Opanuj niesamowite możliwości tego CMS-a
i twórz rozbudowane witryny
oraz aplikacje internetowe!

- Poznaj architekturę Drupala i sposób jego działania
- Naucz się pracować z bazą danych i API formularzy
- Twórz własne moduły i rozszerzaj możliwości tego systemu

Drupal 7

Zaawansowane programowanie

Wydanie III

Todd Tomlinson, John K. VanDyk

Helion

Apress

helion.pl

Idź do

- Spis treści
- Przykładowy rozdział
- Skorowidz

Katalog książek

- Katalog online
- Zamów drukowany katalog

Twój koszyk

- Dodaj do koszyka

Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

Czytelnia

- Fragmenty książek online

Drupal 7. Zaawansowane programowanie



Autorzy: Todd Tomlinson, John K. VanDyk
Tłumaczenie: Krzysztof Rychlicki-Kicior
ISBN: 978-83-246-3367-8
Tytuł oryginału: Pro Drupal 7 Development
Format: 168×237, stron: 528

Opanuj niesamowite możliwości tego CMS-a i twórz rozbudowane witryny oraz aplikacje internetowe!

- Poznaj architekturę Drupala i sposób jego działania
- Naucz się pracować z bazą danych i API formularzy
- Twórz własne moduły i rozszerzaj możliwości tego systemu
- Zobacz, jak optymalnie wykorzystać jQuery w Drupalu

Drupal to nie tylko kolejny, nieco lepszy od innych CMS. Ten system zarządzania treścią zrobił ostatnio oszałamiającą karierę jako doskonały framework do tworzenia wszelkiej maści aplikacji internetowych. Dzięki niezwykłej łatwości i szybkości, z jaką Drupal pozwala programistom tworzyć rozbudowane blogi, profesjonalne witryny korporacyjne czy serwisy społecznościowe, zainteresowanie tą technologią zaczęło dynamicznie wzrastać. Co więcej, choć już dziś wydaje się, że sposoby wykorzystania tego CMS-a ogranicza jedynie ludzka wyobraźnia, system jest wciąż intensywnie rozwijany przez liczną społeczność entuzjastów na całym świecie. Możliwość czerpania z ogromnych zasobów wiedzy tysięcy programistów poprawi jakość Twoich nawet najbardziej wyrafinowanych internetowych projektów.

Jeśli chcesz tworzyć nowoczesne, rozbudowane witryny internetowe z wykorzystaniem Drupala 7, właśnie znalazłeś idealny podręcznik dla siebie! Omówiono tu wszystko, co będzie Ci potrzebne, począwszy od podstawowych zagadnień, takich jak architektura i struktura plików systemu, przez sposoby wykorzystania API formularzy czy dodawania własnych modułów w celu rozszerzania możliwości, aż po metody tworzenia bezpiecznego, wydajnego kodu. Dowiesz się więcej na temat pracy z bazą danych, uruchomienia własnej strony wyszukiwarki oraz korzystania z jQuery w Drupalu. Nauczysz się także sprawnie optymalizować Drupal i przeprowadzać skuteczne testy oraz poznasz wiele innych praktycznych zagadnień, które sprawią, że bez trudu zrozumiesz zasady działania tego CMS-a oraz pokonasz wszelkie problemy napotykane w trakcie pracy.

Z tej książki dowiesz się między innymi:

- jak działa Drupal i jaką ma architekturę
- jak tworzyć dodatkowe, funkcjonalne moduły
- jak pracować z bazą danych i API formularzy
- jak wygląda obsługa użytkowników, segmentów i pól
- jak przechowywać dane w sesjach
- jak zapewnić wyszukiwanie i indeksowanie informacji
- jak wykorzystać jQuery w Drupalu
- jak tworzyć czysty, bezpieczny kod
- jak optymalizować działanie Drupala
- jak przeprowadzić efektywny proces testowania

Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl
© Helion 1991–2011

Spis treści

Przedmowa **15**

O autorach **17**

O korektorach merytorycznych **19**

Podziękowania **21**

Wstęp **23**

Rozdział 1. Jak działa Drupal? **25**

Czym jest Drupal? 25
Stos technologii 25
Rdzeń 26
Interfejs administratora 26
Moduły 27
Haki 27
Skórki 28
Segmenty 28
Pola 29
Bloki 29
Układ plików 29
Obsługa żądania 31
Zadania serwera WWW 31
Proces rozruchu (bootstrap) 32
Przetwarzanie żądania 32
Wyświetlanie danych 32
Podsumowanie 32

Rozdział 2. Tworzenie modułów **33**

Tworzenie plików 33
Implementacja haka 35
Dodawanie ustawień charakterystycznych dla modułu 36
Tworzenie własnej sekcji administracyjnej 42
Wyświetlanie formularza ustawień 43
Walidacja ustawień wysłanych przez użytkownika 44

Przechowywanie ustawień ... 45
Tabela variables w akcji ... 45
Wczytywanie wartości za pomocą funkcji variable_get() ... 46
Co dalej? ... 46
Podsumowanie ... 47

Rozdział 3. Haki, działania i wyzwalacze ... 49
Jak działają zdarzenia i wyzwalacze? ... 49
Działania w praktyce ... 51
Interfejs użytkownika wyzwalaczy ... 51
Twoje pierwsze działanie ... 52
Przypisywanie akcji ... 53
Zmiana wyzwalaczy wywołujących działanie ... 53
Kontekst a działania ... 58
Przygotowanie kontekstu przez moduł wyzwalaczy ... 58
Modyfikacja działań przy użyciu funkcji action_info_after() ... 59
Określanie kontekstu ... 59
Metody przechowywania działań ... 61
Tabela actions ... 61
Identyfikatory działań ... 61
Bezpośrednie wywołanie działania przy użyciu metody actions_do() ... 62
Tworzenie własnych wyzwalaczy przy użyciu hook_trigger_info() ... 62
Dodawanie wyzwalaczy do istniejących haków ... 65
Podsumowanie ... 67

Rozdział 4. System menu ... 69
Mapowanie wywołań zwrotnych ... 69
Mapowanie adresów URL na funkcje ... 69
Tworzenie elementu menu ... 71
Argumenty wywołań zwrotnych strony ... 74
Wywołania zwrotne strony w innych plikach ... 75
Dodawanie łącza do bloku Nawigacja ... 76
Zagnieżdżanie menu ... 77
Kontrola dostępu ... 78
Lokalizacja i dostosowanie ... 80
Tworzenie wywołania zwrotnego tytułu ... 80
Znaki wieloznaczne w elementach menu ... 81
Podstawowe znaki wieloznaczne ... 81
Znaki wieloznaczne i parametry wywołań zwrotnych strony ... 82
Korzystanie z wartości dopasowanej przez znak wieloznaczny ... 82
Znaki wieloznaczne a zamiana parametrów ... 82
Przekazywanie dodatkowych argumentów do funkcji ładowania ... 84
Specjalne argumenty ładowania: %map i %index ... 84
Tworzenie ścieżek na podstawie znaków wieloznacznych z wykorzystaniem funkcji to_arg() ... 85
Znaki wieloznaczne i funkcja to_arg(): przypadki specjalne ... 85
Zastępowanie elementów menu w innych modułach ... 86
Zastępowanie hiperłączy menu w innych modułach ... 87
Rodzaje elementów menu ... 87

Typowe zadania ... 89
Przypisywanie wywołań zwrotnych bez dodawania hiperłączy do menu ... 89
Wyświetlanie elementów menu jako zakładek ... 89
Ukrywanie istniejących elementów menu ... 91
Korzystanie z menu.module ... 91
Typowe błędy ... 91
Podsumowanie ... 92

Rozdział 5. Praca z bazą danych ... 93
Określanie parametrów bazy danych ... 93
Jak działa warstwa abstrakcji bazy danych? ... 94
Łączenie się z bazą danych ... 94
Wykonywanie prostych zapytań ... 95
Pozyskiwanie wyników zapytań ... 97
Pobieranie pojedynczej wartości ... 97
Pobieranie wielu wierszy ... 97
Twórca zapytań i obiekty zapytań ... 97
Pobieranie ograniczonego zbioru wyników ... 98
Stronicowanie wyników zapytań ... 98
Przykłady typowych zapytań ... 99
Modyfikacje rekordów za pomocą funkcji drupal_write_record() ... 100
API schematów ... 101
Plik .install a schematy ... 101
Tworzenie tabel ... 101
Wykorzystywanie modułu schema ... 103
Mapowanie typów pól schematów na typy pól bazy danych ... 104
Deklarowanie wybranych typów kolumn za pomocą funkcji mysql_type ... 106
Zarządzanie tabelami ... 108
Usuwanie tabel przy deinstalacji modułu ... 108
Zmiana istniejących schematów za pomocą funkcji hook_schema_alter() ... 109
Zmiana zapytań istniejących w innych modułach za pomocą haka hook_query_alter() ... 110
Obsługa wielu baz danych w Drupalu ... 110
Tabele tymczasowe ... 112
Tworzenie własnego sterownika bazy danych ... 112
Podsumowanie ... 113

Rozdział 6. Obsługa użytkowników ... 115
Obiekt $user ... 115
Sprawdzanie stanu logowania użytkownika ... 116
Wprowadzenie do haków użytkownika ... 116
Sposób działania funkcji hook_user_view($account, $view_mode) ... 118
Proces rejestracji użytkownika ... 120
Zbieranie informacji o użytkownikach za pomocą modułu profile.module ... 122
Proces logowania ... 123
Dodawanie danych do obiektu $user w momencie ładowania ... 124
Kategorie informacji o użytkownikach ... 125
Logowanie zewnętrzne ... 126
Podsumowanie ... 129

Rozdział 7. Obsługa segmentów 131
Segment — co to takiego? 131
Nie wszystko jest segmentem 133
Tworzenie modułu segmentu 133
Tworzenie pliku .install 134
Tworzenie pliku .info 136
Tworzenie pliku .module 136
Udostępnianie informacji o rodzaju zawartości 136
Zmiana wywołań zwrotnych menu 137
Tworzenie uprawnień zależnych od typu segmentu za pomocą haka hook_permission() 138
Ograniczenie dostępu do typu segmentu za pomocą haka hook_node_access() 139
Dostosowywanie formularza segmentów do naszego typu segmentu 139
Walidacja pól za pomocą haka hook_validate() 140
Zapisywanie danych za pomocą haka hook_insert() 141
Aktualizowanie danych i interakcja z hakiem hook_update() 141
Usuwanie segmentów za pomocą haka hook_delete() 142
Modyfikowanie segmentów naszego typu za pomocą haka hook_load() 142
Hak hook_view() 142
Modyfikowanie segmentów, które nie należą do naszego typu za pomocą haków w postaci hook_node_xxxxx() 144
Sposób przechowywania segmentów 145
Tworzenie typu segmentu z własnymi typami zawartości 147
Ograniczanie dostępu do segmentów 147
Określanie uprawnień do segmentów 147
Proces obsługi dostępu segmentów 149
Podsumowanie 149

Rozdział 8. Obsługa pól 151
Tworzenie rodzajów zawartości 151
Dodawanie pól do rodzaju zawartości 153
Tworzenie własnego pola 156
Dodawanie pól z poziomu kodu 164
Podsumowanie 167

Rozdział 9. System skórek 169
Skórki 169
Instalacja gotowej skórki 169
Tworzenie skórki 170
Plik .info 176
Dodawanie obszarów do skórki 176
Dodawanie do skórki plików CSS 176
Dodawanie skryptów języka JavaScript 177
Dodawanie ustawień do skórki 177
Pliki szablonów — co, gdzie, jak 179
Zarys ogólny 179
Plik html.tpl.php 181
Przesłanianie plików szablonów 188
Pozostałe pliki szablonów 191
Przesłanianie skórkowalnych elementów 192

Przesłanianie plików szablonów ... 194
Dodawanie i modyfikowanie zmiennych szablonu ... 195
Moduł twórcy skórek ... 196
Podsumowanie ... 196

Rozdział 10. Bloki ... 197
Czym jest blok? ... 197
Opcje ustawień bloków ... 199
Rozmieszczenie bloków ... 199
Tworzenie bloku ... 200
Wykorzystywanie haków bloku ... 202
Tworzenie bloku ... 202
Włączanie bloku w zainstalowanych modułach ... 207
Przykłady widoczności bloków ... 208
Wyświetlanie bloku tylko dla użytkowników zalogowanych ... 208
Wyświetlanie bloku tylko dla użytkowników anonimowych ... 208
Podsumowanie ... 208

Rozdział 11. API formularzy ... 209
Przetwarzanie formularzy ... 209
Rozpoczęcie procesu ... 211
Określanie tokenu ... 211
Określanie ID ... 211
Pobieranie dostępnych definicji elementów formularza ... 211
Poszukiwanie funkcji walidacji ... 212
Poszukiwanie funkcji wysyłania formularza ... 212
Modyfikowanie formularzy przed ich utworzeniem ... 213
Tworzenie formularzy ... 213
Modyfikowanie formularzy po ich utworzeniu ... 213
Sprawdzenie, czy formularz został wysłany ... 213
Poszukiwanie funkcji skórki dla formularza ... 213
Modyfikowanie formularzy przed renderowaniem ... 214
Renderowanie formularza ... 214
Walidacja formularza ... 214
Wysyłanie formularza ... 215
Przekierowanie użytkownika ... 215
Tworzenie prostych formularzy ... 216
Właściwości formularza ... 218
Identyfikatory formularzy ... 218
Zbiory pól ... 219
Stosowanie skórek do formularzy ... 221
Wybór funkcji walidacji i przetwarzania za pomocą haka hook_forms() ... 224
Porządek wywołań funkcji skórek, walidacji i przetwarzania ... 225
Tworzenie funkcji walidacji ... 225
Wykorzystywanie zmiennej $form_state do przekazywania danych ... 227
Proces odbudowania formularza ... 228
Tworzenie funkcji przetwarzania ... 229
Modyfikowanie formularzy za pomocą haka hook_form_alter() ... 229
Przetwarzanie formularzy z poziomu kodu za pomocą funkcji drupal_form_submit() ... 230
Formularze dynamiczne ... 231

