem na promowanie produktów jest dodanie do motywu pętli wyświetlającej wpisy naszego typu. Jej kod może wyglądać następująco:

```
<ul>
<?php
    $args = array(
         'post_type' => 'books',
         'post_status' => 'publish',
         'posts_per_page' => -1'
    )
    // Pętla
    $my_query = new WP_Query( $args );
    while ( $my_query->have_posts() ) : $my_query->the_post(); ?>
    <li>
        <a href="<?php the_permalink(); ?>" title="<?php
          the_title_attribute();?>">
            <?php the_title();?>
        </a>
    </li>
<?php endwhile; ?>
</ul>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Bez wątpienia rozpoznajesz ten kod. Jest to pętla zwracająca wszystkie wpisy typy `books` z tytułem jako łączem (tak, wszystkie — dlatego właśnie parametr `posts_per_page` w `WP_Query` został ustawiony na `-1`).

Potrzebujesz czegoś bardziej wyszukanego? Możesz np. utworzyć za pomocą JavaScriptu atrakcyjny suwak do prezentowania swoich książek. Jest też sporo wtyczek do promowania wpisów, których możesz użyć, jeśli uda Ci się je zmusić do działania z własnymi typami wpisów. Są też takie, które przyjmują wszystko, co się im poda. Jak to zwykle bywa, ograniczeniem są wyłącznie Twoje umiejętności i wyobraźnia.

### Użycie grafik i łączy

Jeśli masz ciekawe zdjęcia produktów albo inne materiały promocyjne, to możesz je wykorzystać, aby zachęcić odwiedzających do zainteresowania się Twoją ofertą. Najprostszym sposobem na przyciągnięcie uwagi do produktów jest umieszczenie będących łączami grafik w obszarach widżetów (zwykle w pasku bocznym). W ten sposób można skusić wielu użytkowników do wejścia na strony produktów.

W naszym przykładzie grafiki są używane przede wszystkim we wpisach, ale również i w obszarach widżetów. Rozwiązanie to jest najlepsze dla niewielkich katalogów produktów, gdzie jest niewiele aktualizacji i dodatków i nie trzeba wyświetlać zbyt wielu produktów naraz. Jeśli masz większe wymagania, lepszym rozwiązaniem będzie zapewne użycie pętli.



### Prosty dodatek do menu

Czasami najprostsze rozwiązanie jest zarazem najbardziej oczywiste. Element w menu prowadzący do produktów może nie jest tak atrakcyjny wizualnie jak grafika ani tak przyciągający uwagi jak pętla wyświetlającą najnowsze produkty, ale również może być skuteczny, zwłaszcza że wielu użytkowników czegoś takiego się spodziewa. Jeśli utworzyłeś stronę z produktami, to koniecznie dodaj do menu prowadzący do niej odnośnik.

## BLOG ZE SMAKIEM, CZYLI WITRYNA Z PRZEPISAMI

Niezależnie od tego, czy chcesz publikować przepisy tylko dla siebie, pokazać je kilku znajomym, czy też masz zamiar prowadzić komercyjny portal z przepisami, WordPress jest platformą, która może się nadawać do realizacji tego celu. Myślę, że najlepiej będzie rozpocząć pracę od razu i zobaczymy, co nam wyjdzie.

Najpierw należy określić podstawowy zestaw funkcji witryny. Na razie nie przejmuj się kwestiami wizualnymi, ponieważ rozwiązanie przedstawione w tym rozdziale będzie na tyle elastyczne, że można je będzie zastosować w dowolnym motywie. Całość jest również bardzo prosta. Trzeba utworzyć osobny typ wpisów, aby oddzielić przepisy od zwykłych wpisów i stron. Potrzebne też będą własne taksonomie do oznaczania różnych rodzajów przepisów i składników. I to w zasadzie wszystko, co będzie potrzebne. Bierzemy się więc do pracy!

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## PRZYSTAWKA, CZYLI WYBÓR MOTYWU

Mamy dwie możliwości do wyboru. Albo utworzymy całkiem nowy motyw (na bazie szkieletu lub innego motywu, który nam się podoba), albo wybierzemy jakiś gotowy motyw i utworzymy jego motyw potomny. Dla uproszczenia w tym przykładzie utworzymy potomka motywu Twenty Eleven.

Nasz motyw potomny o nazwie **Tasty Eleven**. Poniżej znajduje się pierwsza część jego pliku style.css:

```
/*
Theme Name: Tasty Eleven
Theme URI: http://tdh.me/wordpress/tasty-eleven/
Author: Thord Daniel Hedengren
Author URI: http://tdh.me
Description: Motyw potomny motywu Twenty Eleven z funkcją publikowania przepisów.
Version: 1.0
License: GNU General Public License
License URI: license.txt
Template: twentyeleven
*/
@import url('../twentyeleven/style.css');
```

Jest to typowy kod pliku CSS motywu. Zwróć uwagę na instrukcję importu arkusza stylów z motywu Twenty Eleven. Własne style możesz wpisać pod tą instrukcją, ale ja w dalszej części tekstu nie będę tego opisywał.



Potrzebny Ci też będzie plik *functions.php*. Utwórz go i możemy przejść do najciekawszych rzeczy.

## DANIE GŁÓWNE — PRZEPISY JAKO TYP WPISÓW

Utworzenie własnego typu wpisów, jak zapewne pamiętasz, jest bardzo łatwe. W przypadku przepisów chodzi nam o to, aby oddzielić je od innych rodzajów treści publikowanych w witrynie. Zwykłe wpisy zarezerwujemy dla wiadomości albo bloga, a statyczne strony dla jeszcze czegoś innego. Oczywiście moglibyśmy sobie poradzić także bez dodawania własnych typów wpisów, np. używając zwykłych wpisów i własnych taksonomii, ale takie rozwiązanie byłoby mniej wygodne.

Zaczniemy od utworzenia typu wpisów do receptur. Nazwiemy go **Receptury**, zdefiniujemy mu odpowiednie adresy oraz sprawimy, żeby dobrze prezentował się w panelu administracyjnym. Poniższy kod jest bardzo podobny do kodu z przykładu o książkach i dlatego nie będę go szczegółowo opisywał. Przypomnę, że można go zapisać albo jako wtyczkę, aby był łatwiejszy do przenoszenia, albo w pliku *functions.php*:

```
// Typ wpisów Receptury
register_post_type( 'recipes', array(
    'label' => 'Receptury',
    'labels' => array(
        'singular_name' => 'Receptura',
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            'add_new_item' => 'Dodaj nową recepturę',
            'add_new' => 'Dodaj nową',
            'edit_item' => 'Edytuj recepturę',
            'new_item' => 'Nowa receptura',
            'view_item' => 'Zobacz recepturę',
            'search_items' => 'Przeszukuj receptury',
            'not_found' => 'Nie znaleziono żadnej receptury',
            'not_found_in_trash' => 'Nie znaleziono żadnej receptury w koszu',
        ),
        'public' => true,
        'show_ui' => true,
        'capability_type' => 'post',
        'has_archive' => true,
        'rewrite' => array(
            'slug' => 'receptury'
        ),
        'query_var' => false,
        'supports' => array(
            'title',
            'editor',
            'author',
            'thumbnail',
            'excerpt',
            'comments'
        )
) );
```



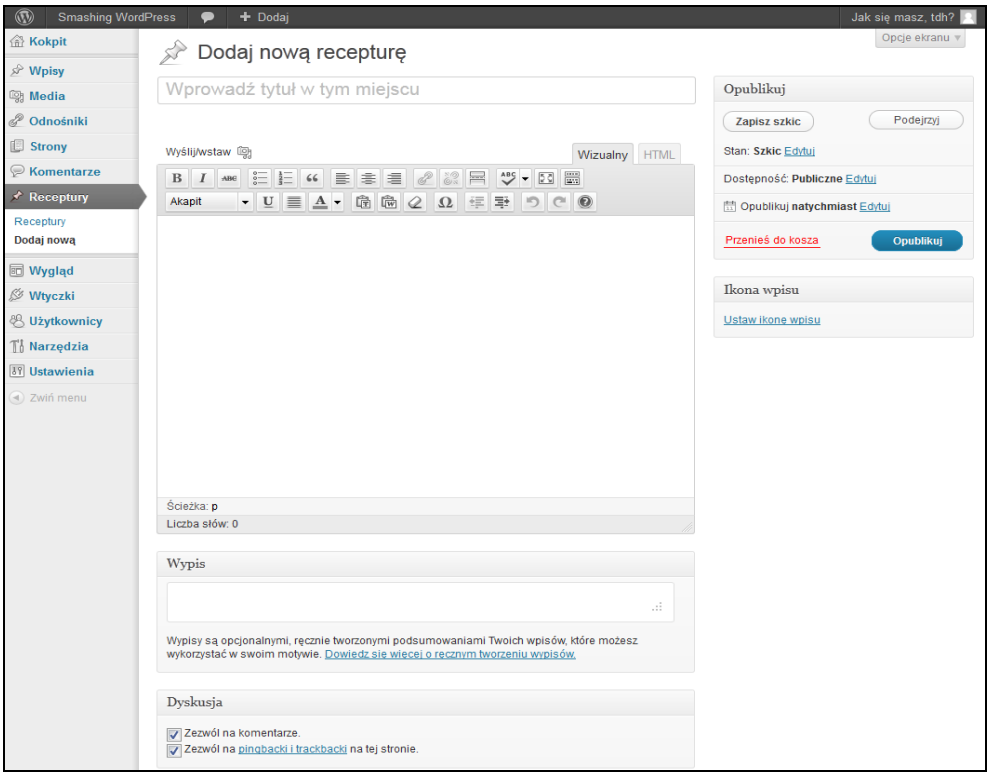
Zapewne zauważyłeś brak taksonomii. Można dodać ich obsługę w tym kodzie, ale nie jest to konieczne, ponieważ w definicji taksonomii można zdefiniować, gdzie dana taksonomia ma się pojawiać. Wkrótce do tego dojdziemy. Na rysunku 14.10 pokazane jest, jak wygląda nasz nowy typ wpisów w panelu administracyjnym.

Własne typy wpisów nie są uwzględniane w zwykłej pętli i musimy coś z tym zrobić. Najpierw utworzymy stronę *archive-recipes.php* do wyświetlania receptur. Jest to, jak zapewne pamiętasz, plik szablonowy, który będzie używany jako archiwum receptur. Inaczej mówiąc, plik ten będzie nam służył do kontrolowania sekcji receptur w witrynie. Jego konkretna zawartość zależy od motywu, ale ogólnie rzecz biorąc, powinien zawierać pętlę. Nie przedstawiam jej po raz kolejny.

## DESER — WŁASNE TAKSONOMIE

Mimo że bardzo dobrze poradzilibyśmy sobie, używając tylko standardowych kategorii i tagów, własne taksonomie dają nam większe możliwości. Utworzymy taksonomię, która będzie działała tak samo jak tagi, ale będzie przeznaczona do określania składników. Dzięki temu będzie można łatwo oznakować każdą recepturę składnikami, które są w niej potrzebne. Dodatkowo automatycznie zostaną utworzone archiwa wszystkich składników i jeśli będziemy chcieli ugotować coś zawierającego pomidory, to wystarczy, że zajrzymy do archiwum tagu o takiej nazwie. Ponadto będzie można tworzyć połączone archiwa różnych składników, aby każdy użytkownik mógł znaleźć dokładnie to, czego potrzebuje. Jeśli to możliwe, różne

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974




Rysunek 14.10. Typ wpisów Receptury



funkcje należy zaplanować przed uruchomieniem witryny, nawet jeśli początkowo nie planuje się ich używać. Gdy ma się gotową infrastrukturę witryny, łatwiej jest ją później rozbudowywać.

Składniki to jednak nie jedyna taksonomia, jaka będzie nam potrzebna. Przepisy powinno też dać się sortować według rodzajów i dlatego utworzymy jeszcze jedną taksonomię podobną do kategorii, czyli hierarchiczną, do określania rodzaju przepisów. Dzięki temu wybrany wpis będzie można przypisać np. do kategorii *Ciasta*, co pozwoli użytkownikom wyszukać wszystkie przepisy na ciasta w Twojej witrynie. Także w tym przypadku zostaną automatycznie utworzone archiwa, które zawsze mogą się przydać. Ta taksonomia będzie hierarchiczna, ponieważ rodzaje przepisów chcemy trochę ściślej kontrolować. Natomiast składniki są nieograniczone i dlatego do ich reprezentowania lepsza była taksonomia niehierarchiczna.

Zaczniemy od utworzenia taksonomii składników:

```
// Taksonomia Składniki
register_taxonomy( 'ingredients', 'recipes', array(
    'hierarchical' => false,
    'labels' => array(
        'name' => 'Składniki',
        'singular_name' => 'Składnik',
        'search_items' => 'Przeszukuj składniki',
        'popular_items' => 'Popularne składniki',
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
        'all_items' => 'Wszystkie składniki',
        'parent_item' => null,
        'parent_item_colon' => null,
        'edit_item' => 'Edytuj składnik',
        'update_item' => 'Aktualizuj składnik',
        'add_new_item' => 'Dodaj nowy składnik',
        'new_item_name' => 'Nowy składnik',
        'separate_items_with_commas' => 'Oddziel składniki przecinkami',
        'add_or_remove_items' => 'Dodaj lub usuń składniki',
        'choose_from_most_used' => 'Wybierz z najczęściej używanych składników',
        'menu_name' => 'Składniki'
    ),
    'show_ui' => true,
    'query_var' => true,
    'rewrite' => array(
        'slug' => 'skladnik'
    ),
) );
```

Większości tego kodu nie trzeba tłumaczyć. Znaczna jego część jest podobna do kodu służącego do tworzenia własnych typów wpisów, a więc nie powinno być problemów ze zrozumieniem. Na samym początku, zaraz po zarejestrowaniu, taksonomię `ingredients` wiążemy z utworzonym wcześniej typem wpisów o nazwie `recipes`. Gdyby trzeba było ją powiązać z kilkoma typami wpisów, należałoby użyć tablicy:



```
register_taxonomy( 'ingredients', array(
    'recipes',
    'page'
)
```

W powyższym kodzie warto zwrócić uwagę na dwa parametry o wartościach `null`. Dotyczą one tylko taksonomii hierarchicznych. Można je opuścić, ale ponieważ w drugiej taksonomii będą używane, to i tutaj je umieściłem.

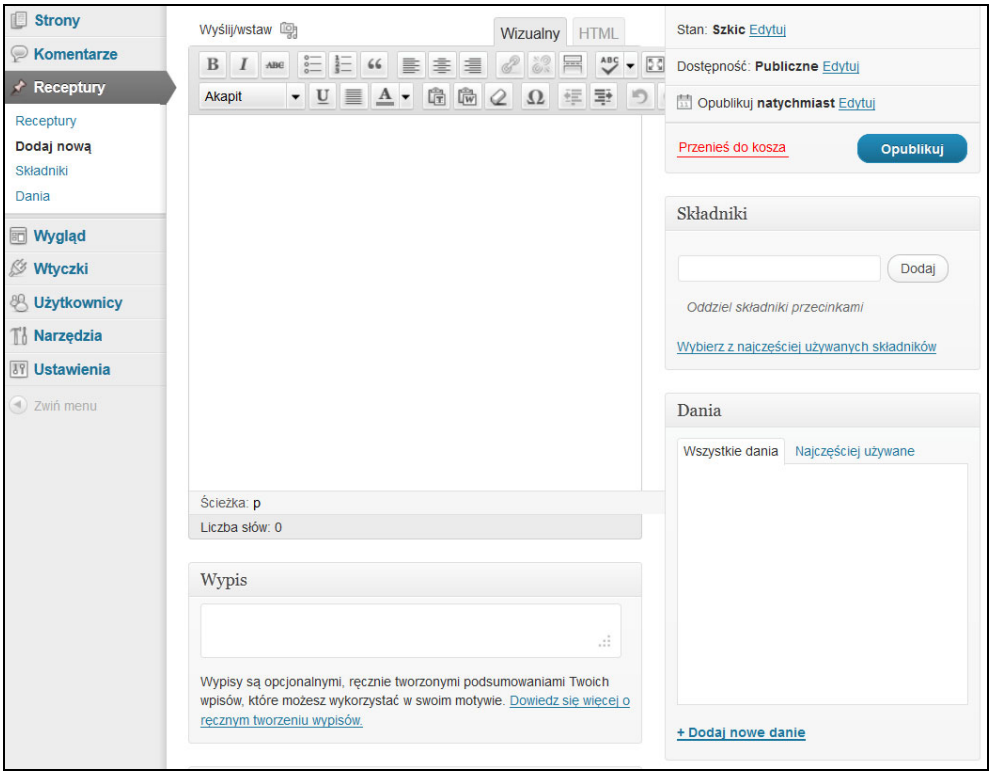
Potrzebujemy też taksonomii dla typów przepisów. Ta taksonomia będzie podobna do kategorii i poza nazwą od kategorii niczym nie będzie się różnić. Nazwiemy ją `dishes` — „dania”, ponieważ większość przepisów w naszym serwisie będzie tego rodzaju:

