Helion

SMASHING MAGAZINE

Thord Daniel Hedengren

PODRĘCZNIK WordPressa

SMASHING MAGAZINE

Najlepszy podręcznik o WordPressie!

Tytuł oryginału: Smashing WordPress: Beyond the Blog

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-6678-2

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
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

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.

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.

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! */`:

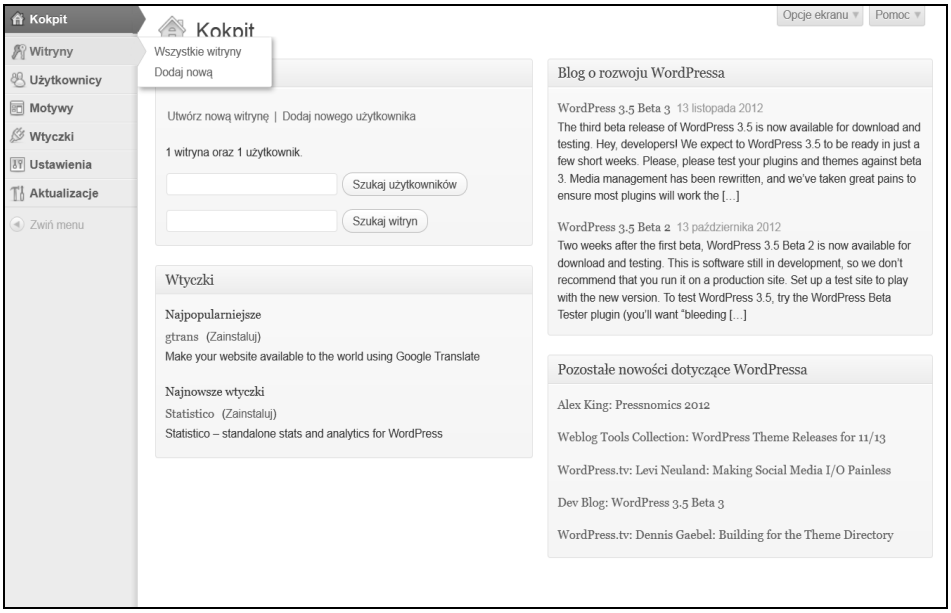
```
define( 'WP_ALLOW_MULTISITE', true );
```

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



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.

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.
```

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.

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

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

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.

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
        )
    );

}
```

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
        )
    );

}
```

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()`:

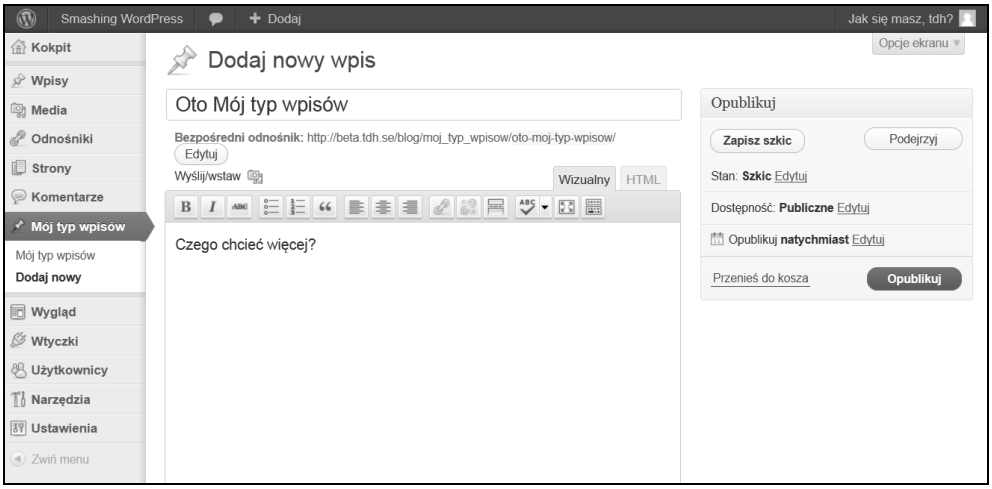
```
// Rejestracja nowego typu wpisów
register_post_type( 'moj_typ_wpisow',
    array(
         'labels' => array(
             'name' => 'Mój typ wpisów'
        ),
         'singular_label' => 'Mój typ wpisów',
          'public' => true,
          'show_ui' => true,
          'capability_type' => '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.



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,
```

```
            '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.