Właściwości API formularzy ... 237
Właściwości formularza ... 237
Właściwości dodawane do wszystkich elementów ... 237
Właściwości dostępne we wszystkich elementach ... 238
Elementy formularza ... 240
Podsumowanie ... 252

Rozdział 12. System filtrów ... 255
Filtry ... 255
Filtry i formaty tekstu ... 256
Instalacja filtru ... 258
Zastosowania filtrów ... 259
Tworzenie własnego filtru ... 259
Implementacja haka hook_filter_info() ... 260
Funkcja przetwarzania ... 260
Funkcja pomocnicza ... 261
Podsumowanie ... 263

Rozdział 13. Wyszukiwanie i indeksowanie informacji ... 265
Tworzenie własnej strony wyszukiwarki ... 265
Domyślny formularz wyszukiwarki ... 266
Formularz wyszukiwania zaawansowanego ... 266
Dodawanie formularza wyszukiwania ... 267
Wykorzystywanie indeksera HTML ... 269
Zastosowania indeksera ... 270
Jak działa indekser? ... 270
Podsumowanie ... 277

Rozdział 14. Obsługa plików ... 279
Metody udostępniania plików przez Drupal ... 279
Zarządzane i niezarządzane API Drupala ... 279
Pliki publiczne ... 280
Pliki prywatne ... 281
Ustawienia PHP ... 281
Obsługa wysyłanych plików ... 282
Pole wysyłania pliku ... 282
Wideo i audio ... 283
API plików ... 283
Schemat bazy danych ... 283
Typowe zadania i funkcje ... 284
Haki uwierzytelniania do pobierania ... 292
Podsumowanie ... 293

Rozdział 15. Obsługa kategorii ... 295
Struktura kategorii ... 295
Tworzenie słownika ... 295
Tworzenie terminów ... 296
Powiązanie słownika z rodzajem zawartości ... 296

Rodzaje kategorii ... 296
Płaska ... 296
Hierarchiczna ... 297
Wielohierarchiczna ... 298
Przeglądanie treści przy użyciu terminów ... 299
Wykorzystywanie operatorów AND i OR w adresach URL ... 299
Automatyczne kanały RSS ... 300
Przechowywanie kategorii ... 301
Słowniki budowane w modułach ... 302
Tworzenie słownika w obrębie modułu ... 302
Informowanie o zmianach w słowniku — haki kategorii ... 302
Typowe zadania ... 304
Wyświetlanie terminów taksonomii związanych z segmentem ... 304
Tworzenie własnych zapytań kategorii ... 304
Funkcja taxonomy_select_nodes() ... 304
Funkcje taksonomii ... 305
Pozyskiwanie informacji o słownikach ... 305
Dodawanie, modyfikacja i usuwanie słowników ... 305
Pobieranie informacji o terminach ... 306
Dodawanie, modyfikacja i usuwanie terminów ... 307
Pobieranie informacji o hierarchii terminów ... 307
Wyszukiwanie segmentów oznaczonych wybranymi terminami ... 309
Dodatkowe informacje ... 309
Podsumowanie ... 310

Rozdział 16. Obsługa pamięci podręcznej ... 311
Zastosowania pamięci podręcznej ... 311
Sposób działania pamięci podręcznej ... 312
Wykorzystywanie pamięci podręcznej w obrębie rdzenia ... 313
System menu ... 313
Pamięć podręczna a przefiltrowany tekst ... 314
Zmienne administracyjne i ustawienia modułów ... 314
Bloki ... 317
API pamięci podręcznej w praktyce ... 318
Podsumowanie ... 321

Rozdział 17. Sesje ... 323
Czym są sesje? ... 323
Sposób użycia sesji ... 323
Ustawienia związane z sesją ... 325
Plik .htaccess ... 325
Plik settings.php ... 325
Plik bootstrap.inc ... 325
Wymaganie cookies ... 326
Sposób przechowywania danych w sesji ... 326
Cykl życia sesji ... 327
Konwersacje w trakcie sesji ... 328
Pierwsza wizyta ... 328
Druga wizyta ... 328
Użytkownik zarejestrowany ... 328

Typowe zadania ... 329
Zmiana terminu ważności cookie ... 329
Zmiana nazwy sesji ... 329
Przechowywanie danych w sesji ... 329
Podsumowanie ... 330

Rozdział 18. jQuery w Drupalu ... 331
Czym jest jQuery? ... 331
JavaScript po staremu ... 332
Jak działa jQuery? ... 333
Wykorzystywanie identyfikatorów CSS ... 333
Wykorzystywanie selektora klasy CSS ... 333
jQuery + Drupal = ? ... 334
Twój pierwszy kod jQuery ... 334
Określanie elementu za pomocą ID ... 336
Łańcuchy wywołań ... 337
Dodawanie i usuwanie klas ... 337
Opakowywanie istniejących elementów ... 337
Zmiana wartości elementów CSS ... 338
Umieszczanie kodu JavaScript ... 338
Przesłanialny kod JavaScript ... 341
Tworzymy kontrolkę do głosowania w jQuery ... 343
Tworzenie modułu ... 345
Wykorzystywanie zachowań Drupala ... 351
Możliwości rozszerzenia modułu ... 351
Zgodność ... 352
Co dalej? ... 352
Podsumowanie ... 352

Rozdział 19. Lokalizacja i tłumaczenie ... 353
Włączanie modułu locale ... 353
Tłumaczenie interfejsu użytkownika ... 353
Łańcuchy znaków ... 353
Tłumaczenie łańcuchów znaków za pomocą funkcji t() ... 354
Zamiana wbudowanych łańcuchów znaków na własne ... 354
Tworzenie nowego tłumaczenia ... 362
Generowanie plików .pot za pomocą modułu translation template extractor ... 362
Tworzenie pliku .pot dla Twojego modułu ... 362
Tworzenie plików .pot dla całej witryny ... 364
Instalacja pełnego tłumaczenia dla języka ... 364
Ustawianie tłumaczenia w trakcie instalacji ... 364
Instalacja tłumaczenia w istniejącej witrynie ... 365
Obsługa języków z pisownią od prawej do lewej ... 366
Wybór języka ... 367
Domyślny ... 368
URL ... 369
Przeglądarka ... 370
Tylko URL ... 370

Tłumaczenie treści ... 370
Wprowadzenie do modułu tłumaczenia treści ... 371
Obsługa wersji językowych ... 371
Obsługa wersji językowych dla tłumaczenia ... 371
Pliki związane z lokalizacją i tłumaczeniem ... 375
Dodatkowe informacje ... 375
Podsumowanie ... 375

Rozdział 20. XML-RPC ... 377
XML-RPC — co i jak? ... 377
Wymagania wstępne dla XML-RPC ... 377
Klienci XML-RPC ... 378
Przykład klienta XML-RPC — pobieramy czas ... 378
Przykład klienta XML-RPC — pobieramy nazwę stanu ... 379
Obsługa błędów u klienta XML-RPC ... 380
Prosty serwer XML-RPC ... 382
Przypisywanie nazw metod za pomocą haka hook_xmlrpc() ... 383
Automatyczna walidacja typów parametrów za pomocą haka hook_xmlrpc() ... 383
Wbudowane metody XML-RPC ... 384
system.listMethods ... 384
system.methodSignature ... 386
system.methodHelp ... 386
system.getCapabilities ... 386
system.multiCall ... 387
Podsumowanie ... 387

Rozdział 21. Tworzenie bezpiecznego kodu ... 389
Obsługa danych przesyłanych przez użytkownika ... 389
Typy danych ... 390
check_plain() i t() — metody na oczyszczenie danych ... 391
Wykorzystywanie funkcji filter_xss() do zapobiegania atakom typu cross-site scripting ... 393
Wykorzystywanie haka filter_xss_admin() ... 395
Bezpieczna obsługa adresów URL ... 395
Tworzenie bezpiecznych zapytań za pomocą funkcji db_query() ... 396
Kontrola dostępu do danych prywatnych — hak hook_query_alter() ... 398
Zapytania dynamiczne ... 399
Uprawnienia i wywołania zwrotne strony ... 399
Ataki typu CSRF (Cross-Site Request Forgery) ... 400
Bezpieczeństwo plików ... 400
Uprawnienia do plików ... 400
Pliki chronione ... 401
Wysyłanie plików ... 401
Nazwy plików i ścieżki ... 402
Kodowanie nagłówków poczty elektronicznej ... 402
Pliki w środowisku produkcyjnym ... 403
Obsługa SSL ... 403
Samodzielne pliki PHP ... 403
Bezpieczeństwo i AJAX, czyli atak przez powtórzenie żądania ... 405

Bezpieczeństwo formularzy API ... 405
Ochrona konta superużytkownika ... 406
Podsumowanie ... 406

Rozdział 22. Dobre praktyki programistyczne ... 407
Standardy kodowania ... 407
Wcięcia i białe znaki ... 407
Operatory ... 407
Rzutowanie ... 408
Instrukcje kontroli przepływu ... 408
Wywołania funkcji ... 408
Deklaracje funkcji ... 409
Nazwy funkcji ... 409
Wywołania konstruktorów ... 410
Tablice ... 410
Cudzysłowy ... 411
Złączanie łańcuchów znaków ... 411
Komentarze ... 411
Przykłady dokumentacji ... 412
Dokumentowanie stałych ... 412
Dokumentowanie funkcji ... 412
Dokumentowanie implementacji haków ... 414
Dołączanie kodu ... 414
Znaczniki kodu PHP ... 414
Średniki ... 415
Przykładowe adresy URL ... 415
Konwencje nazewnicze ... 415
Sprawdzanie stylu kodowania za pomocą modułu coder ... 415
Szybka nawigacja w kodzie Drupala za pomocą narzędzia grep ... 416
Podsumowanie ... 417

Rozdział 23. Optymalizacja Drupala ... 419
Pamięć podręczna — klucz do wydajności Drupala ... 419
Optymalizacja PHP ... 421
Przekierowanie pliku pamięci podręcznej kodu operacyjnego PHP na /dev/zero ... 422
Ustawienia puli procesów PHP ... 422
Dostosowywanie serwera Apache ... 423
mod_expires ... 423
Przenoszenie dyrektyw z pliku .htaccess do pliku httpd.conf ... 423
Preforkowanie MPM vs. Apache MPM Worker ... 424
Dopasowywanie rozmiaru puli Apache ... 424
Zmniejszanie czasu oczekiwania Apache ... 425
Wyłączenie nieużywanych modułów Apache ... 425
Nginx zamiast Apache — zmiana serwera WWW ... 425
Pressflow ... 425
Varnish ... 425
Normalizacja żądań przychodzących w celu zwiększenia wydajności Varnisha ... 426
Varnish: znajdywanie dodatkowych cookies ... 427

Boost ... 427
Boost a Varnish ... 428
Konfiguracja systemu Linux w serwerach o wysokim natężeniu ... 428
Używanie szybkich systemów plików ... 429
Serwery dedykowane a serwery wirtualne ... 430
Unikanie wywoływania zewnętrznych usług sieciowych ... 430
Obniżanie czasu oczekiwania serwera ... 430
Optymalizacja bazy danych ... 431
Włączenie pamięci podręcznej zapytań MySQL ... 431
Wydajność silnika MySQL InnoDB w systemie Windows ... 431
Wydajność Drupala ... 432
Pozbycie się błędów 404 ... 432
Wyłączenie niewykorzystywanych modułów ... 432
Optymalizacje Drupala ... 432
Pamięć podręczna dla stron ... 432
Optymalizacja pasma ... 432
Czyszczenie tabeli sesji ... 433
Zarządzanie ruchem użytkowników zalogowanych ... 433
Wywoływanie programu cron ... 434
Architektury ... 434
Pojedynczy serwer ... 434
Odrębny serwer bazodanowy ... 434
Odrębny serwer bazodanowy i klaster serwerów WWW ... 434
Wiele serwerów baz danych ... 436
Znajdowanie wąskich gardeł ... 436
Serwer WWW z obciążonym procesorem ... 437
Serwer WWW nie dysponuje wystarczającą ilością pamięci RAM ... 437
Podsumowanie ... 440

Rozdział 24. Profile instalacji ... 441
Tworzenie nowego profilu instalacji ... 441
Plik enhanced.info ... 441
Plik enhanced.profile ... 442
Plik enhanced.install ... 443
Haki hook_install_tasks i hook_install_tasks_alter ... 455
Podsumowanie ... 456

Rozdział 25. Testowanie ... 457
Konfiguracja środowiska testowego ... 457
Tworzenie testów ... 459
Funkcje testowe ... 464
Asercje testowe ... 467
Podsumowanie ... 470

Dodatek A Spis tabel bazy danych Drupala ... 471

Dodatek B Zasoby ... 511
Programowanie ... 511
Repozytorium kodu Drupala (GIT) ... 511
Przykłady ... 511

Dokumentacja API Drupala ... 512
Porady dotyczące bezpieczeństwa ... 512
Aktualizacja modułów ... 512
Aktualizacja skórek ... 512
Podręczniki ... 512
Forum ... 512
Listy dyskusyjne ... 513
Development ... 513
Themes ... 513
Translations ... 513
Grupy użytkowników ... 513
Internet Relay Chat ... 513
Ameryka Północna ... 514
Europa ... 515
Azja ... 516
Ameryka Łacińska/Karaiby ... 516
Oceania ... 516
Afryka ... 516
Nagrania wideo ... 517
Dzienniki sieciowe ... 517
Konferencje ... 517
Obsługa Drupala ... 517

Skorowidz ... 519

ROZDZIAŁ 2

Tworzenie modułów

Moduły stanowią podstawowy materiał, budulec, na bazie którego funkcjonuje Drupal. Moduły pozwalają także na rozszerzanie możliwości standardowej wersji Drupala, nazywanej też rdzeniem Drupala. Osobom, które nie miały wcześniej kontaktu z Drupalem, tłumaczymy, że moduły Drupala można porównać do klocków Lego. Korzystając z precyzyjnie określonych reguł, można bez przeszkód łączyć różne moduły i uzyskiwać rozbudowane oraz funkcjonalne rozwiązania.