```
// Taksonomia „dania”
register_taxonomy( 'dishes', 'recipes', array(
    'hierarchical' => true,
     'labels' => array(
         'name' => 'Dania',
         'singular_name' => 'Danie',
         'search_items' => 'Przeszukuj dania',
         'all_items' => 'Wszystkie dania',
         'parent_item' => 'Nadrzędne dania',
         'parent_item_colon' => 'Nadrzędne danie:',
         'edit_item' => 'Edytuj danie',
         'update_item' => 'Aktualizuj danie',
         'add_new_item' => 'Dodaj nowe danie',
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
        'new_item_name' => 'Nazwa nowego dania',
        'menu_name' => 'Dania'
    ),
    'show_ui' => true,
    'query_var' => true,
    'rewrite' => array(
        'slug' => 'danie'
    ),
) );
```

Definicja tej taksonomii jest prawie taka sama jak taksonomii składników. Jedyne różnice to nadanie parametrom `parent_item` i `parent_item_colon` konkretnych wartości zamiast `null` oraz ustawienie parametru `hierarchical` na `true`, co oznacza, że jest to taksonomia hierarchiczna, podobnie jak kategorie. To wszystko. Taksonomie są gotowe do użytku tylko dla typu Receptury, jak widać na rysunku 14.11.




Rysunek 14.11. Taksonomie do tagowania i kategoryzowania receptur

Dzięki temu automatycznie zostaną utworzone archiwa receptur i składników. Oczywiście trzeba poświęcić jeszcze trochę czasu, aby je dopracować, ale na potrzeby przykładu to wystarczy. Jeśli chcesz trochę więcej się pobawić, utwórz specjalne pliki szablonowe dla archiwów.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

### ZIÓŁKO NA TRAWIENIE — PODSUMOWANIE

Przedstawiony kod jest dobry na początek. Masz gotowy własny typ wpisów oraz taksonomie do tagowania i kategoryzowania tych wpisów, dzięki którym otrzymasz odpowiednie archiwa.

Oczywiście można zrobić jeszcze wiele innych rzeczy, np. przenieść cały kod do wtyczki, aby był dostępny także po zmianie motywu. Można też dodać do receptur pola pozwalające na wpisanie czasu gotowania i tym podobnych informacji. Do tego celu nadawałyby się własne pola, których zawartość można wyświetlać za pomocą funkcji `the_meta()` lub `get_post_meta()`. Można też użyć wtyczki, np. More Fields (więcej na jej temat można przeczytać w dodatku A), tworzącej bardziej przyjazne dla użytkownika własne pola i pozwalającej na tworzenie własnych adaptacji.

## TWORZENIE WITRYNY Z ODNOŚNIKAMI

Witryna z odnośnikami może służyć do wielu celów i dlatego przykłady przedstawione w tym podrozdziale będą nieco bardziej ogólne niż poprzednie. Cała idea polega na użyciu wpisów jako łączy, które można sortować według kategorii i tagów. Nie poprzestaniemy jednak na umieszczeniu łącza w treści wpisu. Tytuł wpisu będzie tytułem odnośnika, a docelowy adres URL zapiszemy we własnym polu. Natomiast w polu **Wypis** będziemy wpisywać opisy, szczegółowe informacje i co tam jeszcze zachcemy.



Róbmy jednak wszystko po kolei, oto opis planowanego systemu:

- Każdy wpis blogu ma być łączem w naszej bazie odnośników.
- Zawartość pola tytułu zostanie wykorzystana jako tytuł odnośnika.
- Docelowy adres URL będzie zapisywany w polu własnym o nazwie **URL** (np. **www.google.com**).
- Łącza mogą być porządkowane w dowolnej liczbie kategorii (np. **Wyszukiwarki** albo **Poczta elektroniczna**).
- Łącza mają być oznaczane tagami (np. **darmowe**, **USA**, **czcionki**).
- Do wyświetlania opisów odnośników używana będzie funkcja `the_excerpt()` zamiast `the_content()`, głównie po to, aby pozostawić sobie pole manewru w przyszłości.

Możemy zaczynać właściwą pracę. Najpierw zapiszemy adres URL we własnym polu. W tym celu na stronie dodawania nowego wpisu utworzymy nowe własne pole o nazwie **URL**. W polu wartości tego pola wpiszemy adres, który zostanie wykorzystany do utworzenia łącza z tytułem wpisu jako kotwicą.

Musimy też zmodyfikować pętlę. W projekcie takim jak ten najprawdopodobniej najlepszym rozwiązaniem byłoby utworzenie całkiem nowego motywu, ale na potrzeby przykładu użyjemy motywu domyślnego.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Oto kod pętli:

```
<div <?php post_class(); ?> id="post-<?php the_ID(); ?>">
<?php
    // Własne pole z adresem URL
    $CustomField = get_post_meta( get_the_ID(), 'URL', true );
    if ( !empty( $CustomField ) ) {
?>
    <h2>
        <a href="<?php echo esc_url( $CustomField ) ?>"
          rel="bookmark" title="<?php the_title_attribute(); ?>">
            <?php the_title(); ?>
        </a>
        <?php echo $CustomField; ?>
    </h2>
    <div class="entry">
        <?php the_excerpt(); ?>
    </div>
    <p class="postmetadata">
        Kategoria: <?php the_category( ', ' ) ?>, <?php the_tags( 'tagi: ',
           ', ', '' ); ?> <small><?php the_time( 'F jS, Y' ) ?></small>
    </p>
<?php
    // Brak łącza
    } else {
?>
    <h2><del><?php the_title(); ?></del></h2>
    <div class="entry">
        <p>To łącze jest uszkodzone. Powiadom o tym administratora!</p>
    </div>
    <p class="postmetadata">
        Uszkodzone łącze: <?php edit_post_link( 'Napraw je!', '', ' | ' ); ?>
        Kategoria: <?php the_category( ', ' ); ?>, <?php the_tags( 'tagi: ', ', ',
        ↪'' ); ?> <small><?php the_time( 'F jS, Y' );
        ?></small>
    </p>
<?php } ?>
</div>
```

Wiem, że to brzydka sztuczka, ale koncepcja jest prosta. Musimy upewnić się, że we własnym polu został wpisany adres URL; służy nam do tego poniższy kod:

```
<?php
    // Własne pole z adresem URL
    $CustomField = get_post_meta( get_the_ID(), 'URL', true );
    if ( !empty( $CustomField ) ) {
?>
```

Jeśli zmienna `$CustomField` nie jest pusta, zostanie wykonany kod zwracający tytuł w łączu itd.

Następnie oczywiście tworzymy odnośnik w nagłówku z tytułem, w końcu o to całe to zamieszanie.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
<h2>
    <a href="<?php echo esc_url( $CustomField ) ?>"
      rel="bookmark" title="<?php the_title_attribute(); ?>">
        <?php the_title(); ?>
    </a>
</h2>
```

Najpierw oczyszczamy wartość zmiennej `$CustomField`, która jest adresem URL, za pomocą funkcji `esc_url()`, a następnie ją wyświetlamy. To w zasadzie wszystko.

Chyba że ktoś zapomni wpisać adresu w wartości własnego pola. Jeśli tak się stanie, opisana wcześniej instrukcja warunkowa spowoduje wykonanie klauzuli `else`, która wyświetla przekreślony tytuł i wiadomość, gdyby to kogoś interesowało, że łącze jest uszkodzone. Dla wygody został też dodany odnośnik do edycji. We wpisach bez błędów również może się przydać.

Reszta jest bardzo prosta. Funkcja `the_excerpt()` wyświetla zawartość pola **Wypis** oraz kategorie, tagi i datę.

W ten sposób powstał prosty system do zarządzania odnośnikami oparty na WordPressie. Oczywiście można jeszcze coś dopracować, ale to już kosmetyka do wykonania w plikach szablonowych.



## ALTERNATYWNE ROZWIĄZANIE: ODNOŚNIKOWY FORMAT WPISÓW

Innym rozwiązaniem może być użycie odnośnikowego formatu wpisów. Opcja ta daje mniejsze możliwości, ale w niektórych przypadkach może w zupełności wystarczyć. Odnośnikowy format wpisów pobiera pierwsze łącze znajdujące się w treści.

Aby móc używać tego formatu, najpierw trzeba go włączyć w pliku *functions.php*:

```
add_theme_support( 'post-formats', array( 'link' ) );
```

Do tej tablicy można oczywiście dodać też inne formaty. Wielką zaletą formatów wpisów jest to, że łatwo się je sprawdza. Dostępny jest tag warunkowy `has_post_format()`, za pomocą którego można sprawdzić, czy dany wpis jest w określonym formacie. Informację tę można wykorzystać np. do zdefiniowania odpowiedniego arkusza stylów:

```
if ( has_post_format( 'link' ) {
    echo 'Hej, patrz, to jest łącze!';
}
```

Jeśli wolisz użyć formatów wpisów, musisz nieco zmodyfikować wcześniejszy kod. Więcej informacji o formatach wpisów znajduje się w rozdziale 4.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## KILKA UWAG NA TEMAT ZASTOSOWAŃ

Po co używać WordPressa do tworzenia prostego katalogu odnośników? System ten jest doskonałym wyborem ze względu na łatwość edycji wpisów oraz automatyczne tworzenie różnych kanałów i możliwość sortowania treści na wiele różnych sposobów. Jeśli za pomocą wtyczek umożliwi się dodawanie odnośników użytkownikom, to powstanie solidny system, który można dalej rozbudowywać. Można nawet udostępnić kanał RSS z najnowszymi odnośnikami z wybranych kategorii lub mającymi przypisane określone tagi, aby można je było wyświetlać także w innych serwisach, chociaż w takim przypadku tytuły w kanale powinny wskazywać strony, których adresy zostały wpisane we własnym polu **URL**.

Jak to wykorzystać? Istnieje wiele możliwości. Do popularnych rozwiązań należą niszowe portale z łączami, ale stosując wtyczki, można osiągnąć o wiele więcej. Można na przykład umożliwić głosowanie na łącza albo ich komentowanie. W ten sposób można zachęcić użytkowników do publikowania treści, którą z kolei można wykorzystać do tworzenia list najpopularniejszych łączy itp. Dla zamkniętej grupy witryna taka może być platformą wymiany informacji o znalezionych w internecie zasobach. Ponadto system ten jest dobrą alternatywą dla innych skryptów do publikowania łączy. W końcu niewiele jest platform, które byłyby tak proste w użyciu jak WordPress, a jednocześnie dały się rozbudowywać za pomocą motywów i wtyczek. Kto wie, co można zbudować na bazie takiego systemu.

Można też po prostu użyć tego katalogu do zapisywania łączy do ciekawych miejsc w sieci w kategoriach i archiwach sortowanych według tagów. Funkcje takie są dostępne w kilku usługach do tworzenia zakładek, a więc z pewnością też możesz jakoś to wykorzystać.



## MIESZANIE WPISÓW ODNOŚNIKOWYCH ZE ZWYKŁĄ TREŚCIĄ

Czasami trzeba zmieszać tytuły zawierające odnośniki ze zwykłą treścią. Widziałeś kiedyś witrynę, w której tytuły wpisów są łączami do tych wpisów? Coś takiego jest bardzo łatwo uzyskać. Możesz wykorzystać poprzedni kod i zmodyfikować w nim instrukcję warunkową sprawdzającą wartość własnego pola, na przykład:

```
<div <?php post_class(); ?> id="post-<?php the_ID(); ?>">
    <?php $CustomField = get_post_meta( get_the_ID(), 'URL', true );
    if ( !empty( $CustomField ) ) {
        <div class="linkage">
            <h2>
                <a href="<?php echo esc_url( $CustomField ) ?>"
                  rel="bookmark" title="<?php the_title_attribute(); ?>">
                    <?php the_title(); ?>
                </a>
            </h2>
            <div class="entry">
                <?php the_excerpt(); ?>
            </div>
        </div>
    <?php // Nic tu nie ma, a więc to jest zwykły post!
    } else { ?>
        <h2>
```

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

```
            <a href="<?php the_permalink(); ?>" rel="bookmark" title="Łącze do
            ➥<?php the_title_attribute(); ?>">
                <?php the_title(); ?>
            </a>
        </h2>
        <small><?php the_time( 'F jS, Y' ); ?> by <?php the_author(); ?></small>
        <div class="entry">
            <?php the_content( 'Przeczytaj cały wpis &raquo;' ); ?>
        </div>
        <p class="postmetadata">
            <?php the_tags( 'Tags: ', ', ', '<br />' ); ?> Kategoria: <?php
             the_category( ', ' ); ?> | <?php edit_post_link( 'Edytuj', '', ' |
             '); ?> <?php comments_popup_link('Brak komentarzy &#187;',
             '1 komentarz &#187;', 'Komentarzy: % &#187;'); ?>
        </p>
    <?php } ?>
</div>
```

Jest to poprzedni kod z niewielkimi zmianami. Jeśli własne pole **URL** nie ma wartości, wyświetlana jest treść wpisu z łączem w tytule. Jeśli natomiast ma, robimy to samo co poprzednio. W tej wersji opuściłem kategorie itp. oraz dodałem element `div.linkage`, aby ułatwić dodawanie stylów (chociaż można też użyć funkcji `post_class()`).



Każdy wpis, w którego własnym polu URL została wpisana wartość, będzie miał w tytule łącze prowadzące pod adres wpisany w tym polu, a nie do tego wpisu. Kod ten można łatwo dodać do każdego motywu.

Oczywiście o wiele prościej byłoby użyć formatów wpisów, które można sprawdzać za pomocą funkcji `has_post_format()`.

# INNE ZASTOSOWANIA WORDPRESSA

Jeśli wykażesz się sprytem i inteligencją, to z WordPressa możesz zrobić praktycznie wszystko. Oto kilka dodatkowych podpowiedzi, które może okażą się dla Ciebie inspirujące.

## STRONA Z WYDARZENIAMI I KALENDARZ

Ponieważ tworzenie witryn w oparciu o WordPressa jest bardzo łatwe, system ten zyskał sporą popularność jako platforma do budowania portali o różnych wydarzeniach. Dzięki własnym typom wpisów można łatwo sortować wpisy według mówców czy warsztatów, na stronach można publikować różne informacje, a w blogu po prostu prezentować zwykłe wpisy blogowe. Bardzo łatwo jest utworzyć taką konfigurację i dlatego jest ona tak popularna.

W przypadku serwisu wydarzeniowego trzeba rozwiązać kwestię harmonogramu i funkcji rejestracji. W pierwszym przypadku można użyć jakiejś zewnętrznej usługi kalendarzowej albo jednej z kilku wtyczek tego typu. Jeśli chodzi o rejestrację, to istnieje kilka usług sieciowych obsługujących cały proces, od wstępnej rezerwacji miejsca po płatność. Jeśli

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

chcesz sprzedawać bilety i inne przedmioty związane ze zdarzeniem, zainteresuj się takimi wtyczkami jak WP e-Commerce oraz różnymi koszykami na zakupy. Używano ich już chyba do wszystkiego, od obsługi konferencji WordCamps po małe happeningi.

## INTRANET I WSPÓŁPRACA

Jak wiesz, WordPress jest bardzo łatwy w obsłudze. Dlatego doskonale nadaje się do wewnętrznej komunikacji w zespołach i organizacjach. Niezależnie od tego, czy ma to być nieformalny blog, w którym publikowane będą najświeższe wiadomości od zarządu firmy, czy też źródło informacji w wieloosobowych projektach, WordPress na pewno będzie doskonałym wyborem. Aby w pełni wykorzystać potencjał platformy, można zainstalować dodatkowe wtyczki i dodać inne narzędzia, aby zintegrować WordPressa z innymi systemami działającymi w firmie.

Jeśli szukasz rozwiązania dla grupy współpracowników, sprawdź motyw P2 (*http://wordpress.org/extend/themes/p2*). Jest podobny do Twittera, chociaż można w nim pisać zarówno krótkie, jak i długie teksty. W niektórych firmach używają go do wewnętrznej komunikacji. Może być używany w intranecie albo utajniony za pomocą wtyczek.

Mimo że WordPress to nie wiki, to istnieją też wtyczki pozwalające rozszerzyć możliwości systemu także w tym kierunku. To, czy WordPress nadaje się do użycia, zależy od potrzeb w konkretnym przypadku. Jeśli potrzebujesz tylko zwykłego bloga, to nie ma nic prostszego, ale jeśli Twoje wymagania są inne, musisz dokładnie przemyśleć sprawę.



Do komunikacji wewnętrznej przydatne mogą też być wtyczki społecznościowe.

## SPOŁECZNOŚCI I FORA

Z chwilą pojawienia się wtyczki BuddyPress (*http://buddypress.org*) WordPress stał się również wartą uwagi platformą społecznościową. Dzięki tej wtyczce utworzenie witryny skupiającej społeczność jest o wiele prostsze niż kiedyś. Użytkownicy mogą się bez problemu ze sobą komunikować na różne sposoby, a nawet można udostępnić możliwość tworzenia blogów i forów. Najlepsze jest to, że dzięki pakietowi szablonów BuddyPress (*http://wordpress.org/extend/plugins/bp-template-pack*) wtyczkę tę można zainstalować w praktycznie każdym WordPressie.