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()`

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>';
}
```

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;
}
```

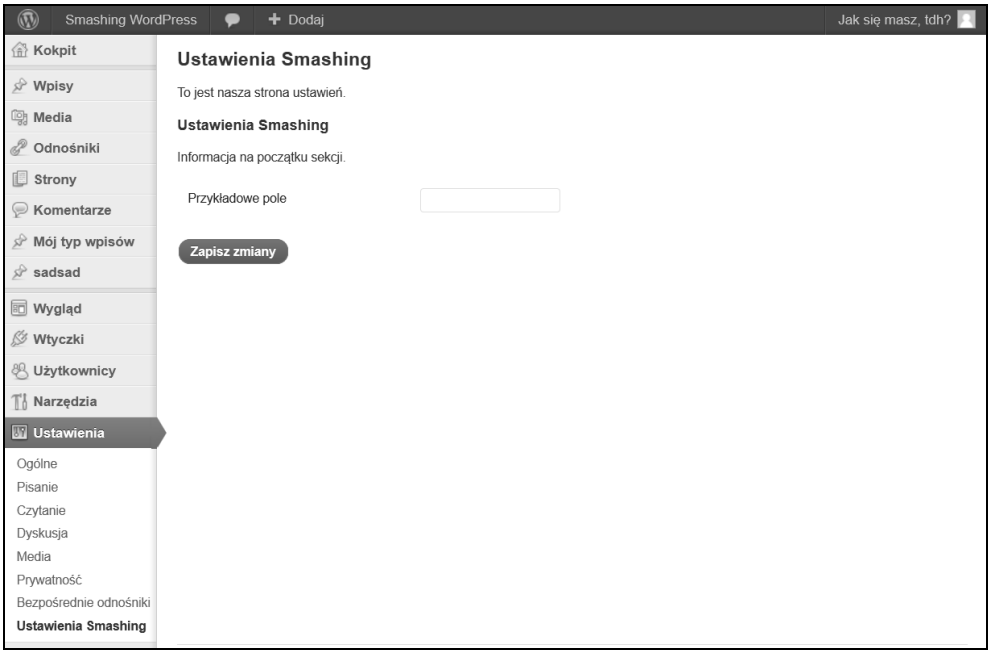
Co robi ten kod? Utworzyliśmy stronę ustawień zawierającą pola ustawień tworzone za pomocą funkcji `settings_fields()` (w tym przykładzie tylko jedno, ale można dodać więcej) i sekcje utworzone za pomocą funkcji `do_settings_section()`. Poniżej znajduje się kompletny kod wtyczki.

```
<?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',
```

```
        '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() {
```

```
// 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.

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

`[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
```

```
    }
    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?
   ```

```
        <?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>
```

```
        </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:

```
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.

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*.

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;

- 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.

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*.

Skorowidz

G

H

I

J

K

L

M

0

P

R

S

T

U

V

W

Z

KOMPLETNE I PRZYJAZNE ŹRÓDŁO INFORMACJI O PLATFORMIE WORDPRESS!

Dwóch gigantów postanowiło połączyć swoje potencjały. Razem stworzyli niezwykłą książkę, którą właśnie trzymasz w rękach. Jednym z nich jest Smashing Magazine, należący do najpopularniejszych serwisów poświęconych tworzeniu stron WWW i nie tylko, a drugim WordPress, lider wśród systemów do prowadzenia blogów. Obok tej pozycji nie możesz przejść obojętnie!

W tej niepowtarzalnej i wyśmienitej książce znajdziesz zbiór najświeższych, najciekawszych i najlepszych informacji poświęconych systemowi WordPress. W czasie lektury dowiesz się, jak używać tej platformy jako systemu CMS oraz narzędzia do blogowania czy projektowania witryn dowolnego typu. Ponadto nauczysz się tworzyć genialne motywy, przydatne wtyczki oraz integrować Wordpressa z portalami społecznościowymi. Szczególną uwagę powinieneś zwrócić na zagadnienia poświęcone SEO — przestrzeganie dobrych zasad z pewnością ułatwi Ci zaistnienie w sieci. Książka ta jest obowiązkową pozycją dla każdego użytkownika platformy WordPress!

Tylko krok dzieli Cię od:

- szczegółowego poznania platformy Wordpress
- przygotowania własnego motywu graficznego
- poprawnej konfiguracji, zgodnej z SEO
- odniesienia sukcesu w sieci WWW

WILEY

Helion

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-6678-2
9 788324 666782
Cena: 59,00 zł