Istnieją dwie główne kategorie modułów Drupala — moduły rdzenia i dodatkowe. Moduły rdzenia są dostarczane wraz ze standardową instalacją systemu. W ich skład wchodzą ankiety, menu, kategorie, wyszukiwarka, subskrybent kanałów i fora. Moduły dodatkowe to wszystkie moduły tworzone przez społeczność Drupala. Rozszerzają one funkcjonalność oferowaną przez rdzeń Drupala. Na stronie *http://drupal.org/project/modules* znajdziesz tysiące modułów, które realizują przeróżne funkcje, począwszy od prostych modułów wyświetlających aktualną datę i czas, aż do zaawansowanych rozwiązań, takich jak sklep internetowy.

W tym rozdziale utworzymy nowy moduł od podstaw. W trakcie pracy poznasz standardy i reguły, jakie muszą spełniać tworzone przez Ciebie (i innych programistów) moduły. Warto utworzyć coś praktycznego, dlatego zajmiemy się potrzebnym zagadnieniem — przypisami. Podczas przeglądania stron typowej witryny Drupala może zdarzyć się, że będzie trzeba dodać krótki przypis — informację o danej stronie. Można by, co prawda, skorzystać z tradycyjnych komentarzy, jednak na ogół komentarze są dostępne dla każdego użytkownika odwiedzającego stronę, ewentualnie tylko dla zalogowanych użytkowników. Przypisy, według naszego założenia, mogłyby być widoczne tylko dla autora segmentu.

Tworzenie plików

Na początku musimy — rzecz jasna — wybrać nazwę dla naszego modułu. Skorzystamy z nazwy **`annotate`** („stworzyć przypis, adnotację") — krótkiej i treściwej. Następnie dla naszego modułu musimy znaleźć miejsce w strukturze katalogów. Moduły dodatkowe i własne są przechowywane w katalogu */sites/all/modules*. Każdy moduł jest przechowywany w podkatalogu mającym taką samą nazwę jak moduł.

Uwaga: Moduły rdzenia są przechowywane w katalogu */modules*. W ten sposób nie musisz obawiać się, że wszystkie dodawane przez Ciebie moduły zostaną nadpisane lub usunięte w trakcie wykonywania aktualizacji.

Jeśli chcesz, możesz utworzyć podkatalog */sites/all/modules/custom* do przechowywania wyłącznie modułów utworzonych przez Ciebie. Dzięki temu inni programiści bez problemu rozróżnią moduły utworzone przez Ciebie od modułów pobranych z witryny *Drupal.org*. Tak też uczynimy w tym przypadku — w podkatalogu */sites/all/modules/custom/annotate* znajdą się wszystkie pliki związane z modułem `annotate`.

Pierwszym plikiem, który utworzymy, będzie plik *annotate.info*. Każdy moduł Drupala 7. musi mieć plik składający się z nazwy modułu i rozszerzenia *.info*. W przypadku modułu `annotate` podstawowy zestaw informacji, który pozwoli Drupalowi prawidłowo rozpoznać nasz moduł, ma następującą postać:

```
name = Przypisy
description = "Moduł pozwala na tworzenie przypisów"
package = Drupal 7. Zaawansowane programowanie.
core = 7.x
files[] = annotate.module
files[] = annotate.install
files[] =  annotate.admin.inc
configure=admin/config/content/annotate/settings
```

Struktura tego pliku jest identyczna dla wszystkich modułów Drupala 7. Atrybut `name` określa nazwę modułu wyświetlaną na stronie konfiguracyjnej *Moduły*. Atrybut `description` zawiera opis modułu, wyświetlany w tym samym miejscu. Atrybut `package` definiuje pakiet (grupę), do której przynależy moduł. Moduły na stronie konfiguracyjnej *Moduły* są grupowane i wyświetlane według pakietów. Pole `core` określa wersję Drupala, do której został dostosowany moduł. Atrybut `php` definiuje wersję PHP wymaganą przez moduł. Wreszcie, atrybut `files` określa nazwy plików związanych z modułem. Dla modułu `annotate` są to pliki *annotate.module* i *annotate.install*.

Plik konfiguracyjny może zawierać także atrybuty opcjonalne. Oto przykład modułu, który wykorzystuje PHP w wersji 5.2, zaś do jego poprawnego działania Drupal musi mieć zainstalowane dwa moduły: forum i kategorie.

```
name = Chochlik na forum
description = Moduł losowo przypisuje odpowiedzi do różnych wątków
core = 7.x
dependencies[] = forum
dependencies[] = taxonomy
files[] = forumconfusion.module
files[] = forumconfusion.install
package = "Forum BonusPak Złego Boba"
php = 5.2
```

Teraz możemy zająć się tworzeniem kodu dla naszego modułu. Utwórz plik o nazwie *annotate.module* wewnątrz katalogu *sites/all/modules/custom/annotate*. Rozpocznij plik od otwierającego znacznika PHP i identyfikacyjnego znacznika CVS, po których następuje komentarz:

```
<?php

/**
 * @file
 * Moduł pozwala użytkownikom na dodawanie prywatnych przypisów do segmentów.
 *
 * Moduł dodaje pole tekstowe w momencie wyświetlenia segmentu.
 * Dzięki temu zalogowani użytkownicy mogą dodawać przypisy.
 */
```

Na początku zwróć uwagę na styl komentarza. Na początku znajduje się ciąg /**, po czym każdy kolejny wiersz rozpoczyna się od gwiazdki poprzedzonej spacją (*). Wreszcie, na końcu komentarza znajduje się sekwencja */. Atrybut `@file` informuje, że następny akapit zawiera opis pliku. Ten jednowierszowy opis jest wykorzystywany również przez `api.module` (*http://drupal.org/project/api*) — moduł odpowiedzialny za tworzenie i formatowanie dokumentacji na podstawie kodu. Przy okazji, zapamiętaj adres *http://api.drupal.org*. Na tej stronie znajdziesz dokumentację wszystkich API wchodzących w skład Drupala. Warto, abyś zapoznał się z tą stroną. Jest to nieocenione źródło informacji dla programistów, którzy tworzą bądź modyfikują moduły.

Tuż za pustym wierszem znajduje się dłuższy opis, przeznaczony dla programistów, którzy będą korzystać (i zapewne poprawiać) nasz kod. Zwróć uwagę, że nie używamy znacznika zamykającego (?>). Element ten jest w PHP opcjonalny. Jego dołączenie mogłoby spowodować problemy z białymi znakami na końcu pliku (*http://drupal.org/coding-standards#phptags*).

■ **Uwaga:** Zastanawiasz się, czemu przykładamy taką wagę do sposobu organizacji plików, katalogów i wszelkich innych elementów w obrębie instalacji Drupala? Odpowiedź jest prosta. Gdy setki programistów z całego świata pracują nad jednym projektem, zachowanie jednolitych standardów znacznie upraszcza i skraca czas pracy związany z łączeniem efektu prac różnych ludzi w jedną całość. Szczegółowe konwencje programistyczne obowiązujące w Drupalu znajdziesz w sekcji *Coding standards* w dokumencie *Developing for Drupal Handbook* (*http://drupal.org/coding-standards*).

Implementacja haka

Drupal intensywnie wykorzystuje mechanizm haków, zwanych też niekiedy wywołaniami zwrotnymi. W trakcie wykonywania swoich zadań Drupal daje modułom możliwość zareagowania na pewne zdarzenia. Przykładowo tuż po załadowaniu segmentu z bazy danych (a jeszcze przed wyświetleniem go na stronie) Drupal sprawdza, czy włączone moduły mają zaimplementowaną funkcję `hook_node_load()`. Następnie funkcje znalezione w modułach są wywoływane jeszcze przed wyświetleniem modułu na stronie. Przykład takiego zachowania sprawdzimy, korzystając z naszego modułu `annotate`.

Pierwszy hak, który będziemy implementować, jest zdefiniowany za pomocą haka `hook_menu()`. Za pomocą tej funkcji będziemy mogli dodać dwa elementy do menu administracyjnego naszej strony. W ten sposób do głównego menu konfiguracyjnego dodamy opcję *Przypisy*, a także podmenu (wewnątrz menu *Przypisy*) o nazwie *Ustawienia*. Kliknięcie tej pozycji spowoduje uruchomienie strony konfiguracyjnej naszego modułu. Elementy menu definiujemy przy użyciu tablic, składających się z kluczy i wartości określających zachowanie Drupala w momencie nadejścia żądania o określonej ścieżce. Tym zagadnieniem zajmiemy się szczegółowo w rozdziale 4., poświęconym menu i wywołaniom zwrotnym Drupala. Odwołanie do haka `hook_menu` nazwiemy `annotate_menu` — zastępując słowo `hook` nazwą tworzonego modułu. Ta zasada obowiązuje dla wszystkich haków — zawsze należy zastępować słowo hak nazwą konkretnego modułu.

Oto kod, który musimy dodać do naszego modułu:

```
/**
 * Implementacja haka hook_menu().
 */
function annotate_menu() {
  $items['admin/config/annotate'] = array(
    'title' => 'Przypis segmentu',
    'description' => 'Dostosuj opcje przypisów segmentu.',
    'position' => 'right',
    'weight' => -5,
    'page callback' => 'system_admin_menu_block_page',
    'access arguments' => array('administer site configuration'),
    'file' => 'system.admin.inc',
    'file path' => drupal_get_path('module', 'system'),
  );
  $items['admin/config/annotate/settings'] = array(
    'title' => 'Ustawienia przypisów',
    'description' => 'Dostosuj sposób działania przypisów.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('annotate_admin_settings'),
    'access arguments' => array('administer site configuration'),
    'type' => MENU_NORMAL_ITEM,
    'file' => 'annotate.admin.inc',
  );
  return $items;
}
```

Nie musisz analizować treści powyższego kodu w tej chwili. Jego działanie jest następujące: gdy użytkownik wejdzie pod adres *http://example.com/?q=admin/config/annotate/settings*, zostanie wywołana funkcja

`drupal_get_form()`, która otrzyma jako argument identyfikator formularza `annotate_admin_settings`. W związku z tym, nastąpi próba znalezienia funkcji opisującej ten formularz w pliku *annotate.admin.inc*. Element menu (a co za tym idzie, możliwość wykonania operacji) otrzymają jedynie użytkownicy z uprawnieniem *administer site configuration*. W momencie wyświetlenia formularza Drupal poprosi o określenie definicji formularza. Po zakończeniu odpytywania wszystkich modułów w celu uzyskania ich elementów menu następuje próba wybrania funkcji odpowiadającej ścieżce, z której nastąpiło żądanie.

■ **Uwaga:** Jeśli chcesz poznać bliżej mechanizm sterujący wszystkimi hakami, zajrzyj na stronę z dokumentacji funkcji `module_invoke_all()`, zadeklarowanej w pliku *includes/module.inc* (*http://api.drupal.org/api/function/module_invoke_all/7*).

Teraz powinno być już jasne, dlaczego funkcja haka nosi nazwę `hook_menu()`.

■ **Wskazówka:** Haki Drupala pozwalają na obsługę przeróżnych zdarzeń systemu. Pełna lista obsługiwanych haków znajduje się w dokumentacji API Drupala (*http://api.drupal.org/api/group/hooks/7*).

Dodawanie ustawień charakterystycznych dla modułu

Drupal zawiera wiele różnych rodzajów segmentów (nazywanych też rodzajami zawartości w interfejsie użytkownika), takich jak artykuły czy zwykłe strony. W związku z tym, chcielibyśmy, aby przypisy mogły być stosowane tylko do niektórych rodzajów segmentów. W tym celu musimy utworzyć stronę z możliwością wyboru rodzajów zawartości, przy których będzie pojawiać się opcja wstawienia przypisu. Strona będzie mieć prostą budowę — obok nazwy każdego dostępnego rodzaju zawartości znajdzie się pole typu checkbox. Zaznaczenie bądź usunięcie znaczenia tego pola będzie stanowiło o możliwości wstawiania przypisów przy danym rodzaju zawartości (rysunek 2.1). Opisana strona powinna być dostępna jedynie dla administratora, dlatego jej kod powinien być ładowany i przetwarzany tylko wtedy, kiedy będzie trzeba. Z tego względu kod strony umieścimy w osobnym pliku (a nie w dotychczas edytowanym pliku `annotate.module`, który będzie ładowany i wykorzystywany przy każdym żądaniu). W poprzednim listingu poinformowaliśmy Drupal, że powinien szukać formularza ustawień w pliku *annotate.admin.inc*, dlatego musimy dotrzymać słowa i utworzyć plik *sites/all/modules/annotate/annotate.admin.inc* o poniższej treści:

```
<?php

/**
 * @file
 * Strona administracyjna z wywołaniami zwrotnymi modułu annotate.
 */

/**
 * Tworzenie formularza dla ustawień przypisów.
 *
 * @ingroup forms
 * @see system_settings_form().
 */
function annotate_admin_settings() {
  // Pobierz tablicę rodzajów segmentów, zawierającą wewnętrzne nazwy jako klucze i przyjazne nazwy jako wartości,
  // np. array('page' => 'Prosta strona', 'article' => 'Artykuły')

  $types = node_type_get_types();
  foreach($types as $node_type) {
```

```
    $options[$node_type->type] = $node_type->name;
  }
$form['annotate_node_types'] = array(
    '#type' => 'checkboxes',
    '#title' => t('Użytkownicy mogą tworzyć przypisy dla następujących rodzajów zawartości'),
    '#options' => $options,
    '#default_value' => variable_get('annotate_node_types', array('page')),
    '#description' => t('Przy wybranych rodzajach zawartości zostaną wyświetlone pola tekstowe
    ↪w celu umożliwienia dodawania prywatnych uwag przez użytkowników.'),
  );

  $form['#submit'][] = 'annotate_admin_settings_submit';
  return system_settings_form($form);

}
```

Formularze w Drupalu są reprezentowane za pomocą struktury drzewiastej, tzn. tablicy tablic (tablic zagnieżdżonych). Przy użyciu tej struktury danych Drupal przekazuje niezbędne informacje do silnika renderującego (odpowiedzialnego za wygenerowanie kodu HTML). Dla zwiększenia czytelności każdy element tablicy został umieszczony w osobnym wierszu. Każdy atrybut formularza został oznaczony znakiem krzyżyka (#). Atrybuty te pełnią dodatkowo rolę kluczy w tablicy. Na początku określamy rodzaj elementów formularza za pomocą wartości checkboxes. Oznacza ona, że pola tego typu będą tworzone z wykorzystaniem tablicy z kluczami, do której odwołujemy się za pomocą zmiennej `$options`.