Fora można tworzyć przy użyciu wtyczki bbPress (*http://bbpress.org*) dostępnej z BuddyPress albo jako oddzielna wtyczka. Niektórzy pamiętają, że bbPress początkowo był samodzielnym projektem, ale od wersji 2.0 jest wtyczką do WordPressa. To bardzo wygodne rozwiązanie, ponieważ fora są tworzone jako własny typ wpisów i można je łatwo dostosować do motywu istniejącej witryny. Istnieją też inne wtyczki forów, ale zanim ich użyjesz, koniecznie sprawdź, jak sprawują się przy dużym ruchu. Jeśli żadne z tych rozwiązań nie spełnia Twoich oczekiwań, zawsze możesz użyć jakiegoś zewnętrznego systemu, np. Vanilla (*http://vanillaforums.org*) bardzo dobrze integruje się z bazą danych użytkowników WordPressa.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Nie ma wątpliwości, że możliwości rozbudowy WordPressa do postaci platformy społecznościowej są ogromne. Do dyspozycji mamy masę wtyczek, za pomocą których znacznie rozszerzymy nasz system.

## BAZY DANYCH

Wszystko zmieniło się w momencie wprowadzenia własnych typów wpisów. Dzięki możliwości tworzenia dodatkowych narzędzi do zarządzania treścią przy użyciu zaledwie kilku wierszy kodu WordPress stał się niezwykle wszechstronny. To również oznacza, że systemu tego można używać do budowania dużych baz danych zawierających dowolne informacje. A jeśli wykorzysta się kanały RSS, które w WordPressie są tworzone praktycznie dla wszystkiego, to można mieć system automatycznie wysyłający aktualizacje do różnych odbiorców.

Może zabrzmi to trochę buńczucznie, ale dzięki własnym typom wpisów możliwości WordPressa stały się praktycznie nieograniczone. Przeszedł długą drogę od wpisów i stron, prawda?

## STATYCZNE WITRYNY



W internecie wciąż istnieją statyczne witryny, tzn. takie, w których nie publikuje się najświeższych wiadomości i które nieczęsto się aktualizuje. Po co używać WordPressa do budowy takich serwisów? Czy nie łatwiej byłoby utworzyć kilka prostych plików HTML? Może i tak, ale jeśli tworzysz witrynę na zamówienie albo dla znajomego, to WordPress pomoże Ci rozwiązać wiele problemów. Nawet jeśli ma to być tylko prosta wizytówka z kilkoma stronami zawierająca podstawowe informacje o czymś, to jeśli użyjesz WordPressa, każdy, nawet półgłówek, będzie mógł ją w razie potrzeby zmodyfikować. Klient nie będzie musiał Cię wzywać, gdy konkurencyjny sklep zmieni godziny otwarcia albo gdy będzie trzeba dodać nową stronę z informacjami o kolejnym biurze.

Budowa prostej statycznej witryny opartej na WordPressie może zająć trochę więcej czasu, niż gdyby użyło się zwykłych plików HTML, ale później ta inwestycja się zwróci dzięki temu, że nie będziesz musiał własnoręcznie dokonywać każdej zmiany. Oczywiście jeśli pobierasz opłaty za godzinę pracy, to pewnie Cię to nie ucieszy, ale jednak postaraj się dostarczyć każdemu klientowi możliwie najlepszy produkt. Będziesz się z tym lepiej czuł, zapewniam.

## DZIENNIKI I NOTATKI

Czasami tak bardzo skupiamy się na możliwościach rozszerzania systemu, że zapominamy o jego korzeniach. WordPress jest doskonałą platformą do publikacji dzienników, co jest raczej oczywiste, ponieważ pierwotnie blogi były rodzajem dzienników internetowych. Możesz utworzyć dziennik z wpisami o Twoim prywatnym życiu i ukryć go przed wyszukiwarkami za pomocą opcji prywatności. Jeśli potrzebujesz większej prywatności, możesz nawet zainstalować wtyczkę Private WordPress (*http://wordpress.org/extend/plugins/private-wordpress/*), która ukrywa witrynę przed niezalogowanymi użytkownikami. Dzięki temu możesz udostępnić swoje zapiski znajomym i rodzinie, a cały pozostały straszny świat o niczym się nie dowie.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Dzienniki tworzy się też nie tylko dla rozrywki. Używa się ich także do publikowania wyników pracy, postępów na studiach, treści wykładów czy przykładów witryn, które utworzyło się w oparciu o WordPressa. Ponieważ system ten standardowo obsługuje kategorie i tagi, można łatwo grupować treść i umożliwić jej znajdowanie użytkownikom. Jeśli potrzebujesz dodatkowych metod kategoryzacji, zawsze możesz użyć własnych taksonomii i typów wpisów. Takie rozwiązanie też stosujemy w mojej firmie — mamy zamkniętego WordPressa, którego używamy do wymieniania się fragmentami kodu projektów.

# MOŻESZ MIEĆ WSZYSTKO, CZEGO CHCESZ

No, może nie wszystko — nawet WordPress ma ograniczenia — ale prawie wszystko. WordPress to wyjątkowo wszechstronna platforma, na której po niewielkich modyfikacjach można prowadzić większość witryn z treścią. Dzięki łatwym w modyfikacji motywom i wtyczkom pozwalającym rozszerzyć funkcjonalność systemu można zrobić prawie wszystko. W tym rozdziale moim celem było uświadomienie Ci tego. Gdy rozpoczynam nowy projekt, nieważne czy własny, czy dla klienta, najpierw zastanawiam się, jakich narzędzi użyć, aby jak najszybciej go zrealizować. Bardzo często stwierdzam, że najlepiej będzie użyć WordPressa.

Może nie wszystko uda Ci się zrealizować przy użyciu WordPressa, ale większość pomysłów na pewno tak. Największą zaletą tego systemu jest to, że można go zamienić w coś kompletnie innego niż oryginalna wersja. Dzięki ruchowi open source mamy doskonałe narzędzie z wszystkimi jego wtyczkami i motywami. Powinniśmy być za to wdzięczni i starać się pomagać tej społeczności, kiedy tylko możemy.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# V

# DODATKI

Dodatek A Niezbędne wtyczki do WordPressa

Dodatek B Motywy bazowe

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# A

# NIEZBĘDNE WTYCZKI DO WORDPRESSA

**LICZBA DOSTĘPNYCH WTYCZEK** do WordPressa jest przytłaczająca. Nie ma nawet najmniejszych szans, żeby wypróbować wszystkie ani nawet nadążyć za zmianami. W tym rozdziale znajduje się opis tylko kilku z tych fantastycznych dodatków do systemu. Każda z opisanych wtyczek ma wielu użytkowników, została dokładnie przetestowana i spełnia konkretne zadanie. Należy jednak pamiętać, że opisane tu wtyczki to tylko kropla w morzu wszystkich wtyczek.

Pamiętaj, że wtyczki, podobnie jak sam WordPress, ciągle się zmieniają. Z tego powodu coś, co dziś działa doskonale, jutro może być już mniej przydatne. Dlatego nie poprzestawaj na lekturze tego rozdziału, tylko poszukaj aktualizacji i innych wtyczek, które mogą lepiej spełniać Twoje wymagania.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*Czasami przed instalacją wtyczki wyświetlane jest ostrzeżenie, że wtyczka została przetestowana do określonej wersji WordPressa, ale nie w tej, której używasz. To niekoniecznie oznacza, że wtyczka nie będzie działać. Po prostu jeszcze jej nie przetestowano w tej konkretnej wersji systemu albo autor nie zaktualizował jej plików. Dlatego nie zrażaj się, jeśli ujrzysz taki komunikat. Zainstaluj testowego bloga i w nim sprawdzaj wszystkie niepewne wtyczki. Większość z nich dobrze działa we wszystkich wersjach WordPressa.*

# WTYCZKI ZWIĄZANE Z TREŚCIĄ

WordPress służy do publikowania treści. Chcielibyśmy, aby użytkownicy ją znaleźli, zainteresowali się nią i spędzili cały wieczór na jej przeglądaniu. W Wikipedii służą do tego łącza i Ty też możesz skorzystać z tego pomysłu. Trzeba tylko sprawić, żeby były interesujące i aby łatwo dało się przeglądać archiwa. Poniżej znajduje się opis wtyczek, które pomogą Ci zainteresować użytkowników treścią witryny.



- WP Greet Box (*http://wordpress.org/extend/plugins/wp-greet-box*) — gdy do Twojej witryny trafia użytkownik z portalu społecznościowego, dobrze by było go jakoś przywitać. Wtyczka WP Greet Box sprawdza, skąd przyszedł użytkownik, i wykorzystuje tę informację do wyświetlenia odpowiedniego napisu powitalnego. Jako powitanie można napisać coś w rodzaju „Witaj, subskrybuj kanał RSS" albo coś bardziej dostosowanego do sytuacji, np. „Witaj, użytkowniku portalu Nasza Klasa. Wiesz, że napisałem o nim książkę?".
- Yet Another Related Posts Plugin (*http://wordpress.org/extend/plugins/yet-another-related-posts-plugin/*) — słowa „Yet Another" (czyli kolejny) w nazwie tej wtyczki to nie tylko ukłon w stronę wielbicieli kodu źródłowego. Jest to także odniesienie do tego, że wtyczek tego typu istnieje ogromna ilość, przez co trudno jest wyłuskać spośród nich prawdziwe perły. Moim zdaniem wymieniona to poważny pretendent do tronu w tej kategorii. Ma dużo opcji i dodatkowo obsługuje automatyczne tworzenie odnośników. Między innymi można określać, jaki stopień podobieństwa między wpisami nas interesuje, co jest przydatne w witrynach zawierających bardzo dużo treści. Wśród innych ustawień znajdują się możliwość dodawania podobnych wpisów do kanałów RSS, obsługa stron, buforowanie oraz system szablonów, który mógłby być trochę prostszy w użyciu, ale i tak jest przydatny.
- WordPress Popular Posts (*http://wordpress.org/extend/plugins/wordpress-popular-posts*) — chcesz promować najpopularniejsze wpisy w swojej witrynie? Jeśli tak, to ta wtyczka jest doskonałym narzędziem, za pomocą którego zachęcisz odwiedzających do przejrzenia głębszych pokładów treści w Twoim serwisie. Jest wiele wtyczek tego typu, ale WordPress Popular Posts umożliwia tworzenie widżetów i jest bardzo dobrze zrobiona.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- WP-PostRatings (*http://wordpress.org/extend/plugins/wp-postratings*) — jest to doskonała wtyczka do oceniania wpisów. Do wyboru jest kilka zestawów grafik reprezentujących oceny oraz można wybrać, gdzie moduł oceniania ma być wyświetlony, wstawiając w wybrane miejsce skrót kodowy `[ratings]` albo wywołując odpowiednią funkcję w pliku szablonowym. Można też wyświetlić najwyżej i najniżej oceniane wpisy, sortować oceny według czasu, kategorii itd.
- GD Star Rating (*http://wordpress.org/extend/plugins/gd-star-rating*) — kolejna wtyczka do oceniania wpisów, która wyróżnia się obsługą kilku rodzajów ocen dla jednego wpisu (np. gwiazdki i kciuk w górę lub w dół) oraz możliwością oceniania komentarzy. Ponadto ma jedną z najładniejszych stron administracyjnych, jakie widziałem, i sporo zaawansowanych ustawień. Warto wziąć ją pod uwagę, jeśli szukasz wtyczki do oceniania wpisów.
- Polldaddy (*http://wordpress.org/extend/plugins/polldaddy*) — to, że hostowana wtyczka Polldaddy w pełni integruje się z WordPressem i umożliwia tworzenie sond bez opuszczania panelu administracyjnego, nie powinno być zaskoczeniem, ponieważ jest własnością Automattic. Już choćby ze względu na samego właściciela warto ją wypróbować. Kolejny powód to dobra infrastruktura zaplecza. Jeśli jednak będziesz chciał usunąć łącza Polldaddy ze swoich ankiet, musisz wykupić konto pro. Jeżeli jest to dla Ciebie nie do przyjęcia, możesz użyć innej wtyczki.
- WP-Polls (*http://wordpress.org/extend/plugins/wp-polls*) — elastyczna wtyczka do tworzenia sond z obsługą archiwów. Sondę można wyświetlić, dodając odpowiedni kod do plików motywu albo używając widżetu. Trochę czasu może zająć dostosowywanie stylu sond do motywu, ale jest to dobra alternatywa dla hostowanej Polldaddy.



# WTYCZKI MULTIMEDIALNE

W przypadku WordPressa słowo **media** najczęściej oznacza obrazy graficzne i to samo dotyczy wtyczek. Gdy się nad tym zastanowić, to można dojść do wniosku, że w sumie nie ma w tym nic dziwnego. Przecież jeśli masz film, to raczej publikujesz go w YouTube, prawda? Dzięki temu znajdzie go więcej osób i nie trzeba przejmować się obciążeniem łączy serwera.

- Lightbox Gallery (*http://wordpress.org/extend/plugins/lightbox-gallery*) — wtyczka umożliwiająca wyświetlanie obrazów z galerii przy użyciu popularnej, częściowo przezroczystej nakładki na stronę. Można także sprawić, aby dowolny obraz wyświetlał się na takiej nakładce. W tym celu wystarczy umieścić go w odnośniku z atrybutem `rel="lightbox"`. To samo można zrobić z każdym innym skryptem, ale w tym jest to już gotowe. Należy jednak pamiętać, że aktualnie wtyczka zawiera trochę na sztywno wpisanych odnośników i jeśli przeniesiesz instalację WordPressa do podfolderu, ale witrynę będziesz publikować w katalogu głównym, to niektóre odnośniki nawigacyjne mogą przestać działać. Miejmy nadzieję, że w przyszłości zostanie to naprawione. W przeciwnym razie możesz być zmuszony do poszukania innych rozwiązań.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Slimbox (*http://wordpress.org/extend/plugins/slimbox*) — jeśli szukasz prostego rozwiązania do publikowania zdjęć na lekkiej nakładce, sprawdź Slimbox. Wtyczka ta znajduje wszystkie łącza prowadzące do obrazów i modyfikuje je. Może nie jest tak wyrafinowana jak inne tego rodzaju wtyczki, ale na pewno jest warta uwagi.
- Podcasting (*http://wordpress.org/extend/plugins/podcasting*) — kiedyś publikowanie podcastów w WordPressie nierozerwalnie wiązało się z używaniem wtyczki PodPress. Niestety wtyczka ta nie jest już rozwijana. Na szczęście istnieje narzędzie pozwalające przenieść treść PodPress do wtyczki Podcasting, która obsługuje iTunes oraz umożliwia publikowanie zarówno podcastów audio, jak i wideo.
- Featured Articles Lite (*http://wordpress.org/extend/plugins/featured-articles-lite/*) — wtyczka umożliwiająca tworzenie przewijanych pokazów istniejących stron i wpisów.

## WTYCZKI ADMINISTRACYJNE

Sekcja opisów wtyczek administracyjnych zawiera najwięcej propozycji, ponieważ przedstawione w niej są rozwiązania od robienia kopii zapasowej po statystyki i nakładki CMS-owe.