Opcje zostaną pobrane z zastosowaniem funkcji `node_type_get_types()`, która zwraca tablicę obiektów. Efekt działania tej funkcji mógłby wyglądać tak:

```
[article] => stdClass Object (
  [type] => article
  [name] => Artykuł
  [base] => node_content
  [description] => Artykuły są wykorzystywane do tworzenia treści o istotnej dacie publikacji,
  ↪np. aktualności, wiadomości prasowych czy też wpisów na blogu.
  [help] =>
  [has_title] => 1
  [title_label] => Tytuł
  [has_body] => 1
  [body_label] => Treść
  [custom] => 1
  [modified] => 1
  [locked] => 0
  [orig_type] => article
)
```

Kluczem powyższego obiektu (`[article] =>`) jest identyfikator rodzaju segmentu, używany wewnętrznie przez Drupala do przetwarzania segmentów, podczas gdy przyjazna nazwa to wartość atrybutu `name`, znajdująca się wewnątrz definicji obiektu.

API formularzy Drupala zobowiązuje nas do umieszcza w atrybucie `#options` par *`klucz => wartość`*. Dlatego właśnie w pętli `foreach` kluczem pary staje się atrybut `type`, a wartością pary — atrybut `name`. Korzystając z tablicy `$options`, Drupal wygeneruje w formularzu pola typu checkbox dla prostej strony, artykułu oraz wszystkich innych rodzajów zawartości, jakie są dostępne w Twojej aplikacji.

Tytuł formularza ustawiamy na podstawie wartości atrybutu `#title`.

■ **Uwaga:** Wszystkie teksty wyświetlane na stronie (takie jak wartości pól `#title` i `#description` naszego formularza) są umieszczane jako argumenty wywołania funkcji `t()`. Funkcja ta znacznie ułatwia tłumaczenie tekstów. Przetworzenie wszystkich tekstów w aplikacji przez funkcję tłumaczącą `t` znacznie ułatwi utworzenie kolejnych wersji językowych. Nie zastosowaliśmy tej funkcji dla elementów menu, ponieważ są one tłumaczone automatycznie.

Kolejny z atrybutów, `#default_value`, określa wartość domyślną elementu formularza. Skoro wartości typu checkboxes mogą zawierać wiele elementów typu checkbox, wartość domyślna dla takiego typu musi być tablicą. Wyrażenie zdefiniowane dla atrybutu `#default_value` jest warte dłuższej analizy:
`variable_get('annotate_node_types', array('page'))`

Drupal pozwala na przechowywanie i odczytywanie dowolnych wartości za pomocą dwóch specjalnych funkcji: `variable_get()` i `variable_set()`. Wartości są zapisywane w tabeli `variables` w bazie danych i dostępne przez cały czas przetwarzania żądania. Oczywiście, nie należy nadużywać tych funkcji zarówno pod względem liczby przechowywanych obiektów, jak i ich wielkości. Wartości są ładowane przy okazji każdego żądania, dlatego takie nadużycia mogłyby spowolnić pracę całego systemu. Mechanizm ten nadaje się natomiast znakomicie do przechowywania wszelkich ustawień. Zwróć uwagę, że do funkcji `variable_get` przekazujemy klucz, który jednoznacznie identyfikuje szukaną wartość (dzięki czemu Drupal może znaleźć ją w tabeli `variables`), a także wartość domyślną. Powyższy zapis możemy interpretować następująco: jeśli w tabeli `variables` nie ma określonych rodzajów segmentów, dla których są włączone przypisy, skorzystaj z tablicy określonej w wartości domyślnej. Domyślnie pozwalamy na dodawanie przypisów jedynie dla prostych (ang. *Basic*) stron.

■ **Wskazówka:** Korzystając z funkcji `system_settings_form()`, pamiętaj, że nazwa elementu formularza (np. `annotate_node_types`) musi być taka sama jak klucz określony w funkcji `variable_get()`.

Opis pełni jedynie funkcję informacyjną — dla administratora strony jest to informacja, do czego służy dane pole. Formularze omówimy szczegółowo w rozdziale 11.

Następnym krokiem w tworzeniu naszego modułu będzie dodawanie i usuwanie pola przypisu do rodzajów zawartości. Jeśli administrator strony włączy dany rodzaj zawartości, pole przypisu zostanie do niego dodane. Usunięcie zaznaczenia pola będzie równoznaczne z usunięciem pola przypisu z danego rodzaju zawartości. Field API Drupala (API związane z polami) pomoże zdefiniować pole i powiązać je z typem zawartości. Field API wykonuje wszystkie operacje związane z konfigurowaniem pola, włącznie z tworzeniem w bazie danych tabeli, w której będą przechowywane treści przypisów, i tworzeniem elementów formularza niezbędnych do pobrania treści przypisu od autora. Field API pozwoli także na powiązanie pola z rodzajem segmentu i dołączenie go do formularza edycji danego segmentu, a także uwzględnienie pola przy wyświetlaniu tegoż segmentu. Field API szczegółowo opisujemy w rozdziale 8.

Najpierw zajmiemy się obsługą wysyłanych formularzy, a zatem napiszemy kod obsługujący wysłanie formularza przez administratora. Chodzi — oczywiście — o formatkę wyboru rodzajów segmentów, które mają mieć włączone dodawanie przypisów. Jeśli pole checkbox danego segmentu nie jest zaznaczone, musimy upewnić się, że segment ten nie ma powiązanego pola przypisu — a jeśli tak jest, pole to należy usunąć wraz z istniejącymi przypisami. Włączenie pola typu checkbox spowoduje sprawdzenie, czy pole przypisu dla danego rodzaju zawartości istnieje. Jeśli nie, zostanie dodane (w przeciwnym przypadku nie trzeba podejmować żadnej akcji).

```
/**
* Przetwarzanie formularza ustawień przypisów.
*/
function annotate_admin_settings_submit($form, $form_state) {
  // Przeiteruj kolekcję checkboksów zawartych w formularzu
  foreach ($form_state['values']['annotate_node_types'] as $key => $value) {
    // Jeśli checkbox dla danego rodzaju zawartości nie jest zaznaczony, sprawdź, czy dany rodzaj zawartości ma zdefiniowane
    // pole przypisu, korzystając z funkcji field_info_instance. Jeśli tak, usuń pole przypisu, ponieważ administrator usunął
    // zaznaczenie danego pola formularza.
    if (!$value) {
      $instance = field_info_instance('node', 'annotation', $key);
      if (!empty($instance)) {
        field_delete_instance($instance);
        watchdog("Przypis", 'Usunięto pole przypisu z następującego rodzaju zawartości:
        %key', array('%key' => $key));
      }
    } else {
```

```
      // Jeśli checkbox dla danego rodzaju zawartości jest zaznaczony, sprawdź, czy pole przypisu dla danego rodzaju
      // zawartości istnieje. Jeśli nie, dodaj pole przypisu do tego rodzaju zawartości.
      $instance = field_info_instance('node', 'annotation', $key);
      if (empty($instance)) {
        $instance = array(
          'field_name' => 'annotation',
          'entity_type' => 'node',
          'bundle' => $key,
          'label' => t('Przypis'),
          'widget_type' => 'text_textarea_with_summary',
          'settings' => array('display_summary' => TRUE),
          'display' => array(
            'default' => array(
              'type' => 'text_default',
            ),
            'teaser' => array(
              'type' => 'text_summary_or_trimmed',
            ),
          ),
        );
        $instance = field_create_instance($instance);
        watchdog('Przypis', 'Dodano pole przypisu do następującego rodzaju zawartości: %key',
          array('%key' => $key));
        }
      }
    } // Koniec pętli foreach.
}
```

Po zbudowaniu skryptu obsługi formularza możemy zająć się utworzeniem pliku *.install* dla naszego modułu. Plik instalacyjny zawiera jedną lub więcej funkcji, które są wywoływane w momencie instalacji lub deinstalacji modułu. W naszym przypadku w trakcie instalacji chcemy utworzyć pole przypisu, które mogłoby być przypisane do rodzajów zawartości przez administratora. Deinstalacja modułu wiązać się będzie z usunięciem powiązań pola przypisu ze wszystkimi rodzajami zawartości, a także usunięciem samego pola z bazy danych Drupala. W tym celu musimy utworzyć nowy plik o nazwie *annotate.install* w katalogu modułu `annotate`.

Pierwszą funkcją, jaką wywołamy, będzie funkcja obsługi haka `hook_install()`. Zgodnie z konwencją nazewniczą Drupala, funkcja otrzyma nazwę `annotate_install()`. W funkcji tej za pomocą Field API sprawdzimy, czy pole istnieje, a jeśli nie, nie pozostanie nam nic innego, jak je utworzyć:

```
<?php

/**
 * Implementacja haka hook_install()
 */

function annotate_install() {

  // Sprawdź, czy pole istnieje.
  $field = field_info_field('annotation');

  // Jeśli nie, utwórz pole.
  if (empty($field)) {
    $field = array(
      'field_name' => 'annotation',
      'type' => 'text_with_summary',
      'entity_types' => array('node'),
      'translatable' => TRUE,
    );
```

```
    $field = field_create_field($field);
  }

}
```

Teraz możemy przejść do tworzenia funkcji deinstalacyjnej, korzystając z haka `hook_uninstall`. Stosowna funkcja otrzyma nazwę `annotate_uninstall`. Skorzystamy w niej z funkcji `watchdog` w celu zapisania w dzienniku (logu) informacji o deinstalacji modułu. Następnie skorzystamy z funkcji `node_get_types()`, aby pobrać wszystkie dostępne rodzaje zawartości. Dzięki temu sprawdzimy, które z rodzajów zawartości mają swoje pola przypisu i usuniemy je. Na koniec będziemy mogli usunąć definicję pola przypisu.

```
/**
 * Kod obsługi haka hook_uninstall()
 */
function annotate_uninstall() {

   watchdog("Moduł Przypisy", "Deinstalacja modułu i usuwanie pól");

   $types = node_type_get_types();
   foreach($types as $type) {
      annotate_delete_annotation($type);
   }
   $field = field_info_field('annotation');

   if ($field) {
      field_delete_field('annotation');
   }

}

function annotate_delete_annotation($type) {

  $instance = field_info_instance('node', 'annotation', $type->type);

  if ($instance) {
     field_delete_instance($instance);
  }

}
```

Zmierzając powoli do końca tworzenia naszego modułu, możemy zająć się ostatnią czynnością — dodaniem kodu do pliku *.module* w celu sprawdzenia, czy osoba przeglądająca dany segment jest jego autorem. Jeśli ten warunek nie jest spełniony, przypis musi zostać ukryty. Do realizacji tej funkcjonalności posłuży hak `hook_node_load()`, wywoływany w trakcie ładowania segmentu. Wewnątrz funkcji obsługi tego haka sprawdzimy, czy osoba przeglądająca segment jest jego autorem. Jeśli tak nie jest, przypis zostanie ukryty przez zastosowanie operacji `unset`:

```
/**
 * Kod obsługi haka hook_node_load()
 */
function annotate_node_load($nodes, $types) {

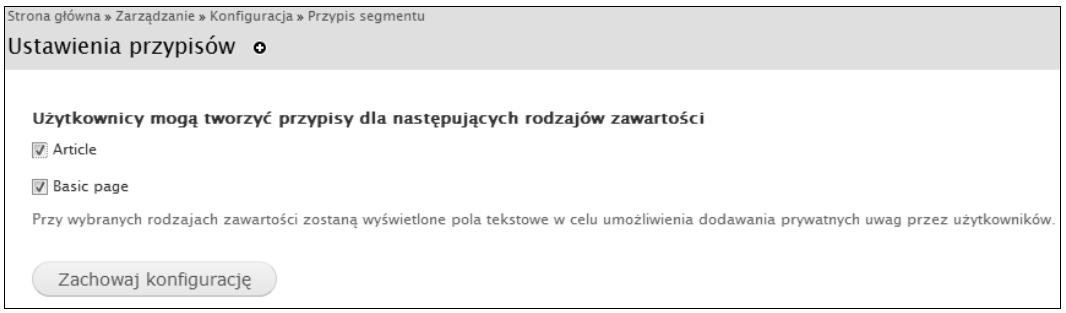
  global $user;

  // Sprawdź, czy osoba przeglądająca segment jest jego autorem. Jeśli nie, ukryj przypis.
  foreach ($nodes as $node) {
     if ($user->uid != $node->uid) {
```

```
        unset($node->annotation);
      }
    }
}
```

Zapisz utworzone pliki (*.info*, *.install*, *.admin.inc*, *.module*), a następnie kliknij łącze *Moduły* w menu administratora na górze strony. Twój moduł powinien znaleźć się w grupie zatytułowanej *Drupal 7. Zaawansowane programowanie.* (Jeśli tak nie jest, sprawdź dokładnie pliki *annotate.info* i *annotate.module* pod kątem składni i upewnij się, że oba pliki znajdują się w katalogu *sites/all/modules/custom*). Po całej pracy wykonanej w tym rozdziale możesz wreszcie włączyć moduł.

Po włączeniu modułu na stronie *admin/config/annotate/settings* powinien znaleźć się formularz konfiguracji dla modułu `annotate.module` (rysunek 2.1).



Rysunek 2.1. *Formularz konfiguracji dla modułu annotate.module*

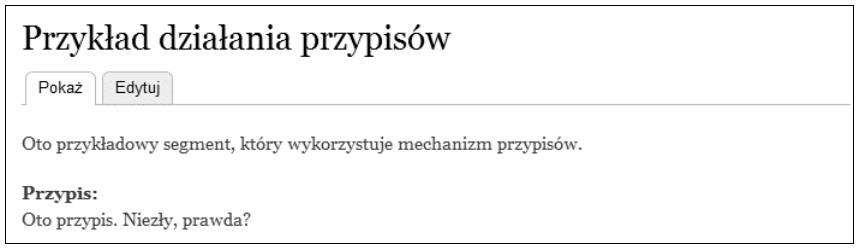
Zaledwie kilka wierszy kodu dało w pełni funkcjonalny formularz konfiguracji naszego modułu, który automatycznie zapisze i zapamięta ustawienia. Jest to dobitny przykład możliwości Drupala.