- Disabler (*http://wordpress.org/extend/plugins/disabler/*) — wtyczka umożliwiająca wyłączenie kilku często niepotrzebnych funkcji WordPressa, takich jak wysyłanie pingów do siebie, automatyczne zapisywanie wpisów, zapisywanie różnych wersji wpisów, kanały RSS czy XML-RPC. Zazwyczaj do wyłączenia każdej z tych funkcji trzeba instalować kilka wtyczek, ale jeśli nie chcesz tego robić, to za pomocą tej wtyczki możesz wyłączyć każdą z nich i jeszcze kilka innych z osobna, używając prostego w obsłudze graficznego interfejsu użytkownika.
- WP No Category Base (*http://wordpress.org/extend/plugins/wp-no-category-base*) — wtyczka umożliwiająca pozbycie się domyślnego słowa **category** z odnośników. W WordPressie słowo to można zmienić na dowolne inne, np. **tematy** albo **produkty**, ale standardowo nie ma możliwości usunięcia go w ogóle. Ta wtyczka to zmienia, dzięki czemu adresy typu *http://domena.com/category/moja-kategoria* zamieniają się w *http://domena.com/moja-kategoria*.
- WP-DB-Backup (*http://wordpress.org/extend/plugins/wp-db-backup*) — kopii zapasowych nigdy za wiele. Ta wtyczka wykorzystuje wbudowanego w WordPressa „pseudocrona" do wysyłania kopii zapasowych bazy danych na podany adres e-mail albo zapisuje je na serwerze. Niezależnie od tego, jakich narzędzi do tworzenia kopii zapasowej już używasz, gorąco zachęcam Cię do zainstalowania również tego. Wtyczka ta robi kopię zapasową standardowych tabel WordPressa, jak również innych, które się wskaże. Nie robi natomiast kopii plików, dla których trzeba użyć innego narzędzia. Możliwe, że w przyszłości wtyczka ta będzie dostępna standardowo z WordPressem. Jestem pewien, że ucieszyłoby to wielu ludzi.
- WP Maintenance Mode (*http://wordpress.org/extend/plugins/wp-maintenance-mode/*) — umożliwia utworzenie strony informującej, że witryna jest w trakcie przebudowy. Strona ta jest wyświetlana zamiast normalnej treści wszystkim użytkownikom oprócz tych, którzy mają uprawnienia do zmieniania opcji motywu.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- Custom Admin Branding (*http://wordpress.org/extend/plugins/custom-admin-branding*) — pozwala dostosować wygląd panelu administracyjnego użytkowników. Można podmienić logo w formularzu logowania i panelu administracyjnym, zmienić kolory różnych elementów itd. Możliwe jest też dodawanie własnych arkuszy stylów, aby precyzyjnie dostosować wybrane elementy.
- Theme My Login (*http://wordpress.org/extend/plugins/theme-my-login*) — zamienia tradycyjną stronę logowania (*wp-login.php*) na stronę motywu. Inaczej mówiąc, sprawia, że formularz logowania jest zintegrowany z motywem i można go formatować za pomocą CSS. Jest to bardzo przydatne w witrynach, w których użytkownicy mogą się logować. Dodatkowo można określić, gdzie użytkownik ma zostać skierowany po zalogowaniu się na swoim koncie.
- Fast Secure Contact Form (*http://wordpress.org/extend/plugins/si-contact-form*) — wtyczek do tworzenia formularzy kontaktowych jest cała masa. Niestety wiele z nich nie radzi sobie z botami spamującymi. Wtyczka Fast Secure Contact Form ma dobre opinie i jest łatwa w obsłudze, ale stanowi jedno z wielu podobnych rozwiązań.
- Google Analyticator (*http://wordpress.org/extend/plugins/google-analyticator*) — wtyczka ułatwiająca podłączenie systemu statystyk Google Analytics (*http://google.com/analytics*) do witryny. Nie wymaga zmieniania czegokolwiek w plikach szablonowych. Wyświetla nawet statystyki w panelu administracyjnym, co jest przydatne dla uzależnionych od sprawdzania statystyk wielokrotnie w ciągu dnia.
- Google Analytics for WordPress (*http://wordpress.org/extend/plugins/google-analytics-for-wordpress*) — kolejna wtyczka pozwalająca włączyć statystyki Google Analytics w witrynie bez potrzeby modyfikowania plików szablonowych. Prosta, chyba że chcesz precyzyjnie śledzić statystyki ruchu.
- WordPress.com Stats (*http://wordpress.org/extend/plugins/stats*) — jeśli przyzwyczaiłeś się do statystyk dostępnych w panelu administracyjnym bloga na WordPress.com, to na pewno spodoba Ci się ta wtyczka. Jest to ten sam program tylko w wersji przeznaczonej dla samodzielnych instalacji WordPressa. Wśród zalet można wymienić prostotę i ładny wygląd, ale wtyczka ta nie śledzi tak szczegółowych statystyk, jak Google Analytics czy jakikolwiek inny „prawdziwy" system statystyk. Do działania wymagany jest darmowy klucz API WordPress.com.
- Broken Link Checker (*http://wordpress.org/extend/plugins/broken-link-checker*) — niewielkie narzędzie, które będzie pilnować Twoich łączy. Po zainstalowaniu przeszukuje bazę danych WordPressa w poszukiwaniu niedziałających odnośników, a następnie wyświetla je oraz udostępnia narzędzia pozwalające coś z nimi zrobić. Bardzo przydatna, ale chyba nie pozostawiłbym jej włączonej przez cały czas, ponieważ sprawdza wszystkie łącza co 72 godziny, chociaż można też włączać sprawdzanie ręcznie.
- WP e-Commerce (*http://wordpress.org/extend/plugins/wp-e-commerce*) — jeśli chcesz zamienić swoją witrynę w sklep albo przynajmniej sprzedawać za jej pośrednictwem kilka produktów, wtyczka WP e-Commerce jest z pewnością tym, czego potrzebujesz. Nauka jej obsługi pewnie zajmie Ci trochę czasu, ale gdy już ją dostosujesz zarówno od strony projektowej, jak i konfiguracyjnej, będzie działać dokładnie tak, jak chcesz. Wtyczka ta ma wiele zaawansowanych opcji, takich jak promocje krzyżowe



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

i kategoryzowanie produktów. A jeśli potrzebujesz czegoś więcej, możesz zamówić dodatkowe płatne rozszerzenia, chociaż w podstawowej formie wtyczka sprawdza się doskonale.

Innym rozbudowanym rozwiązaniem e-commerce dla WordPressa jest wtyczka Jigoshop (*http://wordpress.org/extend/plugins/jigoshop*). Warto ją porównać z WP e-Commerce, zanim się dokona wyboru. Obie wtyczki udostępniają wiele zaawansowanych opcji i trzeba wybrać tę, która lepiej pasuje do określonego projektu.

- Redirection (*http://wordpress.org/extend/plugins/redirection*) — umożliwia tworzenie przekierowań 301. Jeśli np. adres strony *http://domena.pl/firma* zmieni się na *http://firma.pl*, to za pomocą tej wtyczki można zrobić stałe przekierowanie pod nowy adres.
- Pretty Link Lite (*http://wordpress.org/extend/plugins/pretty-link*) — jeśli chcesz skrócić swoje adresy URL, aby móc ich używać w Twitterze lub po prostu ukryć fakt, że są łączami programów partnerskich (niełądnie!), to ta wtyczka jest dla Ciebie. Narzędzie to przyda Ci się, zwłaszcza jeśli planujesz publikować na Twitterze i masz krótką domenę, ponieważ wtyczka ta umożliwia nawet dodanie specjalnego paska (Pretty Bar) podobnego do stosowanego przez Digg i inne serwisy. Pretty Link jest prostym narzędziem do skracania adresów, ale istnieje też bardziej rozbudowana, płatna wersja tej wtyczki.
- Pods (*http://wordpress.org/extend/plugins/pods*) — wtyczka pozwalająca jeszcze bardziej zbliżyć WordPressa funkcjonalnością do typowych systemów CMS. Twórcy nazywają ją **szkieletem** do tworzenia CMS-ów i wcale nie przesadzają. Można tworzyć typ treści, struktury danych, ustanawiać relacje itd. Budując witrynę w oparciu o tę wtyczkę, otrzymasz wiele narzędzi, dzięki którym możesz naprawdę dużo zdziałać. Jedyną jej wadą jest to, że początkowo może trochę przytłaczać, zwłaszcza osoby przyzwyczajone do prostoty WordPressa. Jeśli jednak chcesz zamienić swój system w CMS-a, warto ją wypróbować.
- Download Monitor (*http://github.com/mikejolley/download-monitor*) — jeśli chcesz wiedzieć, jak często pobierany jest jakiś plik, np. udostępniany przez Ciebie motyw do WordPressa albo książka elektroniczna, to dzięki tej wtyczce możesz te informacje zdobyć. Wtyczka udostępnia narzędzie do wysyłania plików (chociaż nie ma obowiązku z niego korzystać), jest przystosowana do lokalizacji, obsługuje kategorie oraz pozwala w łatwy sposób dodawać pliki do pobrania do stron i wpisów. Oczywiście wyświetla też statystyki.
- More Fields (*http://wordpress.org/extend/plugins/more-fields*) — znakomita wtyczka ułatwiająca pracę z własnymi polami zarówno programistom, jak i redaktorom portalu. Umożliwia tworzenie pól za pomocą edytora WYSIWYG, pól wyboru, przycisków radiowych itd. Jest szczególnie przydatna, gdy tworzy się coś dla klienta. Tak właśnie powinny wyglądać własne pola w WordPressie.

  Mogą Cię zainteresować także inne wtyczki tego samego programisty. More Taxonomies (*http://wordpress.org/extend/plugins/more-taxonomies*) ułatwia dodawanie taksonomii, a More Types (*http://wordpress.org/extend/plugins/more-types*) służy do dodawania nowych typów wpisów.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Inną wtyczką poprawiającą funkcjonalność własnych pól jest WP-Custom (*https://github.com/GeekPress/WP-Custom-Fields*), chociaż moim zdaniem More Fields jest lepsza.

- WP Bannerize (*http://wordpress.org/extend/plugins/wp-bannerize*) — doskonała, prosta wtyczka do wyświetlania banerów. Umożliwia zakodowanie miejsc na reklamę w plikach motywu, jak również umieszczanie reklam na widżetach. Reklamy można grupować w pule. Ponadto wtyczka śledzi liczbę wyświetleń i kliknięć reklam oraz automatycznie wyłącza kampanię reklamową, gdy zostanie osiągnięta określona liczba wyświetleń lub kliknięć albo data. Istnieją też inne wtyczki tego typu, ale jeśli będziesz szukać czego bardziej rozbudowanego, to sprawdź DoubleClick for Publishers firmy Google (*http://google.com/admanager*) i OpenX (*http://openx.com*).
- FeedWordPress (*http://wordpress.org/extend/plugins/feedwordpress*) — umożliwia wysłanie kanałów RSS do bazy danych WordPressa i opublikowanie ich jako wpisów. Brzmi jak spełnienie marzeń podbieraczy kanałów RSS, ale wtyczkę tę można wykorzystać także do innych celów, np. do utworzenia witryny pobierającej odpowiednią treść i wyświetlającej ją do czytania użytkownikom.

  Wtyczkę tę można wykorzystać na wiele sposobów. Teoretycznie dzięki niej można zamienić WordPressa w „strumieniowego potwora", sortując wpisy według kategorii itp.

---

*Podbieranie kanałów RSS innych osób jest nieetyczne i może być niezgodne z prawem. Nie rób tego. Napisz własną treść albo zapytaj o pozwolenie.*

---



- Members Only (*http://wordpress.org/extend/plugins/members-only*) — pozwala na udostępnianie zawartości witryny tylko zalogowanym użytkownikom. Zalogowanego użytkownika można skierować pod dowolny adres, a więc wtyczka ta doskonale współpracuje z motywem P2 (*http://wordpress.org/extend/themes/p2*) pozwalającym tworzyć witryny do współpracy w obrębie organizacji. Jest nawet możliwość tworzenia prywatnych kanałów RSS.
- BuddyPress (*http://buddypress.org*) — wtyczka do tworzenia portali społecznościowych przy użyciu WordPressa. To bardzo duże rozszerzenie dodaje do systemu funkcje podobne do Facebooka, takie jak możliwość tworzenia stron profili, grup itd. Są nawet wtyczki rozszerzające BuddyPress.
- bbPress (*http://bbpress.org*) — mimo że jest wiele bardzo dobrych skryptów forów internetowych, np. Vanilla (*http://vanillaforums.org*), czasami wolelibyśmy uniknąć instalowania kolejnej aplikacji. W takich przypadkach z pomocą przychodzi forum bbPress, które jest używane m.in. w portalu WordPress.org. Oprócz łatwej instalacji i konfiguracji zaletą bbPress jest też to, że forum to można rozszerzać za pomocą wtyczek, podobnie jak BuddyPress, z którą również doskonale współpracuje. Stylizowanie forum jest bardzo łatwe. Wystarczy tylko dodać odpowiednie pliki szablonowe, ponieważ bbPress używa własnych typów wpisów.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Poniżej znajduje się opis jeszcze kilku wtyczek administracyjnych.

- Woopra Analytics Plugin (*http://wordpress.org/extend/plugins/woopra*) — doskonała wtyczka dla użytkowników Woopra, podobnie jak wtyczki Google Analytics są przydatne dla użytkowników statystyk Google.
- Random Redirect (*http://wordpress.org/extend/plugins/random-redirect*) — wtyczka umożliwiająca utworzenie odnośnika do losowego wpisu w witrynie.
- Post Editor Buttons (*http://wordpress.org/extend/plugins/post-editor-buttons)* — wtyczka umożliwiająca dodanie własnych przycisków do edytora wpisów w trybie HTML.
- TinyMCE Advanced (*http://wordpress.org/extend/plugins/tinymce-advanced*) — wzbogaca funkcjonalność wizualnego edytora wpisów WordPressa.
- Viper's Video Quicktags (*http://wordpress.org/extend/plugins/vipers-video-quicktags*) — ułatwia dodawanie filmów z różnych portali i wysyłanie na serwer własnych plików.
- Custom Post Type UI (*http://wordpress.org/extend/plugins/custom-post-type-ui*) — wtyczka pozwalająca na tworzenie własnych typów wpisów i taksonomii z poziomu panelu administracyjnego bez konieczności pisania kodu.
- Revision Control (*http://wordpress.org/extend/plugins/revision-control*) — ogranicza liczbę wersji wpisów zapisywanych przez WordPressa w bazie danych.
- Widget Context (*http://wordpress.org/extend/plugins/widget-context*) — wtyczka pozwalająca ustanawiać reguły wyświetlania widżetów (może stanowić zamiennik dla dziesiątków pasków bocznych).
- WP Mail SMTP (*http://wordpress.org/extend/plugins/wp-mail-smtp*) — jeśli nie chcesz używać serwerów poczty swojego hosta do wysyłania wiadomości do list mailingowych z WordPressa, to ta wtyczka jest dla Ciebie. Narzędzie to dobrze współpracuje też z Google Apps.
- Editorial Calendar (*http://wordpress.org/extend/plugins/editorial-calendar*) — ułatwia zarządzanie wpisami dzięki wygodnemu interfejsowi w postaci harmonogramu z możliwością przeciągania i upuszczania elementów.
- Members (*http://wordpress.org/extend/plugins/members*) — wtyczka zwiększająca kontrolę nad uprawnieniami użytkowników. Bardzo ciekawe narzędzie, które warto wypróbować.
- WP Event Ticketing (*http://wordpress.org/extend/plugins/wpeventticketing*) — rozbudowana wtyczka do zarządzania sprzedażą biletów. Obsługuje płatności PayPal i może Ci się przydać przy organizacji konferencji.
- Activate Update Services (*http://wordpress.org/extend/plugins/activate-update-services*) — przywraca sekcję *Usługi aktualizacji* do witryn w sieci witryn.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# WTYCZKI DO ZARZĄDZANIA KOMENTARZAMI I ELIMINOWANIA SPAMU

Walka ze spamem i zarządzanie komentarzami to bardzo ważne czynności, których nie można zaniedbać. Poniżej przedstawiam opisy wtyczek, które Ci w tym pomogą.

- Akismet (*http://wordpress.org/extend/plugins/akismet*) — wtyczka instalowana wraz z WordPressem pozwalająca rozwiązać uciążliwy problem spamu w komentarzach. Do jej używania potrzebny jest klucz API WordPress.com, a więc trzeba mieć konto w tym serwisie. Dostępne są też licencje komercyjne, o których informacje można znaleźć na stronie *http://akismet.com*. Jeśli dużo spamu będzie się przedostawać przez filtr, możesz też wypróbować inne wtyczki do zwalczania spamu.
- TypePad AntiSpam (*http://antispam.typepad.com*) — propozycja firmy Six Apart. Jest konkurencją dla wtyczki Akismet i podobnie do niej działa. Podobnie jak Akismet współpracuje z serwerem, który rejestruje i analizuje spam, w tym komentarze oznaczone jako spam przez administratora witryny, dzięki czemu ciągle się uczy. Zarówno wtyczka, jak i klucz do API TypePad AntiSpam są darmowe, a więc jeśli Akismet nie spełni Twoich oczekiwań, możesz spróbować tego rozwiązania.
- Really Simple CAPTCHA (*http://wordpress.org/extend/plugins/really-simple-captcha*) — CAPTCHA to te denerwujące pola, w których trzeba wpisać napis pokazany na obrazku, aby udowodnić, że jest się człowiekiem. Niestety mimo że jest to niewygodne dla użytkowników, czasami po prostu nie da się inaczej pozbyć natrętnych spamerów i innych eksploiterów. Nie jest to wtyczka, którą włącza się dla wszystkich komentarzy jak leci — można dostosować, kiedy ma być używana. W większości przypadków sprawdza się bardzo dobrze.
- Popular Widget (*http://wordpress.org/extend/plugins/popular-widget/*) — prosta wtyczka do wyświetlania popularnych wpisów w widżecie na pasku bocznym bloga. Ma kilka opcji, które warto wypróbować, aby dostosować widżet do własnych potrzeb.
- Disqus Comment System (*http://wordpress.org/extend/plugins/disqus-comment-system*) — wtyczka pozwalająca w łatwy sposób uruchomić hostowaną usługę obsługi komentarzy Disqus (*http://disqus.com*). Jeśli jej użyjesz, to być może zainteresuje Cię też nieoficjalna wtyczka Disqus Widget (*http://wordpress.org/extend/plugins/disqus-widget/*) wyświetlająca statystyki komentarzy.
- IntenseDebate Comments (*http://wordpress.org/extend/plugins/intensedebate*) — aktualnie istnieją dwie liczące się hostowane usługi obsługi komentarzy, które ze sobą konkurują. IntenseDebate (*www.intensedebate.com*) jest własnością firmy Automattic. Jeśli chcesz skorzystać z narzędzi tej firmy, to ta wtyczka Ci to umożliwi.
- CommentLuv (*http://wordpress.org/extend/plugins/commentluv*) — wtyczka sprawdzająca, czy pod adresem podanym przez użytkownika dodającego komentarz znajduje się kanał RSS i wyświetlająca najnowszy wpis z tego kanału wraz z komentarzem. Ponadto łączy się z serwisem ComLuv (*http://comluv.com*) w celu uzyskania dodatkowych funkcji, takich jak śledzenie linków. Domyślne ustawienia nie są zbyt dobrze skonfigurowane, ale na szczęście można je łatwo pozmieniać.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- BackType Connect (*http://wordpress.org/extend/plugins/backtype-connect*) — wtyczka przeglądająca portale społecznościowe w poszukiwaniu rozmów na temat wpisów na naszym blogu i publikująca je jako komentarze do tych wpisów. Jeśli więc Twój artykuł o mamutach wywołał wielką dyskusję na Twitterze, dyskusja ta zostanie wyświetlona także pod Twoim wpisem. Całkiem fajne narzędzie, ale mieszanie zwykłych komentarzy z komentarzami z platform mikroblogowych może nie najlepiej wyglądać. Używaj go z głową i tylko wtedy, gdy Twoi czytelnicy są na tyle obeznani z tematem, że będą wiedzieć, o co chodzi.