Sprawdźmy teraz główną funkcjonalność naszego modułu. Najpierw musimy włączyć przypisy dla wszystkich rodzajów zawartości. Zaznacz wszystkie pola na stronie konfiguracyjnej i kliknij przycisk *Zachowaj konfigurację*. Następnie utwórz prostą stronę i przewiń ją na sam dół, aż zobaczysz pole *Przypis* (rysunek 2.2).



Rysunek 2.2. *Gotowy formularz przypisu*

Utwórz nowy segment, wprowadzając tytuł, treść i przypis. Po zakończeniu zachowaj zmiany, a powinieneś zobaczyć efekt, taki jak na rysunku 2.3.



Rysunek 2.3. *Segment zawierający przypis*

Możesz zastanawiać się, gdzie Drupal przechowuje wszystkie informacje związane z przypisami — w końcu nie wykonywaliśmy jawnie żadnych operacji związanych z bazą danych. Field API wykonuje wszystkie niezbędne operacje, począwszy od utworzenia tabeli, przez zapis i odczyt danych, aż do usunięcia tabeli w przypadku deinstalacji. Wywołanie funkcji `field_create_field()` powoduje utworzenie tabeli o nazwie `field_data_<nazwa_pola>`. W naszym przypadku nazwą tabeli będzie ciąg `field_data_annotations`. Szczegółowe informacje na temat Field API znajdziesz w rozdziale 4.

Tworzenie własnej sekcji administracyjnej

Drupal zawiera kilka kategorii ustawień administracyjnych — takich jak zarządzanie treścią czy zarządzanie użytkownikami — które pojawiają się na stronie *Konfiguracja*. Może się zdarzyć, że Twojemu modułowi przydałaby się własna kategoria. Nic prostszego — w tym podrozdziale wykreujemy nową kategorię ustawień o nazwie *Przypisy segmentów*. Na początek musimy utworzyć funkcję obsługi haka `hook_menu`.

```
/**
 * Obsługa haka hook_menu().
 */
function annotate_menu() {
  $items['admin/config/annotate'] = array(
    'title' => 'Przypis segmentów',
    'description' => 'Dostosuj opcje przypisów segmentów.',
    'position' => 'right',
    'weight' => -5,
    'page callback' => 'system_admin_menu_block_page',
    'access arguments' => array('administer site configuration'),
    'file' => 'system.admin.inc',
    'file path' => drupal_get_path('module', 'system'),
  );
  $items['admin/config/annotate/settings'] = array(
    'title' => 'Ustawienia przypisów',
    'description' => 'Zmień zachowanie przypisów.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('annotate_admin_settings'),
    'access arguments' => array('administer site configuration'),
    'type' => MENU_NORMAL_ITEM,
    'file' => 'annotate.admin.inc',
  );

  return $items;
}
```

Na rysunku 2.4 przedstawiamy kategorię na stronie *Konfiguracja*, zawierającą łącze do ustawień naszego modułu.



Rysunek 2.4. *Łącze do ustawień modułu przypisów pojawia się jako odrębna kategoria*

Jeśli kiedykolwiek zdarzy Ci się zmienić kod obsługi haka menu, musisz wyczyścić pamięć podręczną (ang. *cache*) menu. Można to zrobić na dwa sposoby — czyszcząc tabelę *cache_menu* lub wybierając opcję *Utwórz ponownie menu* dostępną w module programistycznym Drupala (ang. *devel.module*). Możesz także skorzystać z opcji *Wyczyść dane pamięci podręcznej*, która znajduje się na stronie *Konfiguracja/Wydajność*.

■ **Wskazówka:** Moduł devel (*http://drupal.org/project/devel*) powstał specjalnie dla programistów rozszerzających możliwości Drupala. Przy jego użyciu można łatwo uzyskać dostęp do takich opcji jak czyszczenie pamięci podręcznej, podgląd zmiennych czy śledzenie zapytań do bazy danych. To obowiązkowe narzędzie każdego programisty Drupala.

Dodanie nowej kategorii wymaga wykonania dwóch kroków. Po pierwsze, dodać trzeba element menu, który opisuje nagłówek kategorii. Element menu ma swoją unikalną ścieżkę (*admin/config/annotate*). Zostanie on umiejscowiony w prawej kolumnie, z wagą o wartości 5, dzięki czemu znajdzie się tuż przed kategorią *Usługi web*, co stanowi najwygodniejsze rozwiązanie.

Drugi krok sprowadza się do zagnieżdżenia hiperłącza prowadzącego do ustawień przypisów wewnątrz kategorii Przypisy segmentów. Uda się to osiągnąć po ustawieniu ścieżki oryginalnego menu na wartość *admin/config/annotate/settings*. W momencie ponownego tworzenia menu Drupal analizuje ścieżki w celu określenia związku między nadrzędnymi i podrzędnymi elementami. Efektem będzie wyświetlenie elementu o ścieżce *admin/config/annotate/settings* jako dziecka elementu o ścieżce *admin/config/annotate*.

Drupal ładuje jedynie pliki niezbędne do zrealizowania żądania. Takie podejście sprawia, że udaje się oszczędzić sporo pamięci. Zwróć uwagę, że atrybut `page callback` (wywołanie zwrotne) elementu tablicy `items` odwołuje się do funkcji spoza naszego modułu (`system_admin_menu_block_page()` z modułu `system.module`). W związku z tym, musimy poinformować Drupal o konieczności załadowania pliku *modules/system/system.admin.inc* (domyślnie Drupal próbowałby dołączyć plik *sites/all/modules/custom/annotate/system.admin.inc*). Możemy to osiągnąć, pobierając ścieżkę do modułu *system* i umieszczając ją jako wartość dla atrybutu `file` naszego elementu w tablicy `items`.

Oczywiście, powyższy przykład tworzenia kategorii został zrealizowany głównie w celach dydaktycznych — w praktyce tworzenie kategorii powinno być dobrze uzasadnione — nie ma sensu irytować administratora nadmierną liczbą kategorii.

Wyświetlanie formularza ustawień

W module przypisów administrator może wybrać rodzaje zawartości (segmenty), które mogą być oznaczane przypisami (rysunek 2.1). Przyjrzyjmy się bliżej temu zagadnieniu.

Zmiana ustawień modułu przypisów wiąże się z koniecznością wyświetlenia formularza. Nasz element menu wywoła zwrotnie funkcję `drupal_get_form()`, przekazując jako parametr tablicę `annotate_admin_settings`. Oznacza to, że w momencie nadejścia żądania *http://example.com/?q=admin/config/annotate/settings* zostanie wywołana funkcja `drupal_get_form('annotate_admin_settings')`, co w praktyce spowoduje utworzenie formularza określonego za pomocą funkcji `annotate_admin_settings()`.

Przeanalizujmy funkcję definiującą formularz, która dodaje pole typu checkbox do rodzajów zawartości (rysunek 2.1), a także dwie nowe opcje. Funkcja znajduje się w pliku *sites/all/modules/custom/annotate/annotate.admin.inc*.

```
/**
 * Tworzenie formularza dla ustawień przypisów.
 *
```

```
 * @ingroup forms
 * @see system_settings_form().
 */
function annotate_admin_settings() {
  // Pobierz tablicę rodzajów segmentów, zawierającą wewnętrzne nazwy jako klucze i przyjazne nazwy jako wartości,
  // np. array('page' => 'Prosta strona', 'article' => 'Artykuły')

  $types = node_type_get_types();
  foreach($types as $node_type) {
     $options[$node_type->type] = $node_type->name;
  }
$form['annotate_node_types'] = array(
    '#type' => 'checkboxes',
    '#title' => t('Użytkownicy mogą tworzyć przypisy dla następujących rodzajów zawartości'),
    '#options' => $options,
    '#default_value' => variable_get('annotate_node_types', array('page')),
    '#description' => t('Przy wybranych rodzajach zawartości zostaną wyświetlone pola tekstowe
    ↪w celu umożliwienia dodawania prywatnych uwag przez użytkowników.'),
  );

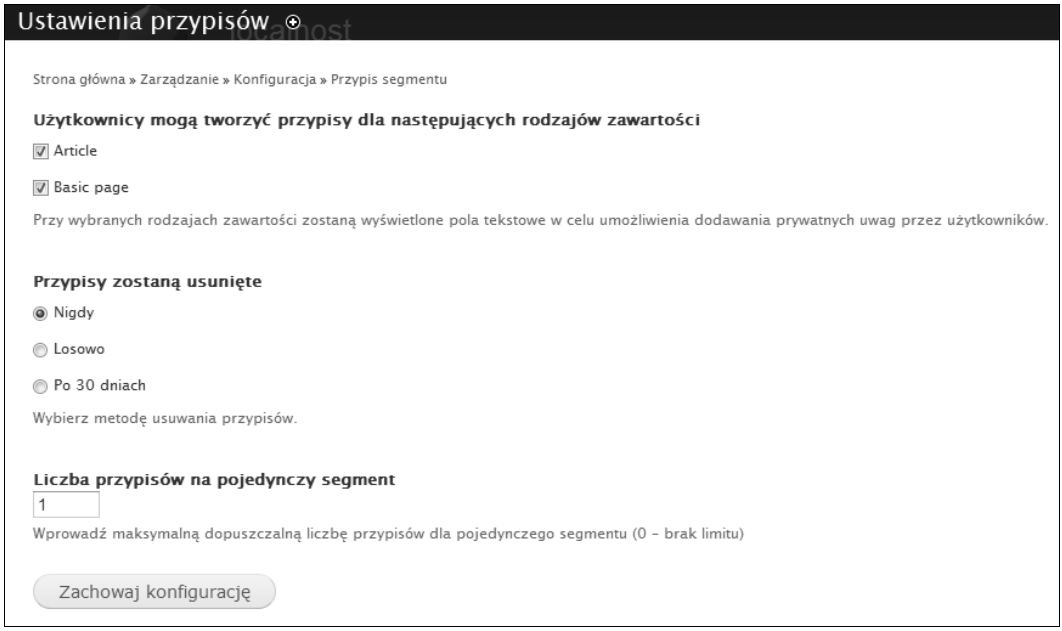
  $form['annotate_deletion'] = array(
    '#type' => 'radios',
    '#title' => t('Przypisy zostaną usunięte'),
    '#description' => t('Wybierz metodę usuwania przypisów.'),
    '#options' => array(
      t('Nigdy'),
      t('Losowo'),
      t('Po 30 dniach')
    ),
    '#default_value' => variable_get('annotate_deletion', 0) // Domyślnie: Nigdy
  );
  $form['annotate_limit_per_node'] = array(
    '#type' => 'textfield',
    '#title' => t('Liczba przypisów na pojedynczy segment'),
    '#description' => t('Wprowadź maksymalną dopuszczalną liczbę przypisów dla pojedynczego segmentu
    ↪(0 – brak limitu)'),
    '#default_value' => variable_get('annotate_limit_per_node', 1),
    '#size' => 3
  );

  $form['#submit'][] = 'annotate_admin_settings_submit';
  return system_settings_form($form);
}
```

Pole `radio` (pole jednokrotnego wyboru) posłuży nam do wyboru metody usuwania przypisów, a z kolei zwykłe pole tekstowe pozwoli na ograniczenie maksymalnej liczby przypisów dla pojedynczego segmentu (implementację tych funkcji pozostawiamy jako ćwiczenie). Zamiast samodzielnie przetwarzać nasz formularz, korzystamy z funkcji `system_settings_form()`, która doda niezbędne przyciski, przeprowadzi walidację i obsłuży wysłany formularz. Na rysunku 2.5 przedstawiamy aktualny widok formularza.

Walidacja ustawień wysłanych przez użytkownika

Skoro funkcja `system_settings_form()` zajmuje się przechowaniem wartości formularza automatycznie, skąd mamy wiedzieć, że wartość wprowadzona w pole *Liczba przypisów* na pojedynczy segment jest istotnie liczbą? Musimy dodać stosowny kod, tworząc funkcję walidacji (`annotate_admin_settings_validate($form, $form_state)`) w pliku *sites/all/modules/custom/annotate/annotate.admin.inc*. W razie jakichkolwiek problemów z przesłaną wartością funkcja włączy komunikat o błędzie:



Rysunek 2.5. *Rozszerzony formularz ustawień, zawierający pole typu checkbox, pole typu radio i pole tekstowe*

```
/**
 * Walidacja formularza ustawień przypisów.
 */
function annotate_admin_settings_validate($form, &$form_state) {
  $limit = $form_state['values']['annotate_limit_per_node'];
  if (!is_numeric($limit)) {
    form_set_error('annotate_limit_per_node', t('Proszę wprowadzić liczbę.'));
  }
}
```

Powyższa funkcja zostanie wywołana w momencie przetwarzania formularza przez Drupal. Po wystąpieniu problemów użytkownik otrzyma stosowny komunikat o błędzie. Dodatkowo pole zawierające niepoprawną zawartość zostanie odpowiednio podświetlone.

Skąd Drupal „wie", że ma wywołać właśnie tę funkcję? Skorzystaliśmy ze specjalnej nazwy, składającej się z nazwy funkcji definiującej formularz (`annotate_admin_settings`) z dołączonym członem `_validate`. Więcej informacji na temat funkcji walidacji znajdziesz w rozdziale 11.

Przechowywanie ustawień

W poprzednim przykładzie dwie operacje, czyli wprowadzenie nowych ustawień i kliknięcie przycisku *Zachowaj konfigurację*, zadziałały bez problemu. Teraz zastanowimy się nad tym, co właściwie stało się po wykonaniu powyższych operacji.

Tabela variables w akcji

Najpierw musimy przyjrzeć się polu *Liczba przypisów na pojedynczy segment*. Wartość domyślna tego pola (`#default_value`) jest uzyskiwana za pomocą wywołania `variable_get('annotate_limit_per_node', 1)`.

Drupal korzysta z tabeli `variables` znajdującej się w bazie danych. Pary klucz-wartość mogą być do niej zapisywane za pomocą metody `variable_set($klucz, $wartosc)`, a następnie odczytywane przy użyciu wywołania `variable_get($klucz, $wartosc_domyslna)`. Powyższe wywołanie metody `variable_get()` dla przypisów możemy więc zinterpretować następująco: ustaw wartość domyślną pola *Liczba przypisów na pojedynczy segment* na wartość przechowywaną w tabeli `variables` dla klucza `annotate_limit_per_node`, a jeśli takiej wartości nie ma, skorzystaj z wartości 1.

■ **Ostrzeżenie:** Przechowywanie wartości w tabeli `variables` wiąże się z możliwością powstania konfliktów nazw. Z tego względu zawsze nadawaj elementom formularza i kluczom tej tabeli takie same nazwy (np. `annotate_limit_per_node`). Nazwy najlepiej tworzyć według pewnego schematu, np. nazwa modułu + czytelny opis zmiennej. Tak utworzone nazwy stosuj zarówno dla elementów formularzy, jak i kluczy wartości z tabeli `variables`.

Pole *Przypisy zostaną usunięte* jest nieco bardziej skomplikowane, ponieważ jest to pole typu radio. Obiekt `#options` dla tego pola ma następującą postać:

```
'#options' => array(
  t('Nigdy'),
  t('Losowo'),
  t('Po 30 dniach')
)
```

Gdy PHP dostaje tablicę bez kluczy, kolejne wartości otrzymują automatycznie klucze liczbowe, dlatego w rzeczywistości powyższa tablica wygląda następująco:

```
'#options' => array(
  [0] => t('Nigdy'),
  [1] => t('Losowo'),
  [2] => t('Po 30 dniach')
)
```

Ustawiając wartość domyślną, korzystamy z poniższej konstrukcji. Domyślna wartość jest określona kluczem 0, co w praktyce przekłada się na użycie wartości `t('Nigdy')`.

```
'#default_value' => variable_get('annotate_deletion', 0) // Domyślnie: Nigdy
```

Wczytywanie wartości za pomocą funkcji variable_get()

Do odczytania zapisanych wcześniej ustawień modułu należy posłużyć się funkcją `variable_get()`:

```
// Pobierz maksymalną dozwoloną liczbę przypisów na pojedynczy segment.
$max = variable_get('annotate_limit_per_node', 1);
```

Warto zwrócić uwagę na podanie wartości domyślnej w wywołaniu funkcji `variable_get()`. Zostanie ona wykorzystana np. w sytuacji, gdy administrator ani razu nie odwiedził strony z ustawieniami.

Co dalej?

Oczywistym krokiem będzie udostępnienie naszego modułu społeczności Drupala, dlatego musimy utworzyć plik *README.txt* — w tym samym katalogu, co pliki *annotate.info*, *annotate.module*, *annotate.admin.inc* i *annotate.install*. Plik *README.txt* zawiera informacje o autorze i instalacji modułu. Informacje na temat licencji nie są konieczne, ponieważ wszystkie moduły wysyłane na stronę *drupal.org* podlegają licencji GPL. Ponadto odpowiedni skrypt na stronie *drupal.org* automatycznie dołączy plik *LICENSE.txt*. Po zakończeniu przygotowania modułu możesz wysłać go do repozytorium kontrybucji na stronie *drupal.org*, a następnie utworzyć stronę projektu, aby mieć kontakt z innymi członkami społeczności.

Podsumowanie

Po przeczytaniu tego rozdziału powinieneś:

- poradzić sobie z utworzeniem modułu Drupala od podstaw,
- rozumieć, jak korzystać z mechanizmu haków,
- wiedzieć, jak należy zapisywać i odczytywać ustawienia charakterystyczne dla danego modułu,
- tworzyć i przetwarzać proste formularze, korzystając z API formularzy Drupala,
- umieć tworzyć nowe kategorie w obrębie głównej strony administracyjnej Drupala,
- tworzyć formularze dla administratorów stron zawierające pola typu checkbox, pola typu radio i pola tekstowe,
- umieć walidować ustawienia i wyświetlać komunikaty o błędach po wystąpieniu jakichkolwiek problemów,
- rozumieć, jak Drupal przechowuje ustawienia z wykorzystaniem systemu trwałych zmiennych.

Skorowidz

A

administer site configuration, 36
administracyjna sekcja, 42
AJAX, 343
  dodatkowe możliwości technologii, 252
  korzyści stosowania technologii, 248
analizowanie struktury witryny, 171
Apache, 26
  dyrektywa MaxClients, 437
  dyrektywa MaxRequestPerChild, 438
  max_execution_time, 431
  mod_expires, 423
  mod_fcgid, 430
  mod_rewrite, 26
  pracownik Apache MPM, 424
  preforkowanie Apache MPM, 424
  przenoszenie dyrektyw z pliku .htaccess, 423
  Timeout, 431
  wyłączenie nieużywanych modułów, 425
  zmiana rozmiaru puli procesów potomnych, 424
  zmniejszanie czasu oczekiwania, 425
API bloków, 198
API formularzy (Form API), 209
API plików, 283
  domyślne URI plików, 284
  typowe zadania i funkcje, 284
  zarządzane i niezarządzane, 279
API schematów, 101
API wyszukiwarki (Search API), 265
Architektury, 434
argumenty wywołań zwrotnych strony, 74
arkusz stylów, 170
asercje testowe, 467
atak przez powtórzenie żądania, 405
ataki typu CSRF
  korzystanie z API formularzy, 400
  unieszkodliwianie złych żądań GET, 400
atrybut
  #default_value, 38
  #options, 37
  apc.mmap_file_mask, 422
  configurable, 55
  default_socket_timeout, 430
  description, 34
  fields, 96
  files, 34
  name, 34
  package, 34
  page callback, 43
  php, 34
  post_max_size, 281
  session.cache_expire, 433
  session.gc_maxlifetime, 433
atrybut formularza, 37
atrybuty klucz-wartość funkcji hook_menu(), 72
automatyczne kanały RSS, 300

B

baza danych, 93
  API, 94
  określanie parametrów, 93
  partycjonowanie, 436
  raportowanie zapytań, 438
  replikacja, 436
  rozmiar pamięci podręcznej zapytań, 431
  silnik, 94
  tworzenie własnego sterownika, 112
  ustawienia niezbędne do nawiązania połączenia, 93
  warstwa abstrakcji, 94
  włączenie pamięci podręcznej, 431
  wybór pliku, 95
  zmiana silnika tabel na MyISAM, 431
bezpieczeństwo
  funkcja check_plain(), 391
  funkcja t(), 392
  kodowanie znaków w adresach URL, 389
  konwertowanie formatu do języka HTML, 391
  oczyszczanie tekstu, 389
  operacje wykonywane przez funkcję filter_xss(), 394
  szkodliwe znaczniki, 391
  zmienne typowane, 390
  zwykły tekst, 390
bezpieczeństwo formularzy API, 405
bezpieczeństwo plików
  ochrona ścieżki systemu plików, 401
  pliki chronione, 401
  pliki odrzucane, 401
  uprawnienia, 400
  usuwanie niebezpiecznych plików, 291
bezpieczna obsługa adresów URL
  filter_xss_bad_protocol(), 395
  funkcja valid_url(), 395
bezpieczne zapytania
  funkcja db_query(), 396
blok, 133
blok Nawigacja, 76
blokada, 439
  na poziomie tabel, 439
  na poziomie wierszy, 439

bloki, 29, 197, 339
  ustawienia widoczności, 199
  przykłady własne, 198
  rozmieszczenie, 199
  tworzenie bloku, 200
  tworzenie z poziomu kodu, 198
  tworzenie za pomocą interfejsu administracyjnego, 198
  wyświetlanie bloku, 208
bloki dostępne domyślnie, 197
błędy, 91
błędy HTTP, 381
błędy sieci, 380
błędy składni wywołania, 381
Boost, 427
brak obsługi JavaScriptu, 252

C

cookies, 115
cron, 434

D

deklarowanie typów kolumn, 106
dobre praktyki programistyczne
  cudzysłowy, 411
  deklaracje funkcji, 409
  dokumentowanie funkcji, 412
  dokumentowanie implementacji haków, 414
  dokumentowanie stałych, 412
  dołączanie kodu, 414
  instrukcje kontroli przepływu, 408
  komentarze, 411
  konwencje nazewnicze, 415
  nazwy funkcji, 409
  operatory, 407
  rzutowanie, 408
  średniki, 415
  tablice, 410
  wcięcia i białe znaki, 407
  wywołania funkcji, 408
  wywołania konstruktorów, 410
  złączanie łańcuchów znaków, 411
  znaczniki kodu PHP, 414
dodawanie nowej kategorii, 43
  dodanie elementu menu, 43
  zagnieżdżenie hiperłącza, 43
dodawanie
  danych do obiektu $user w momencie ładowania, 124
  formularza wyszukiwania, 267
  modyfikowanie zmiennych szablonu, 195
  usuwanie pola przypisu, 38
  łącza do bloku Nawigacja, 76
  metadanych do segmentów, 272
  pola Data wydarzenia, 154
  pola Miejsce wydarzenia, 153
  pól do rodzaju zawartości, 153
  pól z poziomu kodu, 164
  ustawień charakterystycznych dla modułu, 36
  własnego języka, 358
  wyzwalaczy, 65
DOM (Document Object Model), 331
domyślne funkcje skórek dla elementów rdzenia, 214
domyślne opcje wyszukiwania, 359
Doxygen, 411
Drupal, 25
  niezbędne pliki i katalogi, 403
  stałe, 285
dynamiczne części strony, 28
dyrektywa PHP, 326
działanie, 51
  Block current user, 59
  obsługujące wszystkie wyzwalacze, 54
  proste i zaawansowane, 55
  typu odpowiedzi, 60
  typu użytkownika, 60
dziedzina, 147

E

efekt działania zastępników @, % i !, 393
ekstrakcja szablonów, 363
ekstraktor na stronie internetowej, 363
ekstraktor uruchamiany z wiersza poleceń, 363
element
  #children, 214
  #type, 214
elementy formularza
  Markup, 248
  pole
    radio (radio button), 243
    zaznaczalne (checkboxes), 244
    daty (date), 245
    do wysyłania plików (file upload), 246
    hasła (password), 241
    hasła z potwierdzeniem, 241
    tekstowe (text field), 240
    typu wartość (Value), 245
    ukryte (hidden), 245
    wagi (weight), 246
    wyboru (select), 242
  pozycja (Item), 248
  przycisk, 247
    obrazkowy (image button), 247
    wysyłania (submit), 247
  wielowierszowe pole tekstowe, 242
  zbiór pól (fieldset), 247
elementy skórkowalne (themeable), 192
ETS (Easy Template System), 28

F

Fastcgi, 425
fazy rozruchu, 32, 315
  baza danych, 32
  język, 32
  konfiguracja, 32
  nagłówek strony, 32
  pełna, 32
  sesja, 32
filtr, 255
  funkcja pomocnicza, 261
  funkcja przetwarzania, 260
  implementacja haka hook_filter_info(), 260
  instalacja, 258
  Ograniczenie dozwolonych znaczników HTML, 256
  PHP, 271
  tworzenie własnego filtru, 259
  Zamiana nowych linii na HTML, 256
  Zamień adresy URL na odnośniki, 256
  zastosowania, 259
flagi typów elementów menu, 88
format tekstu, 256
  kod PHP, 256
  przefiltrowany HTML, 256
  zwykły tekst, 256
  domyślny, 258
formatowanie wyników wyszukiwania, 267
formularz
  konfiguracji, 61
  modyfikowanie
    dowolnego formularza, 229
    funkcji walidacji formularza logowania, 126
    po utworzeniu, 213
    przed renderowaniem, 214
    przed utworzeniem, 213
    wybranego formularza, 230
    za pomocą haka hook_form_alter(), 229
  rozpoczęcie procesu obsługi, 211
  wyszukiwania zaawansowanego, 266
  zawierający zbiory pól, 220

dynamiczny, 231
wykorzystujący technologię AJAX, 248
funkcja
_drupal_session_read(), 327
_element_info(), 211
action_info_after(), 59
actions_do(), 62
parametry, 62
addClass(), 337
annotate_install(), 39
beep_action_info(), 53
beep_multiple_beep_action(), 61
cache_clear_all(), 320
cache_get(), 319
cache_get_multiple(), 319
cache_set(), 318
check_plain(), 391
check_url(), 391
comment_user_login(), 27
crumbpicker_preprocess_breadcrumb(), 196
db_query_temporary(), 112
drupal_add_js(), 334, 335
drupal_bootstrap(), 315
drupal_encode_path(), 395
drupal_form_submit(), 230
drupal_get_form(), 36, 43, 209, 218
drupal_get_token(), 400
drupal_goto(), 216
drupal_http_request(), 380
drupal_mail(), 51, 402
drupal_redirect_form(), 216
drupal_render(), 214
drupal_session_initialize(), 326
drupal_set_message(), 382
drupal_set_title(), 80
drupal_valid_token(), 400
drupal_write_record(), 100
emoveClass(), 337
enhanced_install(), 443
field_create_field(), 42
field_create_instance(), 165
file_copy(), 285
file_move(), 285
file_scan_directory(), 289
sygnatura, 289
file_space_used(), 292
file_unmunge_filename(), 292
file_valid_uri(), 402
filter_format_save(), 444
filter_xss(), 394
filter_xss_admin(), 395
filter_xss_bad_protocol(), 395
form_builder(), 213
form_set_value(), 226
ini_set(), 326
jQuery(document).ready(), 335
legacysearch_search(), 277
locale(), 357
locale_user_login(), 27
menu_cache_clear_all(), 313
menu_rebuild(), 91, 313
module_invoke(), 65
module_invoke_all(), 65
mysql_type, 106
node_access_acquire_grants(), 148
node_access_rebuild(), 147
node_add_body_field(), 164
node_get_types(), 40
node_page_view(), 32
node_search_execute(), 266
node_type_get_types(), 37
node_user_login(), 27
rawurlencode(), 396
request_uri(), 218
search_index(), 271
setUp(), 459
spam_user_login(), 27
st(), 364
stream_set_timeout(), 430
syslog(), 433
system_settings_form(), 38, 44
t(), 80, 354
taxonomy_select_nodes(), 304
theme(), 191
theme_breadcrumb(), 192, 194
to_arg(), 85
user_external_login_register(), 128
user_load(), 124
user_logout(), 86
user_search_execute(), 266
valid_url(), 395
variable_get(), 38, 46
variable_set(), 38, 204
xmlrpc(), 378
xmlrpc_error(), 383
funkcja przetwarzania, 260
funkcje aktualizujące, 441
funkcje haków użytkownika, 118
funkcje opakowujące, 94
funkcje taksonomii, 305
taxonomy_get_children(), 308
taxonomy_get_parents(), 307
taxonomy_get_parents_all(), 307
taxonomy_get_term_by_name(), 307
taxonomy_get_tree(), 308
taxonomy_get_vocabularies(), 305
taxonomy_load_term(), 306
taxonomy_term_delete(), 307
taxonomy_term_save(), 307
taxonomy_vocabulary_delete(), 305
taxonomy_vocabulary_load(), 305
taxonomy_vocabulary_save(), 305
funkcje testowe, 464
funkcje walidacji, 288
file_validate_extensions(), 288
file_validate_image_resolution(), 288
file_validate_is_image(), 288
file_validate_name_length(), 288
file_validate_size(), 289