I jeszcze parę ciekawych drobiazgów.

- Cookie for Comments (*http://wordpress.org/extend/plugins/cookies-for-comments*) — wtyczka pomagająca zwalczać spam, utrudniająca działanie botom spamującym poprzez użycie cookie.
- WP-Hashcash (*http://wordpress.org/extend/plugins/wp-hashcash*) — kolejna wtyczka do zwalczania spamu, którą warto wypróbować, jeśli ma się z nim problemy.
- ReplyMe (*http://wordpress.org/extend/plugins/replyme*) — wysyła wiadomość e-mail do autora komentarza, gdy ktoś odpowie bezpośrednio na jego komentarz. Działa oczywiście tylko przy włączonym zakorzenianiu komentarzy.



## WTYCZKI MEDIÓW SPOŁECZNOŚCIOWYCH

Sieci społecznościowe są bardzo popularne i mamy do wyboru wiele portali tego typu. Każdy taki portal udostępnia specjalne narzędzia, które można dodać bezpośrednio do motywu (zobacz rozdział 10.), ale można również użyć specjalnych wtyczek. Skoro w tych czasach wszystko jest połączone w jedną wielką sieć, to czemu by nie podłączyć do niej też swojej witryny? Możesz pochwalić się swoimi wpisami na Twitterze albo pozwolić użytkownikom pokazywać swoją treść w serwisach typu Digg. Większość tych funkcji można zaimplementować bezpośrednio w plikach motywu (odpowiedni kod można znaleźć na stronach partali), ale jeśli chcesz sobie ułatwić pracę albo skorzystać z dodatkowych funkcji, to wypróbuj wtyczki opisane poniżej.

- Lifestream (*http://wordpress.org/extend/plugins/lifestream*) — wtyczka dodająca funkcję strumieniowania do witryny. Po instalacji należy podać konta i kanały RSS, z których mają być pobierane dane, a następnie wstawić skrót kodowy na wybranej stronie. Ponadto dzięki dodatkowi klas CSS każdy element można dostosować do własnych potrzeb. Wtyczka standardowo obsługuje największe portale społecznościowe, ale można jej używać praktycznie z każdym serwisem, który udostępnia kanał RSS. Narzędzie jest przygotowane do lokalizacji i często aktualizowane.
- Twitter Tools (*http://wordpress.org/extend/plugins/twitter-tools*) — wtyczka łącząca WordPressa z Twitterem i umożliwiająca wysyłanie wpisów do Twittera z WordPressa. Jest prosta w konfiguracji, a nawet pozwala na określenie tagów dla wpisów, które mają być wysyłane do Twittera. Dzięki temu można np. utworzyć kategorię uwag na marginesie i wszystko, co w niej zostanie opublikowane, wysyłać do Twittera.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

- ShareThis (*http://wordpress.org/extend/plugins/share-this*) — wysyłanie wpisów do portali społecznościowych stało się bardzo modne, a ta wtyczka ułatwia robienie tego dla każdego wpisu. ShareThis to w istocie coś więcej niż wtyczka. Jest to usługa, w której przechowywany jest formularz zatwierdzania, dzięki czemu jeśli założysz konto, możesz śledzić różne statystyki.

  Jedynym mankamentem tego rozwiązania może być to, że nasza witryna będzie zależna od zewnętrznej usługi. Jeśli serwer ShareThis będzie działał wolno albo stanie się niedostępny, to samo stanie się z funkcjami udostępniania.
- Share Buttons by Lockerz / AddToAny (*http://wordpress.org/extend/plugins/add-to-any*) — wtyczka zawierająca hostowany przycisk udostępniania Lockerz Share (wcześniej AddToAny). Jeśli podoba Ci się ta usługa, możesz też wypróbować Subscribe Button (*http://wordpress.org/extend/plugins/add-to-any-subscribe*).

Oto jeszcze kilka innych ciekawych wtyczek dotyczących mediów społecznościowych.

- Wickett Twitter Widget (*http://wordpress.org/extend/plugins/wickett-twitter-widget*) — prosta wtyczka wyświetlająca wpisy z Twittera.
- Twitter for WordPress (*http://wordpress.org/extend/plugins/twitter-for-wordpress*) — kolejna wtyczka służąca do wyświetlania wpisów z Twittera.
- Sociable (*http://wordpress.org/extend/plugins/sociable*) — popularna wtyczka pozwalająca dodać na strony odnośniki do portali społecznościowych.
- Tweet Old Post (*http://wordpress.org/extend/plugins/tweet-old-post*) — wtyczka pozwalająca wysyłać do Twittera wpisy o losowych starych wpisach w WordPressie. W ten sposób można uzyskać dodatkowy ruch w witrynie, jeśli ma się oddanych fanów na Twitterze.
- Simple Social Bookmarks (*http://wordpress.org/extend/plugins/simple-social-bookmarks*) — kolejna wtyczka do obsługi mediów społecznościowych, obsługuje ponad 200 portali.
- SexyBookmarks (*http://wordpress.org/extend/plugins/sexybookmarks*) — popularna wtyczka do wyświetlania ikon portali społecznościowych i prezentująca je w atrakcyjny sposób.



## WTYCZKI SUBSKRYPCJI I DO OBSŁUGI URZĄDZEŃ PRZENOŚNYCH

Kanały RSS są obsługiwane już od dłuższego czasu przez większość smartfonów i niektóre zwykłe telefony z wyższej półki. Coraz więcej treści użytkownicy czytają na urządzeniach przenośnych, dlatego warto udostępniać wersję witryny przystosowaną do oglądania w tych „zabawkach". Poniższe wtyczki pomogą Ci w tym.

- Align RSS Images (*http://wordpress.org/extend/plugins/align-rss-images*) — spływające do lewej lub prawej krawędzi obrazy mogą bardzo dobrze wyglądać na normalnych stronach, ale w kanałach RSS mogą nie spływać, tylko stać na samym środku. Jednym

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

z rozwiązań jest ich usunięcie, ale szkoda byłoby to robić. Lepiej użyć wtyczki Align RSS Images, która znajduje elementy z domyślnymi klasami `alignleft` i `alignright` WordPressa i zamienia je na odpowiedniki HTML. Nie trzeba definiować żadnych ustawień, wtyczka działa od razu po zainstalowaniu.

- RSS Footer (*http://wordpress.org/extend/plugins/rss-footer*) — dodaje wiersz treści na początku lub końcu każdego elementu w kanale RSS. Dzięki niej można dodać informacje o prawach autorskich, aby utrudnić działanie robotom zbierającym treść albo promować witrynę lub produkty wśród użytkowników korzystających z czytników RSS. Bardzo wygodne i łatwe w obsłudze narzędzie.
- Disable RSS (*http://wordpress.org/extend/plugins/disable-rss*) — wtyczka wyłączająca kanały RSS WordPressa. Może być przydatna w statycznych witrynach, w których kanały nie są używane.
- Subscribe2 (*http://wordpress.org/extend/plugins/subscribe2*) — znakomite narzędzie pozwalające użytkownikom subskrybować nowości w witrynie i wysyłające powiadomienia na adres e-mail użytkowników. Można wysyłać informacje o najnowszych wpisach codziennie albo co tydzień. Można też wysyłać e-maile do zarejestrowanych użytkowników jak tradycyjny newsletter. Konfiguracja i szablony są proste, a więc szybko można się uporać z ustawieniami. Jeśli chodzi o wysyłanie poczty e-mail, to niektóre hosty to utrudniają, a więc należy wszystko dokładnie przetestować, zanim wyśle się mailing. Ponadto dostawcy usług internetowych mogą Cię potraktować jako spamera, a więc uważaj.
- MobilePress (*http://wordpress.org/extend/plugins/mobilepress*) — wtyczka włączająca specjalny motyw, gdy użytkownik korzysta z urządzenia przenośnego. Można ustawić, w jakich sytuacjach ma być włączany ten specjalny motyw. Ponadto dostępny jest interfejs podobny do standardowego, pozwalający utworzyć motyw dostosowany wyglądem do indywidualnych potrzeb.



Jeszcze kilka wtyczek, które też mogą Ci się spodobać.

- WordPress Mobile Edition (*http://wordpress.org/extend/plugins/wordpress-mobile-edition*) — kolejna wtyczka dostosowująca witrynę do wyświetlania na urządzeniach przenośnych.
- WordPress Mobile Pack (*http://wordpress.org/extend/plugins/wordpress-mobile-pack*) — kolejne rozwiązanie ułatwiające pracę użytkownikom urządzeń przenośnych.
- WPtouch (*http://wordpress.org/extend/plugins/wptouch*) — wtyczka przystosowująca witrynę do urządzeń z ekranem dotykowym, takich jak telefony z systemem Android i iPhone'y. Dla osób bardziej wymagających dostępna też jest wersja profesjonalna.

# WTYCZKI DOTYCZĄCE SEO I WYSZUKIWANIA

Akronim SEO pochodzi od angielskich słów **Search Engine Optimization** i oznacza optymalizowanie witryn internetowych pod kątem wyszukiwarek, aby jak najwięcej osób mogło za ich pośrednictwem znaleźć naszą treść. Na szczęście nie musisz wynajmować

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

eksperta od SEO, ponieważ istnieją wtyczki do WordPressa, które pomogą Ci dostosować Twoją witrynę.

- All in One SEO Pack (*http://wordpress.org/extend/plugins/all-in-one-seo-pack*) — wtyczka dodająca ustawienia do wpisów pozwalające optymalizować wpisy pod kątem wyszukiwarek. Wiele osób dałoby się za nią pokroić i nie ma wątpliwości, że jest pomocna. Już samo jej zainstalowanie i pozwolenie, aby działała, wystarcza, aby polepszyć jakość witryny. Aby jednak maksymalnie wykorzystać potencjał narzędzia, należy każdy wpis dostosować z osobna.
- WordPress SEO by Yoast (*http://wordpress.org/extend/plugins/wordpress-seo*) — jest wiele wtyczek SEO do WordPressa, ale WordPress SEO by Yoast Joosta de Valka należy do najbardziej lubianych. Jest łatwa w obsłudze nawet dla początkujących, ale mimo to udostępnia także wiele zaawansowanych opcji dla zaawansowanych użytkowników. Warto wziąć ją pod uwagę, jeśli szukasz wtyczki SEO.
- Google XML Sitemaps (*http://wordpress.org/extend/plugins/google-sitemap-generator*) — wtyczka tworząca mapę witryny w formacie XML i aktualizująca ją, gdy w serwisie zostanie opublikowany nowy wpis lub zmodyfikowany już istniejący. Pomaga wyszukiwarkom internetowym w przeglądaniu treści witryn, co jest bardzo korzystne. Wtyczka próbuje nawet informować wyszukiwarki, gdy w witrynie coś się zmieni.
- Better Search (*http://wordpress.org/extend/plugins/better-search*) — standardowa wyszukiwarka WordPressa nie jest, delikatnie mówiąc, zbyt dobra, przez co wielu użytkowników szuka rozwiązań zastępczych, takich jak wtyczki czy własne wyszukiwarki Google (*http://google.com/cse*). Wtyczka udoskonala algorytm wyszukiwania oraz obsługuje popularne zapytania i umożliwia tworzenie map popularnych wyszukiwań. Zastępuje standardową wyszukiwarkę, a więc łatwo ją wypróbować.
- HeadSpace2 SEO (*http://wordpress.org/extend/plugins/headspace2*) — jedna z łatwiejszych w obsłudze wtyczek SEO. Zawiera świetne strony konfiguracyjne i pozwala na więcej niż tylko dostosowanie metadanych i opisów, można np. za jej pomocą zarządzać statystykami Google Analytics. Czy jest lepsza od innych rozwiązań tego typu? To zależy od Twoich potrzeb i oczekiwań. Na pewno jednak jest to jedna z najłatwiejszych w obsłudze wtyczek.
- Robots Meta (*http://wordpress.org/extend/plugins/robots-meta*) — utworzenie pliku *robots.txt* dla wyszukiwarek internetowych jest dobrym pomysłem, a wtyczka Robots Meta pomoże Ci w tym, udostępniając proste opcje dla kategorii i innych typów archiwów. Bardzo przydatne narzędzie dla osób znających się na zagadnieniach SEO.
- Global Translator (*http://wordpress.org/extend/plugins/global-translator*) — wtyczka umożliwiająca dodanie do stron widżetu z flagami państw, których kliknięcie powoduje przetłumaczenie treści na określony język (który musi być obsługiwany przez używanego elektronicznego translatora, np. Tłumacz Google albo BabelFish). Wtyczka obsługuje buforowanie i bezpośrednie odnośniki pomagające w optymalizacji pod kątem wyszukiwarek.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

I jeszcze kilka dodatkowych wtyczek.

- Search Everything (*http://wordpress.org/extend/plugins/search-everything*) — kolejna modyfikacja funkcji wyszukiwania z wyróżnianiem słów kluczowych i rozszerzoną paletą opcji wyszukiwania.
- GD Press Tools (*http://wordpress.org/extend/plugins/gd-press-tools*) — wtyczka nie dla początkujących. Dodaje wiele opcji konfiguracyjnych dla wszystkich komponentów, od metatagów po własne pola i crona. Zachować ostrożność.
- Breadcrumb Trail (*http://wordpress.org/extend/plugins/breadcrumb-trail*) — wtyczka umożliwiająca dodanie do stron nawigacji okruszkowej poprzez wywołanie specjalnej funkcji w plikach szablonowych. Podobna do Yoast Breadcrumbs (*http://wordpress.org/extend/plugins/breadcrumbs*).

# KOD ŹRÓDŁOWY I DANE WYJŚCIOWE

W tej kategorii wybór wtyczek jest ogromny. Część z nich pokrywa się funkcjonalnością, niektóre w zasadzie do niczego nie są przydatne, ale jest też sporo takich, które nie pasowały do innych kategorii, a też warto o nich wspomnieć. Większość z opisanych tu wtyczek dotyczy prezentowania kodu źródłowego i modyfikowania sposobu zwracania danych, a więc jest to mieszanina różnych narzędzi.



- SyntaxHighlighter Evolved (*http://wordpress.org/extend/plugins/syntaxhighlighter*) — jeśli kiedykolwiek publikowałeś kod źródłowy w swoich wpisach lub na stronach, to wiesz, że standardowe mechanizmy WordPressa mogą z tego zrobić niezły bigos. Oczywiście są różne rozwiązania typu „pastebin", ale chyba wygodniej jest rozwiązać problem na miejscu, korzystając z wtyczki SyntaxHighlighter Evolved. Dzięki niej kod będzie prezentowany poprawnie i na dodatek zostanie odpowiednio pokolorowany. Jest też możliwość definiowania własnych stylów, dzięki czemu ramki z kodem można dostosować do swojej witryny. Wtyczek tego rodzaju jest więcej, ale ta jest niezawodna.
- Blog Time (*http://coffee2code.com/wp-plugins/blog-time*) — wyświetla znacznik czasu serwera. Czas ten można wyświetlić przy użyciu widżetu albo wywołując funkcję `blog_time()` w wybranym miejscu pliku szablonowego. To nie jest zegar, tylko narzędzie wyświetlające znacznik czasu, który też w niektórych przypadkach może być przydatny.
- WP-Cirrus (*http://wordpress.org/extend/plugins/wp-cumulus*) — wtyczka, do której powstania inspiracją była przestarzała już wtyczka WP-Cumulus, wyświetlająca trójwymiarową chmurę tagów. Można jej używać jako widżetu albo w dowolnym miejscu wpisów, korzystając ze skrótu kodowego [wp-cirrus].
- wp-Typography (*http://wordpress.org/extend/plugins/wp-typography*) — wtyczka poprawiająca typografię na stronach, np. zastosuje spacje niełamliwe między liczbami i jednostkami, upiększy cudzysłowy, myślniki itp.
- Widget Logic (*http://wordpress.org/extend/plugins/widget-logic*) — prosta i genialna wtyczka dodająca do każdego widżetu pole, w którym można wpisać tag warunkowy.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Dzięki temu można np. wybrany widżet wyświetlić tylko na stronach pojedynczych wpisów, używając tagu `is_single()`. To pomaga zwiększyć dynamikę witryny.