G

globalny obiekt $language, 368
grep, 416

H

hak, 50
haki, 27
hook_action_info(), 53, 57
hook_block_configure(), 202
hook_block_info(), 202
hook_block_save(), 202
hook_block_view(), 202
hook_comment_insert(), 50
hook_element_info(), 211
hook_field_schema(), 157
hook_file_download(), 292
hook_filter_info(), 255
hook_flush_caches(), 321
hook_form_alter(), 229
hook_insert(), 141
hook_install, 39, 455
hook_load(), 142
hook_menu, 35, 71
hook_node_access(), 139
hook_node_access_records(), 147
hook_node_grants(), 147
hook_node_insert(), 50
hook_node_load(), 35
hook_node_update_index(), 272
hook_permission(), 78, 138, 400
hook_process_HOOK(), 178
hook_query_alter(), 110, 399
hook_schema_alter(), 109
hook_search_execute(), 269
hook_search_info(), 268
hook_search_page(), 267
hook_theme(), 143, 191
hook_trigger_info(), 62
hook_uninstall, 40
hook_update(), 141
hook_update_index(), 273
hook_user_login, 27, 50

haki
hook_user_view(), 118
hook_validate(), 140
hook_view(), 142
hook_xmlrpc(), 383
phptemplate_preprocess_breadcrumb(), 196
haki indeksera HTML, 273
haki kategorii, 302
haki uwierzytelniania do pobierania, 292
haki użytkownika, 116
haki w postaci node_xxxx, 144
haki wyszukiwania, 267
hook_search_access(), 267
hook_search_admin(), 267
hook_search_execute(), 267
hook_search_info(), 267
hook_search_reset(), 267
hook_search_status(), 267

I

identyfikator formularza, 218
annotate_admin_settings, 36
identyfikator nadania (grant ID), 148
identyfikatory działań, 61
implementacja haka, 35
indekser, 265
sposób działania, 270
zastosowania, 270
indekser HTML, 269
indeksowanie dynamicznie, 271
indeksowanie treści niebędącej segmentem, 273
index.php, 61
InnoDB, 439
instalacja domyślna, *Patrz* układ plików, 29
instrukcja jQuery, 336
interfejs
administratora, 26
Field API, 29
przypisywania wyzwalaczy, 51
ustawień związanych z obsługą plików, 280
użytkownika wyszukiwarki, 266
ISAPI, 32
ISAPI Rewrite, 32
ISO 8601 — międzynarodowy format daty, 379

J

jQuery, 331
dodawanie i usuwanie klas, 337
dołączanie kodu JavaScript, 338
kontrolka do głosowania, 343
łańcuchy wywołań, 337
określanie elementu za pomocą ID, 336
opakowywanie istniejących elementów, 337
przesłanialny kod JavaScript, 341
zmiana wartości elementów CSS, 338

K

kaskadowe arkusze stylów CSS, 28
katalog
enhanced, 441
files, 30, 280, 281
includes, 29
menu, 69
misc, 29, 352
modules, 29, 33
private, 30
profiles, 29
scripts, 30
sites, 30
system, 179
templates, 188
themes, 31
translations, 364
kategorie
przechowywanie kategorii, 301
struktura kategorii, 295
typowe zadania, 304
tworzenie własnych zapytań kategorii, 304
wyświetlanie terminów taksonomii, 304
kategorie informacji o użytkownikach, 125
kategorie wyzwalaczy, 64
klient XML-RPC, 378
klucze
access callback, 78
drupal_private_key, 211
file, 75
load arguments, 84
mysql_type, 106
page arguments, 74
pgsql_type, 106
size, 106
type, 106
klucze liczbowe, 46
klucze tablicy, 53
configurable, 53
label, 53
triggers, 53
type, 53
kod
HTML, 193, 351
JavaScript, 331
jQuery, 231
operacyjny APC, 421
PHP, 421
kompatybilność jQuery, 352
komputery równoważące obciążenie, 435
konflikt przestrzeni nazw, 147
kontekst, 58
kontrola dostępu, 78
konwencje programistyczne obowiązujące w Drupalu, 35
konwersacje w trakcie sesji, 328
konwersje pomiędzy formatami tekstowymi, 390
kopiowanie plików, 285

L

LAMP, 429
LANGUAGE_RTL, 366
liczebność, 154
listing definicji stylów CSS, 173
logowanie zewnętrzne, 126
szczegółowy przebieg, 128
lokalizacja, 353
lokalizacja i dostosowanie, 80

Ł

łącze
Moduły, 51
Struktura, 51
Wyzwalacze, 51
Zarządzaj polami, 163, 282
łączenie się z bazą danych, 94

M

mapowanie
adresów URL na funkcje, 69
typów pól schematów na typy pól bazy danych, 104
wywołań zwrotnych, 69
mechanizm
automatycznej konwersji typów, 390
cross-site scripting, 393
konwersji danych tekstowych Drupala, 390
watchdog(), 433
obsługi formularzy, 210
menu administracyjne, 35
MENU_CALLBACK, 76
MENU_DEFAULT_LOCAL_TASK, 79

MENU_NORMAL_ITEM, 76, 87
metoda
  actions_do(), 62
  currentTime.getCurrentTime(), 378
  drupalCreateUser(), 460
  fetchField(), 97
  toggleClass(), 337
  variable_get(), 46
  variable_set(), 46
metody podłączania się do haków indeksera, 272
metody przechowywania działań, 61
metody wbudowane XML-RPC
  system.getCapabilities, 386
  system.listMethods, 384
  system.methodHelp, 386
  system.methodSignature, 386
  system.multiCall, 387
migracja, 26
moduł kontroli dostępu, 148
moduł odpowiedzi, 191
moduł twórcy skórek (theme developer module), 196
moduł wyzwalaczy, 51
moduł zewnętrzny, 148
moduły, 27
  annotate, 33
  beep, 53
  Blog, 457
  blogapi, 389
  coder, 415
  comment, 27
  Content, 371
  Database logging, 433
  Date, 153
  Date API, 153
  Date Popup, 153
  devel, 43, 438
  forum_access.module, 148
  Forums, 302
  hide.module, 120
  job_post.module, 147
  legalagree.module, 120
  locale, 27, 353
  markednode.module, 109
  memcache, 326
  menu.module, 91
  mod_expires, 423
  mod_php, 421
  mod_rewrite, 31
  node, 27
  node.module, 32
  profile.module, 122
  schema, 103
  spam.module, 27
  statistics, 438
  syslog, 433
  Testing, 457
  TinyMCE, 212
  translation template extractor, 362
  Trigger, 62
  user.module, 58
  Views, 103
moduły dodatkowe, 33
moduły rdzenia, 26, 33
  ankiety, 33
  fora, 33
  kategorie, 33
  menu, 33
  subskrybent kanałów, 33
  wyszukiwarka, 33
modyfikacja działań, 59
modyfikacje rekordów, 100
modyfikator range, 98
modyfikowanie okruszków Drupala, 193
MyISAM, 439

N

narzędzie
  Alternative PHP Cache (APC), 420
  vmstat, 436
  YSlow, 437
nawigacja w kodzie Drupala, 416

O

obiekt
  #options, 46
  $user, 115
    dostęp do obiektu, 116
    elementy obiektu, 117
obsługa
  kategorii, 295
  pamięci podręcznej, 311
  plików, 279
  plików multimedialnych, 283
  plików wysyłanych, 282
  pól, 151
  segmentów, 131
  SSL, 403
  użytkowników, 115
  wielu baz danych, 110
  żądania, 31
obszar skórki, 176
ochrona konta superużytkownika, 406
odwołanie do haka, 35
odwrotne proxy, 327
ograniczanie dostępu
  do segmentów, 147
  określonego przez uprawnienia, 399
ograniczenie wyników zapytania, 399
określanie
  głębokości słowników, 300
  katalogu tymczasowego, 290
  kontekstu, 59
  tokenu, 211
operatory AND i OR w adresach URL, 299
optymalizacje Drupala, 432

P

Pager, 98
pamięci podręcznej
  czyszczenie, 43, 319, 320
  pojemniki do obsługi, 312
  stałe, 317
  wyłączanie, 314
  zastosowania, 311
pamięć podręczna
  bazy danych, 420
  bloków, 317
  Drupala, 420
  filtrów, 314
  kodu operacyjnego (opcode cache), 437
  kodu operacyjnego APC, 421
  kodu operacyjnego PHP, 419
  menu, 313
  odwrotnego proxy, 419
  przeglądarki, 423
  stron, 315
  stron anonimowych, 316
  tabel Drupala, 313
  tabeli variables, 314
parametr
  #form, 213
  #form_state, 213
  $context, 58
  $object, 58
  $replace, 285
  $reset, 320
PDO (PHP Data Object), 26
PHP-FPM (FastCGI Process Manager), 422
plik
  .htaccess, 26, 280
  .inc, 32
  .info, 32, 50, 136
  .install, 32, 101, 134
  .module, 32, 136
  .profile, 32
  .test, 32
  .theme, 32
  annotate.admin.inc, 36, 46

plik
annotate.info, 34, 46
annotate.install, 34, 39, 46
annotate.module, 34, 46
authorize.php, 31
block.tpl.php
bootstrap.inc, 32
comment.tpl.php, 191
common.inc, 380
cron.php, 31
default.settings.php, 30
enhanced.info, 441
enhanced.install, 443
atrybuty bloków, 445
atrybuty rodzajów zawartości, 448
opcje tworzenia konta użytkownika, 450
enhanced.profile, 442
field.tpl.php, 188, 189
form.inc, 214, 216
general.pot, 363
html.tpl.php, 179, 181
html.tpl.php, 182
httpd.conf, 423
index.php, 31
install.php, 31
INSTALL.txt, 434
installer.pot, 364
jquery.js, 335
konfiguracyjny VCL, 426
LICENSE.txt, 46
markednode.install, 109
menu.inc, 69, 91
menufun.module, 76
menufun_greeting.inc, 78
node.module, 266
node.tpl.php, 185, 186
page.tpl.php, 176, 181, 182, 184
pager.inc, 99
php.ini, 281
Portable Object, 361
potx.inc, 364
potx-cli.php, 364
README.txt, 46
region.tpl.php, 185
robots.txt, 31
session.inc, 326
settings.php, 30, 93, 324
sponsor.tpl.php, 143
style.css, 171
style-rtl.css, 366
template.php, 178, 195
theme.inc, 192
update.php, 31
user.module, 266
user.pages.inc, 86
web.config, 32
xmlrpc.php, 31, 377
pliki
metody udostępniania, 279
.pot, 362
prywatne, 279, 281
przenoszenie, 285
przesłanianie plików szablonów, 188, 194
publiczne, 279
samodzielne pliki PHP, 403
szablonów, 28, 191
szablonów predefiniowane, 176
tymczasowe, 279
związane z lokalizacją i tłumaczeniem, 376
pobieranie
adresu URL dla pliku, 289
dostępnych definicji elementów formularza, 211
pojedynczej wartości, 97
wielu wierszy, 97
podziału formularza odrębne zbiory pól, 219
pole, 29, 151
CAPTCHA, 228
checkbox, 38
Etykieta, 57
Liczba przypisów na pojedynczy segment, 46
parameters, 61
Przedrostek kodu języka w ścieżce, 369
Przypisy zostaną usunięte, 46
wysyłania pliku, 282
zawierające ID, 211
pole tekstowe
Prywatna ścieżka systemowa plików, 30
pole typu
Char, 104
Datetime, 106
Float, 105
Integer, 105
Liczbowe, 105
Numeric, 106
Serial, 105
Tekstowe, 104
Text, 105
Typ binarny blob, 106
Varchar, 104
zawarte w rdzeniu Drupala, 156
porównanie fragmentów kodu Drupal i jQuery, 332
Portable Object, 362
poszukiwanie
funkcji skórki dla formularza, 213
funkcji walidacji, 212
funkcji wysyłania formularza, 212
powiązania
haków i wyzwalaczy, 52
między filtrami i modułami, 258
pozyskiwanie wyników zapytań, 97
prefiks ścieżki, 369
Pressflow, 425
proces
logowania, 123
odbudowania formularza, 228
rejestracji użytkownika, 120
rozruchu (bootstrap), 32
trasowania menu, 70
tworzenia elementów trasowania i hiperłączy, 71
profil instalacji
standardowa (Standard), 441
minimalna (Minimal), 441
profilowanie kodu, 437
proste zapytania, 95
protokół XML-RPC, 377
przechowywanie ustawień, 45
przechwytywanie sesji, 326
przeglądanie treści przy użyciu terminów, 299
przekazywanie (dispatching), 69
danych z wykorzystaniem $form_state, 227
dodatkowych argumentów do funkcji ładowania, 84
przekierowanie użytkownika, 215
przepływ sterowania w procesie
logowania, 123
logowania zewnętrznego, 127
przesłanianie skórkowalnych elementów, 192
przeszukiwanie drzewa DOM, 331
przetwarzanie
formularzy, 209
formularzy z poziomu kodu, 230
wstępne (preprocessing), 270
żądania, 32
przygotowanie kontekstu przez moduł wyzwalaczy, 58
przyjazne adresy URL, 26, 32
przypisy segmentów, 42
przypisywanie
akcji, 53
wywołań zwrotnych, 89
pseudolosowy token, 211
pula procesów PHP, 422