- WP Super Cache (*http://wordpress.org/extend/plugins/wp-super-cache*) — obowiązkowa wtyczka dla osób prowadzących popularne witryny z dużym ruchem, ale nie chcących „zajeździć" swojego serwera. Wtyczka tworzy statyczne pliki, dzięki czemu zamiast za każdym razem sięgać do bazy danych, system serwuje statyczne pliki. Jeśli planujesz wbić się na stronę główną Digg ze swoim blogiem technologicznym, to ta wtyczka utrzyma Twoją witrynę online. Wtyczka WP Super Cache jest lepiej utrzymana od swojej poprzedniczki o nazwie WP Cache, ale ma jedną wadę: buforuje także dynamiczną treść, przez co listy najnowszych komentarzy mogą **przestać** być aktualne. Można jednak zaznaczyć, co ma być buforowane, a co nie.

  Jest też wtyczka o nazwie W3 Total Cache (*http://wordpress.org/extend/plugins/w3-total-cache*), która może być znakomitą alternatywą dla WP Super Cache. Wypróbuj obie i zdecyduj, która bardziej Ci odpowiada.

- Query Posts (*http://wordpress.org/extend/plugins/query-posts*) — wtyczka dodająca widżet pozwalający utworzyć w pasku bocznym dostosowaną do własnych potrzeb pętlę WordPressa bez znajomości języka PHP! Może być bardzo przydatna dla osób chcących wyświetlić treść w wymyślony przez siebie sposób w pasku bocznym lub jakimkolwiek innym obszarze na widżety. Dobrze integruje się z wtyczką Get the Image (*http://wordpress.org/extend/plugins/get-the-image*) umożliwiającą pobieranie obrazów z treści wpisów, własnych pól i załączników.



I jeszcze parę ciekawych propozycji:

- WP-DBManager (*http://wordpress.org/extend/plugins/wp-dbmanager*) — przyspiesza działanie bazy danych, dokonując niezbędnych napraw, oraz umożliwia robienie kopii zapasowych bazy danych.
- PHP Snippets (*http://wordpress.org/extend/plugins/php-snippets*) — wtyczka ułatwiająca wykonywanie kodu PHP we wpisach i widżetach. Jej działanie opiera się na specjalnych plikach *.snippet.php*, które umieszcza się w specjalnie skonfigurowanym katalogu i których zawartość można wykonać za pomocą skrótu kodowego. Na przykład jeśli utworzymy plik o nazwie *moj-kod.snippet.php*, to we wpisie możemy go wykonać za pomocą skrótu kodowego `[moj-kod]`.
- WP-PageNavi (*http://wordpress.org/extend/plugins/wp-pagenavi*) — wtyczka rozszerzająca funkcjonalność nawigacji między stronami. Zawiera kilka ustawień dotyczących stylu, ale żeby działała, trzeba dodać wywołanie specjalnej funkcji do plików szablonowych.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PRZESTROGA NA ZAKOŃCZENIE: CZY NA PEWNO POTRZEBUJESZ TEJ WTYCZKI

Pracując z WordPressem, łatwo dać się ponieść emocjom i przesadzić z liczbą wtyczek. Jest ich tak dużo i dzięki nim można tyle zrobić, a ponadto tak łatwo się je instaluje z poziomu panelu administracyjnego, że naprawdę czasami trudno jest się oprzeć.

Jednak powinieneś wykazać silną wolę. Każda dodatkowa wtyczka i funkcja powiększa system i powoduje jego spowolnienie. Nie chodzi tylko o same pliki, które są umieszczone w folderach wtyczek, ale przede wszystkim o to, co zapisane w nich skrypty robią z bazą danych i jak wiele wysyłają do niej zapytań.

Nie zrozum mnie źle, wtyczki są naprawdę przydatne. Wielu fantastycznych witryn opartych na WordPressie bez nich po prostu by nie było, a poza tym możliwość rozszerzania systemu za pomocą wtyczek stanowi właśnie jego siłę. Samo to, że cały dodatek został poświęcony wtyczkom i temu, co można dzięki nim zrobić, już o czymś świadczy!

Tylko nie daj się ponieść. Używaj tylko tych wtyczek, które są Ci potrzebne, i staraj się, aby Twój system był tak prosty, jak to tylko możliwe.



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# B

# MOTYWY BAZOWE

**MOTYWÓW DO WORDPRESSA** jest zatrzęsienie. Jednak gdy będziesz szukać czegoś dla siebie, większość z nich odrzucisz, ponieważ nie spodoba Ci się styl, funkcjonalność albo po prostu będą zbyt niskiej jakości. Ty jednak wiesz już co nieco o budowie motywów i możesz utworzyć własny, który będzie dokładnie odpowiadał Twoim potrzebom. Możliwe, że w tym celu zechcesz użyć jakiegoś motywu jako bazy. W tym dodatku znajduje się lista motywów, które pomogą Ci zaoszczędzić na czasie oraz mogą stać się dla Ciebie inspiracją do dbania o jakość.

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

*W dodatku tym znajduje się opis tylko darmowych motywów. Istnieje ogromny rynek płatnych motywów, wśród których podobnie jak wśród darmowych można znaleźć lepsze i gorsze sztuki. Jeśli zdecydujesz się na użycie płatnego motywu, dokładnie przeczytaj warunki użytkowania i upewnij się, że jest udostępniany na licencji zgodnej z GPL.*

# JAK WYBRAĆ MOTYW

Wybór odpowiedniego motywu to niełatwe zadanie. Przede wszystkim należy zastanowić się, co jest nam potrzebne. Jeśli np. planujesz szybko uruchomić jakąś witrynę, to możliwe, że wystarczy jakiś gotowy motyw, w którym ewentualnie dokonasz paru modyfikacji. Jeśli natomiast planujesz utworzyć szkielet motywu, to powinieneś szukać czegoś całkiem innego, co będzie odpowiadało wymogom, jakie stawiasz swojemu przyszłemu szkieletowi. Do każdego projektu potrzebny jest inny motyw i przy wyborze należy kierować się tą podstawową zasadą.

Jeśli chcesz utworzyć szkielet motywu, ale chciałbyś szybko rozpocząć pracę, to możesz poszukać jakiegoś motywu, który będzie w dużym stopniu od razu spełniał Twoje oczekiwania. W takim przypadku możesz wybrać motyw praktycznie pozbawiony stylu albo taki, który całkiem dobrze wygląda i trzeba go tylko dostosować. Niezależnie która z opisanych sytuacji Ci się przydarzy, chcę, abyś sobie uświadomił, że tak naprawdę Twoja praca będzie polegać na utworzeniu nowej gałęzi istniejącego motywu. Oznacza to, że gdy wybrany przez Ciebie motyw zostanie zaktualizowany, będziesz musiał te aktualizacje wdrożyć także u siebie.



Z drugiej strony, jeśli użyjesz gotowego szkieletu, zaoszczędzisz sporo czasu i będziesz korzystać z darmowych aktualizacji, a dodatkowo będziesz mógł szybciej rozpocząć pracę nad własnym motywem. Pamiętaj tylko, aby wybrać motyw, który jest na bieżąco aktualizowany oraz zgodny z najnowszą wersją WordPressa i ogólnie najświeższymi trendami w technologiach tworzenia stron internetowych. Warto wybrać motyw z językiem HTML5, który jest przyszłościowy.

Niezależnie od tego, co wybierzesz, warto przyjrzeć się najnowszym rozwiązaniom, ponieważ w ten sposób można się wiele nauczyć.

## ZNACZENIE SŁOWA SZKIELET

Słowa **szkielet** w książce tej używałem dość beztrosko. Nie było to przypadkowe, ponieważ nie ma ogólnej zgody co do tego, czym jest szkielet, motyw bazowy i prawidłowy motyw nadrzędny. Gdybyśmy chcieli się czepiać, to moglibyśmy stwierdzić, że większość motywów nazywanych szkieletami w istocie wcale nimi nie jest — są raczej motywami bazowymi. Szkielet to najogólniej mówiąc zestaw funkcji, na których można budować własny motyw.

Nie jest to jednak miejsce ani pora na wymądrzanie się i denerwowanie programistów, którzy myślą, że utworzyli szkielet, a w istocie mają motyw bazowy. Po prostu wybierz ten motyw, który Ci się spodoba, i nie przejmuj się drobiazgami.

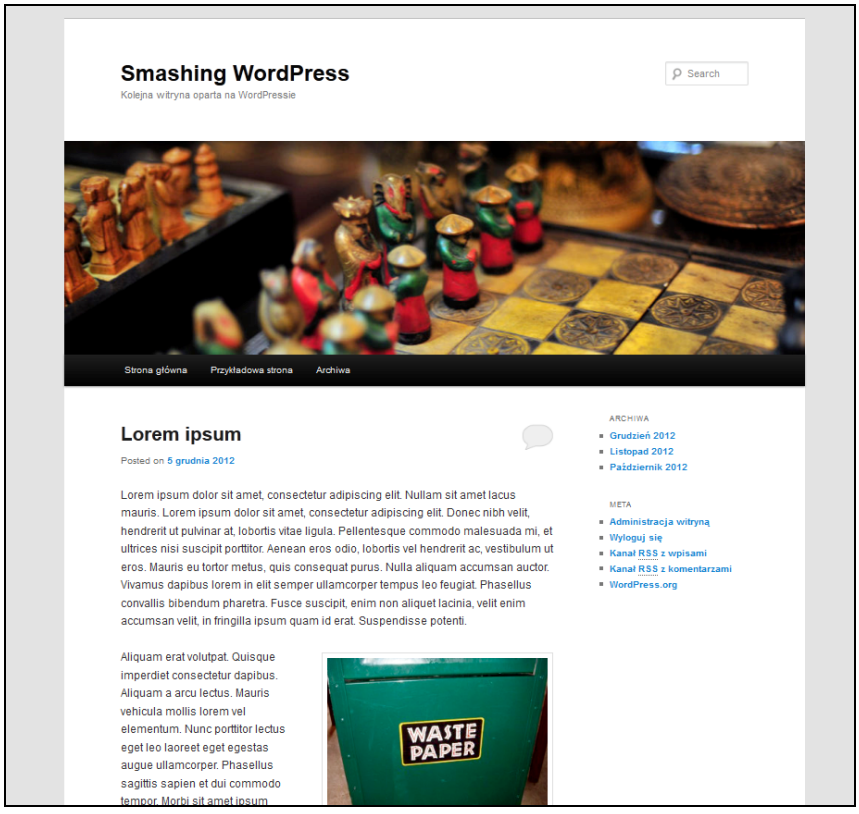
ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PROPOZYCJE MOTYWÓW

Poniżej przedstawiam listę motywów, które powinny Cię w jakiś sposób zainspirować. Niezależnie od tego, czy chcesz utworzyć własny motyw bazowy, szybko uruchomić witrynę, czy tylko przekonać się, jak fajne mogą być motywy, to jest dobre miejsce do rozpoczęcia. Moim celem nie jest olśnić Cię przepięknym projektem, lecz pomóc Ci znaleźć coś, od czego będziesz mógł rozpoczynać swoje projekty. Wybierz to, co dla Ciebie najlepsze, i zrób z tego dobry użytek.

## MOTYWY TWENTY TEN I TWENTY ELEVEN

Może to banał, ale domyślne motywy WordPressa doskonale nadają się do nauki. Zarówno Twenty Ten (*http://wordpress.org/extend/themes/twentyten*), jak i Twenty Eleven (*http://wordpress.org/extend/themes/twentyeleven*) to porządne motywy, które warto dokładnie przejrzeć (rysunek B.1). Zajrzyj też do motywu o nazwie **Twenty Ten Five** (*www.twentytenfive.com*), który jest przerobioną na HTML5 wersją motywu Twenty Ten. Zobaczysz, jak można z jednego motywu zrobić coś nowego i na dodatek lepszego.



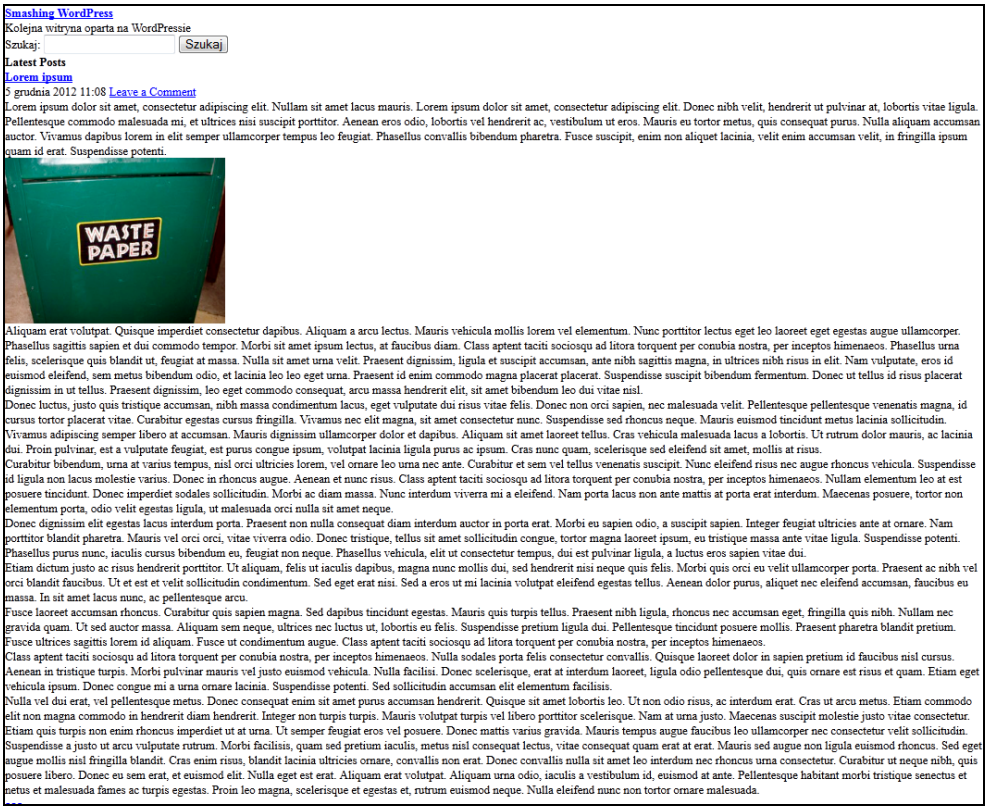

Rysunek B.1. Motyw Twenty Eleven



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## STARKERS

Starkers (*http://viewportindustries.com/#starkers*) to świetny motyw, na bazie którego można zbudować własny motyw (rysunek B.2). Jest kompletnie pozbawiony elementów projektowych, przez co wygląda, jakby strony ładowały się bez arkuszy stylów. Kod HTML jest dobrze napisany, a więc nawet jeśli nie chcesz go użyć, to i tak warto się mu przyjrzeć, aby się czegoś nauczyć. Możesz też go rozgałęzić i utworzyć na jego bazie własny szkielet. Jedno jest pewne, jest to dobry punkt do rozpoczęcia pracy.



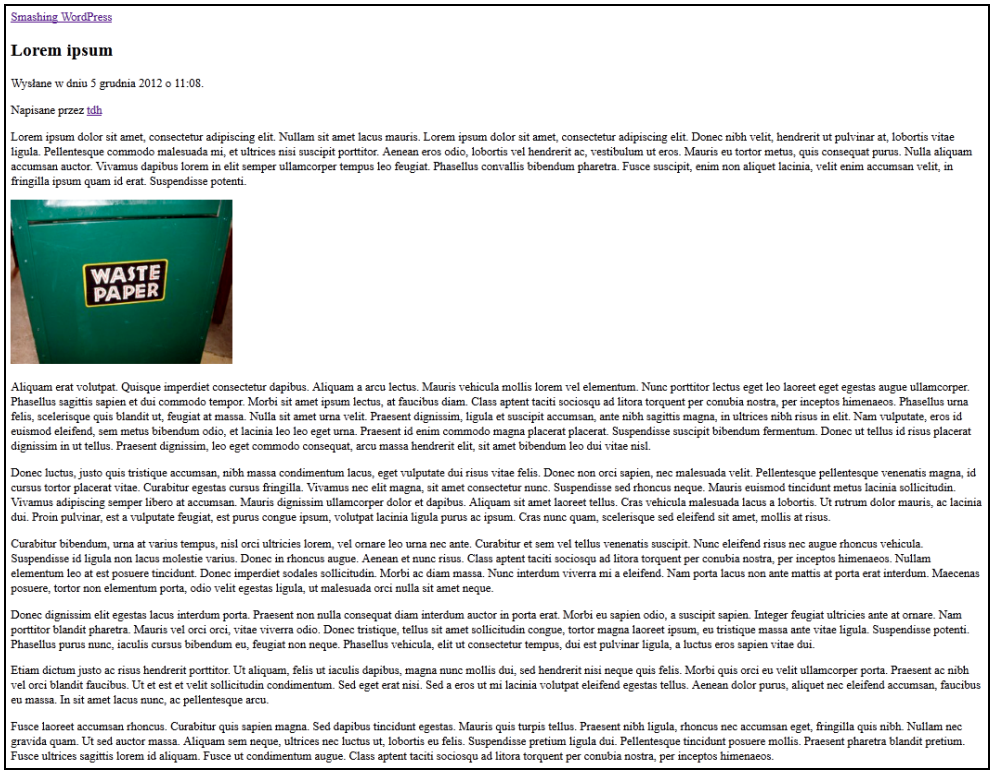

Rysunek B.2. Motyw Starkers, jak widać, jest dość surowy



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## ROOTS

Roots (*www.rootstheme.com*) to motyw zbudowany przy użyciu HTML5 Boilerplate (*http://html5boilerplate.com*) i kilku systemów siatkowych, które mogą Ci się bardzo przydać (rysunek B.3). Motyw ten dość mocno zmienia działanie WordPressa poprzez modyfikację działania funkcji `wp_head()` oraz przepisuje adresy URL. Może Ci się to spodobać albo i nie. Ponadto jest trochę ustawień dających dodatkową kontrolę nad motywem. Nawet jeśli Ci się nie spodoba, to warto go poznać, bo sporo można się nauczyć.



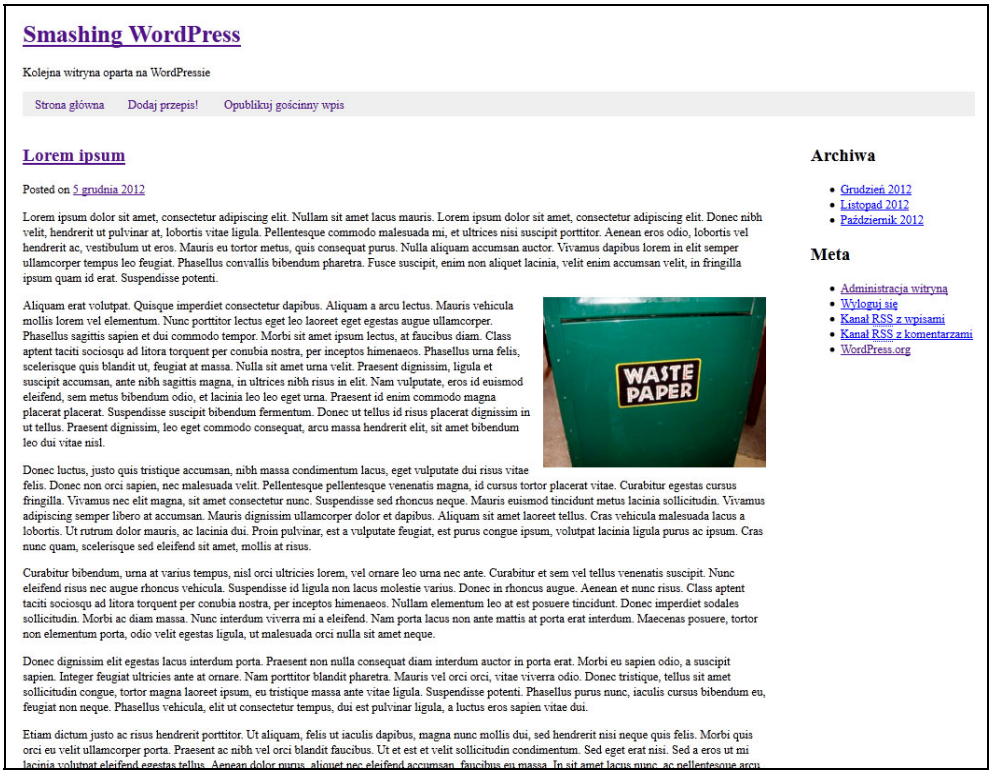

Rysunek B.3. Motyw Roots



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## TOOLBOX

Toolbox (*http://wordpress.org/extend/themes/toolbox*) to prosty motyw napisany przy użyciu języka HTML5 i przystosowany do formatowania za pomocą CSS (rysunek B.4). Nie ma on z założenia pięknie wyglądać od razu po zainstalowaniu. Trzeba dodać własne grafiki, zdefiniować kolory, kroje pisma itd. Warto go wybrać, gdy trzeba szybko utworzyć dobrze wyglądającą witrynę z prostą prezentacją treści. W takiej sytuacji wystarczy tylko napisać arkusze stylów oraz dodać obrazy i gotowe.



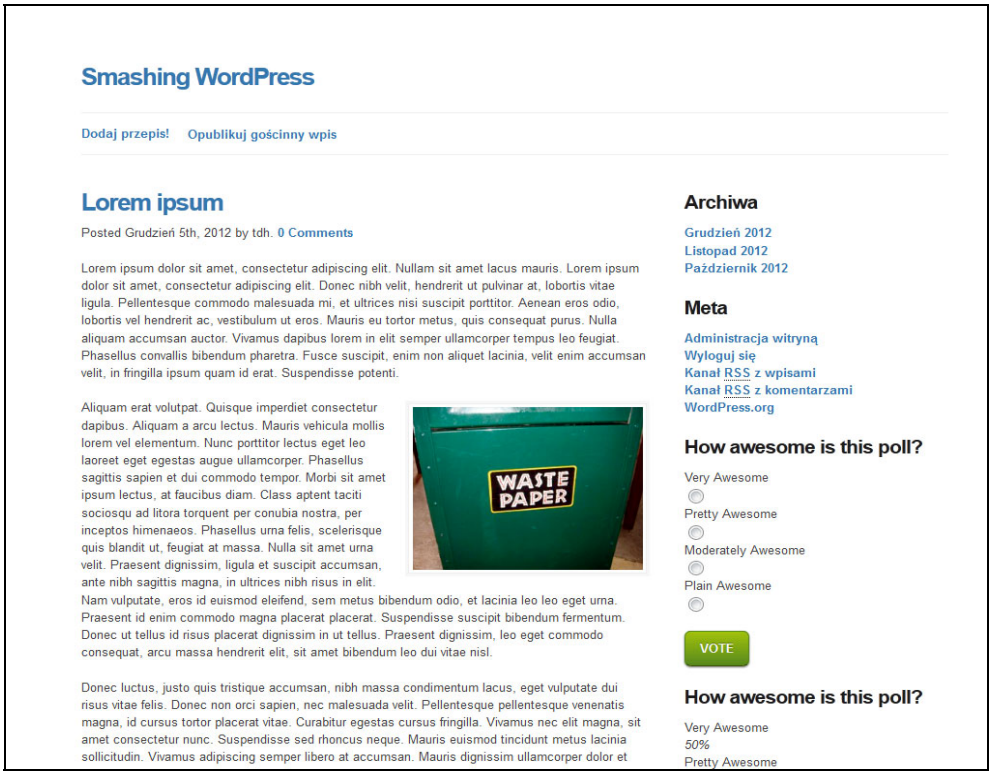

Rysunek B.4. Motyw Toolbox doskonale nadaje się na bazę własnego motywu



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## CONSTELLATION

Constellation (*http://constellationtheme.com*) to dość prosty motyw zbudowany przy użyciu HTML5 Boilerplate (rysunek B.5). Dzięki zastosowanym w nim zapytaniom o media motyw ten automatycznie dostosowuje się do różnych rozdzielczości, aczkolwiek jego układ nie jest w pełni płynny. Warto mu się przyjrzeć, jeśli potrzebujesz czegoś prostego na szybki start albo chcesz się zapoznać z zastosowaniami zapytań o media.




Rysunek B.5. Motyw Constellation



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## SPECTACULAR

Spectacular (*http://bit.ly/hJQpjn*) to motyw z rozbudowanym stylem. Zamieściłem go tu raczej dla inspiracji niż jako propozycję do użycia we własnych projektach (rysunek B.6). Jest to doskonały przykład tego, jak bardzo motyw może koncentrować się na warstwie wizualnej i być mniej lub bardziej przydatny do budowy mniejszych witryn. Układ treści odpowiada typowym wymaganiom mniejszych serwisów i analizując go, możesz dowiedzieć się czegoś nowego, co wykorzystasz w przyszłości we własnych projektach.

Rysunek B.6. Motyw Spectacular



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## BONES

Bones (*http://themble.com/bones*) to motyw utworzony przy użyciu HTML5, dostępny w prostej standardowej (zwanej **klasyczną** — rysunek B.7) i responsywnej płynnej wersji. Dobrze nadaje się jako baza do budowy każdej witryny.

Rysunek B.7. Wersja klasyczna motywu Bones

# TWÓJ MOTYW, TWOJE ZASADY

Niezależnie od tego, jaki motyw wybierzesz, pamiętaj, że mimo iż ktoś inny wykonał dużą część pracy, to jednak Ty będziesz jego użytkownikiem. Nie bój się wprowadzać zmian i go rozwijać. Oczywiście doceniaj czyjąś pracę, jeśli jest tego warta, i jeśli znajdziesz jakieś ciekawe rozwiązanie, podziel się nim ze społecznością.

Motywy opisane w tym dodatku nie są atrakcyjne wizualnie, ale mają tylko służyć jako jeden z etapów budowy witryny oraz pozwolić Ci na zaoszczędzenie czasu i pracy. Czasami wystarczy tylko dopisać trochę arkuszy CSS i dodać elementy wizualne, aby motyw zaczął wyglądać tak, jak sobie zaplanowałeś. W innych przypadkach trzeba utworzyć całkiem nową gałąź i stworzyć całkiem nowy własny motyw. Na przykład szwedzki blog Nutopia (*http://nutopia.se*) jest zbudowany na bazie rdzenia HTML5 oraz składa się głównie z CSS i elementów wizualnych (rysunek B.8).



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

Rysunek B.8. Szwedzki blog Nutopia

Bez względu na to, jaki projekt zamierzasz zrealizować, pamiętaj, że wybór odpowiedniego motywu bazowego pozwoli Ci zaoszczędzić dużo czasu i pracy. Nawet jeśli będziesz musiał utworzyć własny motyw bazowy, to wyjdzie Ci to na dobre.

Czas zacząć tworzyć i publikować. W końcu po to jest WordPress!



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# Skorowidz


## A

archive.php, 114
attachment.php, 120

## B

baza danych, 23
  kasowanie widżetów, 31
  kopia zapasowa, 33
  masowa edycja wpisów, 32
  modyfikowanie, 30
  ograniczanie uprawnień użytkownika, 39
  struktura, 31
  tabele, 32
  ustawienia, 26
  wersje wpisów, 27
  wtyczki, 216
    odinstalowywanie, 209
  zewnętrzny serwer, 26
  zmiana hasła użytkownika, 32

## C

category.php, 47, 120
CMS, 231
  funkcje społecznościowe, 232
  menu, 242
  modularność, 232
  pomoc, 232
  widżety, 240
  własne
    taksonomie, 240
    typy wpisów, 239
  WordPress, 232, 238
  WP-CMS Post Control, 235
  wybór, 233
Codex, 44
comments.php, 46, 50, 93, 109, 130
content.php, 110
content-single.php, 112

## F

Facebook, 246
  Facebook Connect, 254
  Get Code, 247
  Lubię to, 246
  narzędzia, 246
  widżet, 248
Flickr, 288
  pokaz slajdów, 290
  publikowanie zdjęć, 288
footer.php, 46, 50, 92, 93, 100
formaty wpisów, 72, 282
functions.php, 47, 67, 73, 117, 124, 125, 126, 127, 128, 146, 152, 168, 170, 171, 172, 191, 221, 223, 225
funkcja
  add_action(), 173, 197, 236
  add_filter(), 127, 198
  add_help_tab(), 204
  add_image_size(), 169, 324
  add_meta_box(), 166
  add_options_page(), 205
  add_post_meta(), 167
  add_settings_field(), 206
  add_settings_section(), 206
  add_theme_support(), 73, 74, 169, 324
  admin_url(), 305
  bloginfo(), 99, 182
  body_class(), 261
  comment_class(), 261
  comments_template(), 109
  dashboard_reminder(), 215
  delete_post_meta(), 167
  do_action(), 173
  do_settings_sections(), 206
  do_shortcode(), 302
  esc_attr(), 207
  esc_url(), 336
  fetch_feed(), 300


ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

funkcja
form(), 212
get_field_id(), 214
get_field_name(), 214
get_footer(), 105
get_header(), 105
get_option(), 207
get_post(), 285
get_post_format(), 74
get_sidebar(), 105
get_stylesheet_ directory_uri(),150, 271
get_template_part(), 111, 148
get_transient(), 299
has_post_format(), 338
load_textdomain(), 199
mt_rand(), 284
next_image_link(), 279
next_post_link(), 278
post_class(), 258, 260, 261, 326
previous_image_link(), 279
previous_post_link(), 278
query_posts(), 75, 77, 124, 202
rand(), 284
register_nav_menus(), 118
register_post_type(), 202
register_sidebar(), 128, 295
register_taxonomy(), 200
register_widget(), 212
remove_action(), 174, 198
remove_filter(), 198
remove_meta_box(), 235
rewind_posts(), 79
set_transient(), 299
settings_fields(), 205
simpleblog_ load_scripts(), 118
simpleblog_ themesetup(), 118
simpleblog_register_menus(), 118
simpleblog_register_sidebars(), 118
simpleblog_theme_setup(), 118
smashing_post_demo_meta_box(), 167
smashing_register_sidebars(), 128
smashing_rss_promotion(), 302
smashing_text_example(), 301
smashings_settingsdemo_do_page(), 205
smashings_settingsdemo_init(), 206
smashingshortcode, 197
smashingtheme_setup(), 74
SmashingWidget(), 212
strip_tags(), 213
superfunction(), 174
the_content, 127
the_date(), 109
the_excerpt(), 336
the_post_thumbnail(), 169
update(), 212, 213
update_post_meta(), 167
widget(), 212
wp_add_dashboard_widget(), 215
wp_admin_css_color(), 273
wp_editor(), 315
wp_enqueue_script(), 98, 252, 269, 294
wp_enqueue_style(), 226, 272, 307
wp_footer(), 93, 101, 172
wp_head(), 93, 97, 98, 172, 197
wp_header(), 101
wp_list_comments(), 114, 133
wp_login_form(), 305
wp_mail(), 303
wp_nav_menu(), 99, 101, 265
wp_register_script(), 270
wp_register_style(), 307
wp_reset_postdata(), 79, 81, 84
wp_tag_cloud(), 318
WPLANG, 25



## G

galeria, 276
lightbox, 280
strona
ustawień mediów, 276
załącznika, 276
stylizowanie, 278
tworzenie, 276
wyświetlanie losowych obrazów, 285
gettext GNU, 181
GlotPress, 181
Google, 252
przycisk +1, 252
WordPress, 252

## H

haki, 153
add_meta_boxes, 235
admin_init, 206, 273
admin_menu, 205
after_setup_theme, 118, 168, 179, 173, 181
akcji, 171, 197
dodawanie akcji, 173
excerpt_length, 64

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

filtrów, 197
lista haków, 172
login_head, 271
odłączanie akcji, 174
post_class, 261
the_content, 126, 174
the_excerpt, 197
the_title, 197
tworzenie własnych, 173
user_register, 304
używanie, 172
widgets_init, 128, 236
wp_enqueue_script, 252, 307
wp_footer, 172
header.php, 46, 50, 92, 93, 96
HTML5, 94

## I

ikony wpisów, 169
image.php, 120
index.php, 30, 46, 50, 92, 93, 103, 107, 109, 236
instalacja, 22
formularz z danymi witryny, 24
instalatory, 28
interfejs instalatora, 23
klucze
tajne, 25
uwierzytelniania, 25
multisite, 28
przycisk
Wyślij, 23
Zainstaluj WordPressa, 23
ręczna, 23
serwer baz danych, 26
ustawianie
adresu URL, 27
ścieżki, 27
w podfolderze, 29
wersje wpisów, 27
wp-config-sample.php, 24
zmiana języka, 25
z kreatorem, 22
instrukcja
echo, 150, 180, 260
endif, 61
endwhile, 60
have_posts(), 62
if, 98, 127
return, 180
Template, 145
while, 60

## J

JavaScript, 268
biblioteki, 268
rejestrowanie skryptów, 270

## K

kanały RSS, 181, 245, 297
mieszanie zawartości kanałów, 300
parser kanałów, 298
tworzenie własnego, 183
wtyczki, 355
wyświetlanie, 297
kanały subskrypcji, 182
klucze
tajne, 25, 39
uwierzytelniania, 25
komentarze, 130, 253
comments.php, 130
projektowanie, 130
w wątkach, 131
wtyczki, 353
wyróżnienie autora wpisu, 133
zakorzenianie, 132
zarządzanie, 353
zewnętrzny system komentarzy, 254

## L

lightbox, 280
linki partnerskie, 319
loop.php, 48, 61
loop-category.php, 62
loop-index.php, 61
loop-single.php, 65, 67

## M

media społecznościowe, 245, 256
Facebook, 246
Google, 252
Twitter, 248
wtyczki, 354
menu
CMS, 242
motywy, 262
Narzędzia, 34