R

rdzeń (core), 26, 147
renderowanie
  formularza, 214
  modelu DOM, 335
  strony, 270
rodzaje elementów menu, 87
rodzaje kategorii, 296
  hierarchiczna, 297
  płaska, 296
  wielohierarchiczna, 298
role aktywnego użytkownika, 327

S

schemat bazy danych, 283
schemat bazy danych dla bloków, 201
segment, 28, 131
  aktualizowanie
    danych - hook_update(), 141
  atrybut, 132
    changed, 132
    comment, 132
    created, 132
    language, 132
    nid, 132
    promote, 133
    status, 132
    sticky, 133
    title, 132
    tnid, 133
    translate, 133
    type, 132
    uid, 132
    vid, 132
  dostosowywanie formularza, 139
  dziedziczenie, 132
  modyfikowanie, 142, 144
  ograniczanie dostępu, 139
  określanie uprawnień, 147
  proces obsługi dostępu, 149
  przykładowy obiekt, 145
  sposób przechowywania, 145
  tworzenie
    modułu, 133
    piku .info, 136
    pliku .install, 134
    pliku .module, 136
    uprawnień zależnych, 138
  typ blog, 132
  typ forum, 132
  typ poll (ankieta), 132
  udostępnianie informacji, 136
    base, 137
    description, 137
    has_title, 137
    help, 137
    locked, 137
    name, 137
    title_label, 137
  usuwanie, 142
  walidacja pól, 140
  zapisywanie danych, 141
  zmiana wywołań zwrotnych menu, 137
sekcja
  Data Handling (obsługa danych), 281
  File Uploads (wysyłanie plików), 281
selektor
  #intro, 333
  klas CSS, 333
selektory jQuery, 333
separator okruszków, 195, 196
serwer
  dedykowany, 430
  pamięci podręcznej, 425
  wirtualny, 430
  XML-RPC, 382
serwer WWW, 26
  Apache, 26, 421
  IIS, 26
  lighttpd, 26
  Nginx, 425
sesja, 323
  cykl życia, 327
  sposób użycia, 323
  struktura tabeli sessions, 327
  typowe zadania, 329
    przechowywanie danych w sesji, 326, 329
    zmiana nazwy sesji, 329
    zmiana terminu ważności cookie, 329
  ustawienia, 325
    plik .htaccess, 325
    plik bootstrap.inc, 325
    plik settings.php, 325
  wymaganie cookies, 326
  zapisanie do tabeli sessions, 327
silnik MySQL InnoDB, 431
silnik wyszukiwarki Drupala, 265
skórki, 28, 169
  dodawanie obszarów, 176
  dodawanie plików CSS, 176
  dodawanie skryptów języka JavaScript, 177
  dodawanie ustawień, 177
  instalacja gotowej skórki, 169
  odbudowa rejestru skórek, 194
  Pixture Reloaded, 170
  plik .info, 176
  pliki szablonów, 179, 180
    block.tpl.php, 180
    field.tpl.php, 180
    node.tpl.php, 180
  schemat tworzenia skórki, 170
  skórki bazowe (starter themes), 170
skrypt sysctl_set.sh, 428
słowniki budowane w modułach, 302
słowniki kategorii, 296
słuchacz zdarzeń, 349
specjalne argumenty ładowania
  %map, 84
  %index, 84
sprawdzanie
  katalogów, 285
  przestrzeni dyskowej, 292
  stanu logowania użytkownika, 116
  czy formularz został wysłany, 213
stos technologii Drupala, 26
strona
  Dodaj treść/Prosta strona, 371
  Działania, 55
  Dziennik i błędy, 433
  Formaty tekstu,, 258
  Interfejs tłumaczenia, 361
  Języki, 365
  Kategorie, 295
  Konfiguracja, 42
  Menu, 77
  Moduły, 34, 108, 353
  Podstawowe dane, 395
  Rodzaje zawartości, 147, 151, 163, 282, 296, 371
  Schema, 103
  Struktura, 53
  System plików, 280
  Testowanie, 457
  Treść, 371
  Wydajność, 313
strona administracji treścią, 372
struktura drzewiasta formularza, 37
strumienie opakowujących (stream wrapper), 279
styl kodowania, 415
superużytkownik, 139
system
  filtrów, 255, 259
  menu, 69
    dostosowanie menu, 69
    kontrola dostępu, 69
    mapowanie wywołań zwrotnych, 69
  plików ramfs, 428
  plików sieci lokalnej, 428
  skórek, 169
  szablonów PHP, 28

SZBD (system zarządzania bazą danych), 26
MySQL, 26
PostgreSQL, 26
SQLite, 26
szybkie systemy plików, 429

Ś

ścieżka do mapy pamięci, 422
ścieżka pliku pamięci podręcznej, 422

T

tabela
actions, 61
actions_aid, 61
authmap, 129
block – wykaz kolumn, 200
cache - schemat, 312
cache_block, 313
cache_bootstrap, 313
cache_field, 313
cache_filter, 313
cache_form, 313
cache_image, 313
cache_menu, 313
cache_page, 313
cache_path, 313
cache_update, 313
field_data_<nazwa_pola>, 42
file_managed, 279
locales_source, 357
locales_target, 357
login_history, 125
menu_links, 91
menu_router, 91
node, 132, 145
node_access, 147
node_revisions, 132
sessions, 115, 324
struktura tabeli file_managed, 283
system, 108
wysyłania, 284
taxonomy_vocabulary, 302
technote, 273
variables, 38, 45
watchdog, 433
tabele
indeksowane, 270
taksonomii Drupala, 301
tymczasowe, 112
usuwane przy deinstalacji modułu, 108
znormalizowane, 301
tablica
$_POST, 215
$_SESSION, 279
$database, 94
tablica asocjacyjna, 100
tablica zagnieżdżona, 217
taksonomia, 295
technologia AJAX, 343
technologia jQuery, 331
tłumaczenie
interfejsu użytkownika, 353
łańcuchów znaków za pomocą funkcji t(), 354
proces tłumaczenia, 355
w trybie dokładnie na czas (just-in-time), 357
zamiana łańcucha znaków, 354
eksportowanie, 361
przesłanianie, 355
zamiana za pomocą modułu locale, 357
tłumaczenie treści, 370
tokenizacja, 269
trasowanie (routing), 69
trigger, *Patrz* wyzwalacz 50, 52
tworzenie
bloku, 202
elementu menu, 71
funkcji przetwarzania, 229
funkcji skórek
wstępne przetwarzanie, 222
wybór funkcji, 222
funkcji walidacji, 225
plików, 33
prostych formularzy, 216
rodzajów zawartości, 151
słownika w obrębie modułu, 302
ścieżek na podstawie znaków wieloznacznych, 85
tabel, 101
testów, 459
typu segmentu z własnymi typami zawartości, 147
własnego pola, 156
własnej strony wyszukiwarki, 265
własnych uprawnień, 400
własnych wyzwalaczy, 62
wywołania zwrotnego tytułu, 80
typy danych tekstowych, 107
tekst HTML (HTML text), 391
tekst rozbudowany (Rich text), 391
zwykły tekst (Plain text), 390

U

udostępnienie wyszukiwarce aliasów ścieżek, 268
układ plików, 29
ukrywanie elementów menu, 91
uprawnienie
administer nodes, 139
administer site configuration, 395
create job, 139
delete own job, 139
edit any job, 139
edit own job, 139
elete any job, 139
rate content, 343
receive greeting, 78
usługa LDAP, 126
usprawnienia wprowadzane do puli procesów PHP, 422
ustawienia PHP, 281
ustawienia wersji językowych, 372
użytkownik skojarzony z sesją, 327

V

Varnish, 425
normalizacja żądań przychodzących, 426
znajdywanie dodatkowych cookies, 427

W

waga, 271
walidacja
dla typu elementu, 215
formularza, 214
pojedynczych elementów, 228
tokenu, 215
wbudowana, 215
walidacja ustawień, 44
warstwa abstrakcji bazy danych, 26
warstwa stosu, 26
wąskie gardła, 436
wątki, 424, 426
wbudowany mechanizm obsługi sesji, 325
wewnętrzna ścieżka, 31
własny profil instalacji, 441
właściwości
#markup, 221
#prefix, 221
#suffix, 221

właściwości formularza, 218, 237
#access, 238
#action, 237
#after_build, 238
#array_parents, 238
#attached, 238
#attributes, 237
#built, 237
#default_value, 239
#description, 237
#disabled, 239
#element_validate, 239
#method, 237
#parents, 239
#post_render, 239
#pre_render, 239
#prefix, 239
#process, 239
#required, 238
#states, 239
#suffix, 240
#theme, 240
#theme_wrappers, 240
#title, 240
#tree, 238, 240
#type, 238
#weight, 240
włączanie bloku w zainstalowanych modułach, 207
współdzielony, zamontowany system plików, 435
wtyczka Firebug, 171
wtyczka IE Developers Toolbar, 171
wybór funkcji skórki, 223
wybór funkcji walidacji, 224
wybór języka, 367
wydajność Drupala
wyłączenie niewykorzystywanych modułów, 432
zmiana dyrektywy ErrorDocument, 432
wydarzenie, 151
wykorzystywanie
haków bloku, 202
identyfikatorów CSS, 333
jQuery w Drupalu, 334
selektora klasy CSS, 333
zachowań Drupala, 351
wysyłanie formularza, 215
wysyłanie plików, 286
wyszukiwanie elementów w dokumencie, 333
wyświetlanie
danych, 32
elementów menu jako zakładek, 89
formularza ustawień, 43
wywołania zwrotne, 35, 69
strony w innych plikach, 75
walidacji, 215
wyzwalacz, 50, 52

X

XML-RPC, 377

Z

zachowania (behaviors), 351
zaciemnianie, 377
zaciemnianie nazw plików, 291
zadania serwera WWW, 31
zagnieżdżanie menu, 77
zamiana łańcuchów znaków w ścieżce okruszków, 354
zapisywanie danych do pliku, 284
zapytania
aktualizujące — UPDATE, 95
obiekty zapytań (query objects), 97
pobierające —SELECT, 95
pobieranie ograniczonego zbioru wyników, 98
przykłady, 99
stronicowanie wyników, 98
twórca zapytań (query builder), 97
usuwające — DELETE, 95
wstawiające — INSERT, 95
zarządzanie tabelami, 108
zastępowanie
elementów menu w innych modułach, 86
hiperłączy menu w innych modułach, 87
zbieranie informacji
o użytkownikach, 122
zdalne wywołanie procedury (RPC), 377
zdarzenia, 49
wykaz zdarzeń Drupala, 49
zewnętrzne systemy uwierzytelniania
Lightweight Directory Access Protocol, 115
Pubcookie, 115
zewnętrzny system przechowywania danych, 436
zmiana
$breadcrumb_delimiter, 195
$classes, 178
$form, 211
$form_id, 211
$form_state, 211, 227
$language, 368
$options, 37
$time, 379
$user, 60
PHPSESSID, 326
Table_locks_immediate, 440
Table_locks_waited, 440
user_picture_path, 287
wyzwalaczy wywołujących działanie, 53
znacznik
CVS, 34
PHP, 34
znajdywanie plików w katalogu, 289
znaki wieloznaczne, 81
znak %, 81
wartość dopasowana, 82
zamiana parametrów, 82
parametry wywołań zwrotnych strony, 82

Ż

żądanie HTTP POST, 378

Drupal 7. Zaawansowane programowanie

Drupal to nie tylko kolejny, nieco lepszy od innych CMS. Ten system zarządzania treścią zrobił ostatnio oszałamiającą karierę jako doskonały framework do tworzenia wszelkiej maści aplikacji internetowych. Dzięki niezwykłej łatwości i szybkości, z jaką Drupal pozwala programistom tworzyć rozbudowane blogi, profesjonalne witryny korporacyjne czy serwisy społecznościowe, zainteresowanie tą technologią zaczęło dynamicznie wzrastać. Co więcej, choć już dziś wydaje się, że sposoby wykorzystania tego CMS-a ogranicza jedynie ludzka wyobraźnia, system jest wciąż intensywnie rozwijany przez liczną społeczność entuzjastów na całym świecie. Możliwość czerpania z ogromnych zasobów wiedzy tysięcy programistów poprawi jakość Twoich nawet najbardziej wyrafinowanych internetowych projektów.

Jeśli chcesz tworzyć nowoczesne, rozbudowane witryny internetowe z wykorzystaniem Drupala 7, właśnie znalazłeś idealny podręcznik dla siebie! Omówiono tu wszystko, co będzie Ci potrzebne, począwszy od podstawowych zagadnień, takich jak architektura i struktura plików systemu, przez sposoby wykorzystania API formularzy czy dodawania własnych modułów w celu rozszerzania możliwości, aż po metody tworzenia bezpiecznego, wydajnego kodu. Dowiesz się więcej na temat pracy z bazą danych, uruchomienia własnej strony wyszukiwarki oraz korzystania z jQuery w Drupalu. Nauczysz się także sprawnie optymalizować Drupal i przeprowadzać skuteczne testy oraz poznasz wiele innych praktycznych zagadnień, które sprawią, że bez trudu zrozumiesz zasady działania tego CMS-a oraz pokonasz wszelkie problemy napotykane w trakcie pracy.

- Jak działa Drupal i jaką ma architekturę
- Jak tworzyć dodatkowe, funkcjonalne moduły
- Jak pracować z bazą danych i API formularzy
- Jak wygląda obsługa użytkowników, segmentów i pól
- Jak zapewnić wyszukiwanie i indeksowanie informacji
- Jak tworzyć czysty, bezpieczny kod
- Jak przechowywać dane w sesjach
- Jak optymalizować działanie Drupala
- Jak przeprowadzić efektywny proces testowania
- Jak przygotowywać testy jednostkowe

Nr katalogowy: 6872

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

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

helion.pl
księgarnia internetowa

Cena 79,00 zł
ISBN 978-83-246-3367-8
9 788324 633678

Informatyka w najlepszym wydaniu