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

menu
  przesuwane drzwi, 263
  rozwijane, 265
  ulepszanie, 262
  Ustawienia/Bezpośrednie odnośniki, 29
  własne, 170
  WordPress, 263
metadane, 164
  moduł meta, 166, 168
motyw, 46, 91, 118
  bazowy, 134, 153
  Bones, 369
  budowa, 92
  Constellation, 367
  deklarację motywu, 95
  elementy promocyjne, 126
  kategorie, 157
  komentarze, 130
    obsługa, 93
  komercyjny, 136
  kontrolowanie, 56
  lista klas, 162
  menu, 262
    własne, 170
  modyfikowanie panelu administracyjnego, 272
  nadrzędny, 144
  nagłówek, 92, 96
  Notes Blog, 145
  obszary
    widżetów, 103
    właściwej treści, 92
  pasek boczny, 92, 102
  pętla, 60
  planowanie, 156
  pliki szablonowe, 46, 118
  prosty projekt bloga, 93
  przyspieszanie działania, 185
  publikowanie, 136
  Roots, 365
  Spectacular, 368
  Starkers, 364
  stopka, 92, 100
  strona opcji, 178
  szablony, 46, 144
    archiwów, 114
    stron, 158
  szkieletowy, 153, 362
  tagi, 157, 258
  taksonomie, 157
  techniki stylizacji, 158
  Toolbox, 366
  treść główna, 103
    szerokość, 125
  Twenty Eleven, 93, 329, 363
  Twenty Ten, 93, 363
  Twenty Ten Five, 363
  wiadomość o błędzie, 116
  własne pola, 133, 259
  wpisy
    ikony, 169
    stylizowanie, 159
    wstępy, 65
  wybór, 362
  wyświetlanie wyników wyszukiwania, 116
  zarządzanie, 48
multikanał, 300
multimedia, 275
  ikony, 287
  lightbox, 280
  obrazy nagłówkowe, 287
  osadzanie treści multimedialnej, 283
  pokaz slajdów, 290
  publikowanie zdjęć, 287
  tworzenie galerii obrazów, 276
  ustawienia treści osadzonych, 283
  wtyczki, 347
  wyświetlanie losowych obrazów, 284
multisite, 28, 193

## O

obszar widżetów, 103
  deklarowanie, 128
  tworzenie, 118
oEmbed, 284
opcje
  echo, 55
  Indexes, 40
  multisite, 28
  WP_CONTENT_URL, 27
  WP_DEBUG, 28, 139
  WP_DEBUG_DISPLAY, 28
  WP_DEBUG_LOG, 28
  WP_HOME, 27
OpenID, 254



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

## P


page.php, 46, 111, 112, 120, 121
parametr
  author, 183
  capability_type, 322
  cat, 76
  day, 183
  echo, 52
  exclude, 52
  format, 52
  hour, 183
  include, 52
  keyword, 183
  large, 169
  largest, 52
  link, 52
  medium, 169
  minute, 183
  monthnum, 183
  number, 52
  order, 52
  orderby, 52
  p, 183
  pagedcat, 76
  posts_per_page, 77
  post-thumbnails, 169
  public, 322
  second, 183
  separator, 52
  smallest, 52
  tag, 76
  taxonomy, 52
  thumbnail, 169
  topic_count_text_callback, 52
  unit, 52
  year, 183
pasek boczny, 102, 148
pętla, 59, 60
  else, 61
  endif, 61
  endwhile, 60
  powiadomienie o błędzie, 60
  promowanie produktów, 327
  struktura, 60
  tworzenie wielu pętli, 79
  while, 60
  klasa WP_Query, 62
  wpisy, 64
    przyklejanie, 69
  wstawianie reklam, 266
  wypisy, 63
  zapisywanie, 61
PHP, 44, 49
  pętla, 59
  tagi
    dołączania plików, 49
    szablonowe, 49
  wtyczki, 197
phpMyAdmin, 35, 38
pliki
  .htaccess, 40, 45
  .mo, 179
  .po, 179
  footer.php, 46, 50, 92, 93, 100
  functions.php, 47, 67, 73, 117, 124, 125, 126, 127, 128, 146, 152, 168, 170, 171, 172, 191, 221, 223, 225
  językowe, 179, 180
  loop.php, 48, 61
  loop-category.php, 62
  loop-index.php, 61
  loop-single.php, 65, 67
  POT, 179
  print.css, 306
  sidebar.php, 46, 50, 92, 102, 128
  style.css, 69, 92, 145, 146
  szablonowe, 95, 118
    404.php, 120
    archive.php, 120
    attachment.php, 120
    category.php, 120
    comments.php, 113
    content.php, 110
    content-single.php, 112
    front-page.php, 120
    header.php, 96
    hierarchia, 121
    image.php, 120
    index.php, 30, 46, 50, 92, 93, 103, 107, 109, 236
    motywy potomne, 146
    page.php, 46, 111, 112, 120, 121
    search.php, 46, 116, 120
    single.php, 46, 65, 111, 120, 160
    single-attachment.php, 120
    style.css, 69, 92, 95, 145, 146
    szablony stron, 122
    taxonomy.php, 120, 174
    text.php, 120
    video.php, 120




ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

pliki
uninstall.php, 210
wp-blog-header.php, 30
wp-config.php, 26, 27, 33, 38, 40, 193
wp-config-sample.php, 24, 26
zasady używania, 120
Poedit, 180
pola własne, *Patrz* własne pola
print.css, 306
przenośność, 227
przesuwane drzwi, 263
przyciski
+1, 252
Dodaj własne pole, 165
Get Code, 247
Lubię to, 246
Pobierz skrócony odnośnik, 251
Tweetnij, 249
Więcej, 63
Wyślij, 23
Zainstaluj WordPressa, 23
przyklejanie wpisów, 69, 163
dodatkowy obszar nagłówkowy, 163
publikowanie
ogłoszeń, 163
wtyczki, 346



## R

rdzeń, 44
roadblocks, 241

## S

screenshot.png, 117
search.php, 46, 116, 120
SEO, 183
wtyczki, 356
sidebar.php, 46, 50, 92, 102, 128
single.php, 46, 65, 111, 120, 160
skróty kodowe, 210, 224, 301
dodawanie, 301
zagnieżdżanie, 302
strona załącznika, 276
strony błędu 404, 268
strony opcji motywu, 177
style.css, 69, 92, 95, 145, 146
system zarządzania treścią, *Patrz* CMS
szablony, 46
hierarchia, 121
szkielet, 362

## T

tagi
ciąg znaków, 54
dołączania plików, 49
comments_template(), 50
get_calendar(), 54
get_footer(), 50, 93
get_header(), 50, 93
get_sidebar(), 93
get_template_directory_uri(), 49
get_template_part(),50, 61, 62, 65, 74
domyślne parametry chmury tagów, 52
kontrolowanie motywu, 56
liczby całkowite, 54
lista tagów, 49
łańcuch zapytań, 53
metoda funkcyjna, 53
pobieranie treści, 51
sposoby wykorzystania, 258
szablonowe, 49, 63
bloginfo(), 49
body_class(), 72, 99, 161
comments_template(), 93
edit_comment_link(), 51
edit_post_link(), 51
header_image(), 171
in_category(), 258
post_class(), 71, 158, 159, 163
query_posts(), 80
sticky_class(), 71
the_content(), 64, 66, 172
the_date(), 55
the_excerpt(), 64, 65
the_meta(), 165
the_permalink(), 247
the_time(), 55
the_title(), 63
the_title_attribute(), 63
wp_list_comments(), 131
wp_get_attachment_link(), 286
wp_nav_menu(), 170
wp_tag_cloud(), 52, 53
tworzenie własnych, 198
typy danych, 54
wartości logiczne, 54
warunkowe, 55
get_sidebar(), 56
has_post_thumbnail(), 169
is_attachment(), 279

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

is_category(), 56
is_front_page(), 55, 57
is_home(), 77, 119
is_single(), 112, 119
is_sticky(), 71, 163
is_tag(), 258
taxonomy_exists(), 261
taksonomie, 174, 200
terminy, 174
własne, 174, 240, 261, 330
tworzenie, 200
zastosowania, 175
taxonomy.php, 120, 174
text.php, 120
Twitter, 248
Sign In with Twitter, 254
skracanie adresów URL, 251
TweetMeme, 251
Tweetnij, 249
Twitterfeed, 251
widżet, 249
WordPress, 248

## U

uninstall.php, 210

## V

video.php, 120

## W

widżety, 47, 127
CMS, 240
deklarowanie obszarów widżetów, 128
dla wtyczki, 211
dodawanie dynamizmu, 241
Facebook, 248
kokpitu, 214, 216
obszarów
deklarowanie, 128
tworzenie, 118
proces tworzenia, 212
rejestracja, 214
SmashingHello, 212
Tekst, 241
Twitter, 249
zmiana sposobu wyświetlania, 129
własne pola, 133, 157, 160, 259
nagłówki w stylu czasopism, 133
użyteczność, 134
wtyczki, 222
WordPress, 15, 22
API Transients, 216, 299
API ustawień, 204
BuddyPress, 232
buforowanie treści, 299
CMS, 232, 238
dokumentacja, 44
dostosowywanie stylu, 270
drukowanie treści, 306
eksportowanie danych, 35
elementy multimedialne, 275
osadzanie treści, 283
Facebook, 246
formaty wpisów, 72, 282
formularze
logowania, 271, 304
z danymi witryny, 24
funkcje, 293
komentarzy, 130
społecznościowe, 232
Google, 252
import danych, 36
instalacja, 22
Interfejs instalatora, 23
JavaScript, 269
kanały
RSS, 181, 297
subskrypcji, 182
kokpit, 214
komentarze, 253
konfiguracja statycznej witryny, 237
menu, 263
modularność, 232
modyfikowanie bazy danych, 30
motywy, 46
planowanie, 156
potomne, 145
publikowanie, 136
zarządzanie, 48
MU, 193
multikanał, 300
multisite, 28, 193
oEmbed, 284
optymalizacja, 186
panel administracyjny, 235
pętla, 60
pliki szablonowe, 95, 236




ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

WordPress
pola własne, 164
pomoc, 232
promowanie produktów, 326
protokół Atom, 181
przesuwane drzwi, 263
publikowanie
bazy wiedzy, 316
linków partnerskich, 319
motywu, 136
przepisów, 328
recenzji przez użytkowników, 312
wiadomości przez użytkowników, 312
wpisów przez użytkowników, 311
rdzeń, 44
rejestrowania skryptów, 270
rozszerzenie funkcjonalności, 228
SimplePie, 298, 300
sklep internetowy, 320
strony
szablony, 121, 158
ustawień bezpośrednich odnośników, 30
ustawień mediów, 277
stylizowanie galerii, 277
tagi
dołączania plików, 49
szablonowe, 49
taksonomie, 174
TinyMCE, 315
Twenty Eleven, 363
Twenty Ten, 363
Twitter, 248
tworzenie
galerii obrazów, 276
katalogu, 321
strony wpisów, 323
tablicy ogłoszeń, 313
typowy układ bloga, 92
ustawienia
mediów, 125
ścieżki do instalacji, 27
bazy danych, 26
użytkownik z uprawnieniami administratora, 39
wiadomości e-mail, 303
widżety, 47, 211
własne
pola, 133, 157
taksonomie, 240
typy wpisów, 239
komponenty z kartami, 294
wpisy, 61, 258
edycja, 68
formaty, 72, 157, 282
niestandardowego typu, 326
przyklejanie, 69
wersje, 27
wtyczki, 191, 345
identyfikacja, 196
wykonywanie kopii zapasowej, 32
wymuszenie szyfrowania SSL, 40
wypisy, 63
wyświetlanie
proponowanego artykułu, 80
treści zewnętrznych, 242
edytora wpisów, 315
zabezpieczanie, 38
zmiana hasła użytkownika, 32
zmiana hostingu, 34
wp_commentmeta, 31
wp_comments, 31
wp_links, 31
wp_options, 31, 32
wp_postmeta, 31
wp_posts, 31, 32
WP_Query, 62, 79
wp_term_relationships, 31
wp_term_taxonomy, 31
wp_terms, 31
wp_usermeta, 31
wp_users, 31
wp-blog-header.php, 30
wp-config.php, 26, 27, 38, 40, 193
wp-config-sample.php, 24, 26
wp-content, 33, 45
wpisy, 61
chwytliwe wstępy, 65
drukowanie, 306
edycja, 68
formaty, 72, 157, 282
ikony, 169
kategorie, 157
kontrolowanie treści, 258
liczba wyświetlanych wpisów, 77
lista wpisów, 65
niestandardowego typu, 326
nowy typ, 203
początkowy fragment wpisu, 64
przyklejanie, 69, 163
publikowanie przez użytkowników, 311
ramka z najnowszymi wpisami, 79



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

słowa kluczowe, 157
stylizowanie, 159
tagi, 258
taksonomia, 261
tytuł wpisu, 63
wersje, 27
własne pola, 259
własne typy, 176, 202, 239
tworzenie, 329
wstępna segregacja, 157
zwiększenie kontroli, 258, 261
wtyczki, 186, 191
administracyjne
Activate Update Services, 352
bbPress, 351
Broken Link Checker, 349
BuddyPress, 351
Custom Admin Branding, 349
Custom Post Type UI, 352
Disabler, 348
Download Monitor, 350
Editorial Calendar, 352
Fast Secure Contact Form, 349
FeedWordPress, 243, 351
Google Analyticator, 349
Google Analytics for WordPress, 349
Jigoshop, 350
Members, 352
Members Only, 351
More Fields, 350
Pods, 350
Post Editor Buttons, 352
Pretty Link Lite, 350
Random Redirect, 352
Redirection, 350
Revision Control, 352
Theme My Login, 349
TinyMCE Advanced, 352
Viper's Video Quicktags, 352
Widget Context, 352
Woopra Analytics Plugin, 352
WordPress.com Stats, 349
WP Bannerize, 351
WP e-Commerce, 349
WP Event Ticketing, 352
WP Mail SMTP, 352
WP Maintenance Mode, 348
WP No Category Base, 348
WP-DB-Backup, 348
advanced-cache.php, 192
aktywacja dla całej sieci, 194
argumenty priorytetu, 198
bazy danych, 216
Blog Time, 358
blog-deleted.php, 192
blog-inactive.php, 192
blog-suspended.php, 192
blok identyfikacyjny wtyczki, 195
budowa, 195
db.php, 192
db-error.php, 192
do publikowania treści, 346
GD Star Rating, 347
Polldaddy, 347
WordPress Popular Posts, 346
WP Greet Box, 346
WP-Polls, 347
WP-PostRatings, 347
Yet Another Related Posts Plugin, 346
do rdzenia, 192
dodawanie funkcjonalności, 198
elementy wizualne, 204
funkcje nadpisujące, 199
identyfikacja w WordPressie, 196
informacja o licencji, 195
install.php, 192
instrukcja obsługi, 204
kanały RSS
Align RSS Images, 355
Disable RSS, 356
MobilePress, 356
RSS Footer, 356
Subscribe2, 356
WordPress Mobile Edition, 356
WordPress Mobile Pack, 356
WPtouch, 356
licencja, 204
maintenance.php, 192
media społecznościowe, 354
Lifestream, 354
SexyBookmarks, 355
Share Buttons by Lockerz / AddToAny, 355
ShareThis, 355
Simple Social Bookmarks, 355
Sociable, 355
Tweet Old Post, 355
Twitter for WordPress, 355
Twitter Tools, 354
Wickett Twitter Widget, 355



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

wtyczki
metody inkorporowania, 197
multimedialne, 347
Featured Articles Lite, 348
Lightbox Gallery, 347
Podcasting, 348
Slimbox, 348
na funkcje, 225
object-cache.php, 192
obowiązkowe, 192, 193
odinstalowywanie, 209
One Quick Post, 313
PHP Snippets, 359
plik PHP, 204
Post From Site, 311
przenośność, 227
publikowanie, 217
Query Posts, 359
rodzaje, 192
rozszerzanie funkcjonalności, 222
SEO
All in One SEO Pack, 357
Better Search, 357
Breadcrumb Trail, 358
GD Press Tools, 358
Global Translator, 357
Google XML Sitemaps, 357
HeadSpace2 SEO, 357
Robots Meta, 357
Search Everything, 358
WordPress SEO by Yoast, 357
skróty kodowe, 210
Smashing Post Type, 203
strona ustawień, 209
sunrise.php, 192
SyntaxHighlighter Evolved, 358
taksonomie, 200
Theme Check, 139
tworzenie, 193
tworzenie widżetów, 211
ustawienia, 204
używanie haków, 197
Widget Logic, 358
własne pola, 222
WP Super Cache, 359
WP-Cirrus, 358
WP-CMS Post Control, 235
WP-DBManager, 359
wpisy, 202
WP-PageNavi, 359
WP-Typography, 358
Your Classified Ads, 314
zgodność wsteczna, 217
zwykłe, 192

## Z

znacznik warunkowy, 95



ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974

# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

**1. ZAREJESTRUJ SIĘ**
**2. PREZENTUJ KSIĄŻKI**
**3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**
http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA


ebookpoint kopia dla: Pawel Domanski pawel@dexterteam.eu A1219139974