Kompendium wiedzy o platformie Java EE6!

Java EE6

Zaawansowany przewodnik

Wydanie IV

Eric Jendrock, Ricardo Cervera-Navarro, Ian Evans, Devika Gollapudi,
Kim Haase, William Markito, Chinmayee Srivathsa

Helion

ORACLE

Tytuł oryginału: The Java EE 6 Tutorial: Advanced Topics (4th Edition)

Tłumaczenie: Rafał Jońca

ISBN: 978-83-246-7393-3

Authorized translation from the English language edition, entitled: THE JAVA EE 6 TUTORIAL: ADVANCED TOPICS, Fourth Edition; ISBN 0137081863; by Eric Jendrock; and by Ian Evans; and by Devika Gollapudi; and by Kim Haase; and by Chinmayee Srivathsa; and by Ricardo Cervera-Navarro; and by William Markito; published by Pearson Education, Inc, publishing as Prentice Hall.

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

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/jee6z4
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/jee6z4.zip

Printed in Poland.

Spis treści

Wstęp 15

Część I. Wprowadzenie 19

Rozdział 1. Przegląd technologii 21

Najważniejsze cechy platformy Java EE 6 22
Model aplikacji Javy EE 23
Rozproszone, wielowarstwowe aplikacje 23
Bezpieczeństwo 25
Komponenty Javy EE 25
Klienty Javy EE 25
Komponenty webowe 27
Komponenty biznesowe 28
Warstwa danych 29
Kontenery Javy EE 29
Usługi kontenera 30
Typy kontenerów 30
Obsługa usług sieciowych 31
XML 32
Protokół transportowy SOAP 32
Standard formatu WSDL 33
Budowanie i wdrażanie aplikacji Javy EE 33
Pakowanie aplikacji 33
Role w procesie wytwarzania aplikacji 35
Dostawca oprogramowania Java EE 36
Dostawca narzędzi 36
Dostawca komponentów aplikacji 36
Budowniczy aplikacji 37
Wdrożeniowiec oraz administrator 37
API Javy EE 6 37
Technologia Enterprise JavaBeans 39
Technologia Java Servlet 40
Technologia JavaServer Faces 41
Technologia JavaServer Pages 42
Biblioteka JavaServer Pages Standard Tag Library 42
Java Persistence API 42

Java Transaction API ... 43
API Javy dla usług sieciowych typu REST ... 43
Managed Beans ... 43
Contexts and Dependency Injection for the Java EE Platform (JSR 299) ... 44
Dependency Injection for Java (JSR 330) ... 44
Bean Validation ... 44
Java Message Service API ... 44
Architektura Java EE Connector ... 45
JavaMail API ... 45
Java Authorization Contract for Containers ... 45
Java Authentication Service Provider Interface for Containers ... 46
API Javy EE 6 wchodzące w skład platformy Java Platform, Standard Edition 6 i 7 ... 46
Java Database Connectivity API ... 46
Java Naming and Directory Interface API ... 47
JavaBeans Activation Framework ... 47
Java API for XML Processing ... 48
Java Architecture for XML Binding ... 48
SOAP with Attachments API for Java ... 48
Java API for XML Web Services ... 48
Java Authentication And Authorization Service ... 49
Narzędzia serwera GlassFish ... 49

Rozdział 2. Używanie przykładowych aplikacji z samouczka ... 51

Wymagane oprogramowanie ... 51
Java Platform, Standard Edition ... 51
Java EE 6 Software Development Kit ... 52
Komponent samouczka Javy EE 6 ... 53
NetBeans IDE ... 53
Apache Ant ... 54
Uruchamianie i zatrzymywanie serwera GlassFish ... 55
▼ Uruchamianie serwera GlassFish z poziomu środowiska NetBeans IDE ... 56
Uruchamianie konsoli administracyjnej ... 56
▼ Uruchamianie konsoli administracyjnej w środowisku NetBeans IDE ... 56
Uruchamianie i zatrzymywanie serwera Java DB ... 56
▼ Uruchamianie serwera bazy danych przy użyciu środowiska NetBeans IDE ... 57
Budowanie przykładowych aplikacji ... 57
Struktura katalogów z przykładami samouczka ... 57
Pobieranie najnowszych aktualizacji samouczka ... 58
▼ Aktualizacja samouczka za pomocą centrum aktualizacji ... 58
Debugowanie aplikacji Javy EE ... 59
Używanie logów serwera ... 59
Używanie debuggera ... 59

Część II. Warstwa webowa 61

Rozdział 3. Technologia JavaServer Faces — tematy zaawansowane 63

Cykl życia aplikacji JavaServer Faces 63
Omówienie cyklu życia JavaServer Faces 64
Faza przywrócenia widoku 66
Faza zastosowania wartości zapytania 67
Faza przetwarzania walidacji 68
Faza aktualizacji wartości modelu 68
Faza wywołania aplikacji 69
Faza renderowania odpowiedzi 69
Częściowe przetwarzanie i rendering 70
Cykl życia aplikacji faceletów 70
Model komponentów interfejsu użytkownika 71
Klasy komponentów interfejsu użytkownika 71
Model renderingu komponentów 73
Model konwersji 74
Model zdarzeń i procesów ich obsługi 75
Model walidacji 77
Model nawigacji 78

Rozdział 4. Wykorzystanie technologii Ajax wraz z technologią JavaServer Faces 81

Technologia Ajax — wprowadzenie 82
Wykorzystanie technologii Ajax wraz z technologią JavaServer Faces 83
Wykorzystanie technologii Ajax wraz z faceletami 83
Użycie znacznika f:ajax 84
Wysłanie żądania Ajax 86
Użycie atrybutu event 86
Użycie atrybutu execute 86
Użycie atrybutu immediate 87
Użycie atrybutu listener 87
Monitorowanie zdarzeń po stronie klienta 87
Obsługa błędów 88
Otrzymywanie odpowiedzi Ajax 89
Cykl życia żądania Ajax 90
Grupowanie komponentów 90
Wczytywanie kodu JavaScript jako zasobu 91
Użycie API dla kodu JavaScript w aplikacji z faceletami 91
Użycie adnotacji @ResourceDependency w klasie ziarna 92
Przykładowa aplikacja ajaxguessnumber 93
Pliki źródłowe 93
Uruchomienie przykładu ajaxguessnumber 95
Dodatkowe informacje na temat użycia technologii Ajax wraz z technologią JavaServer Faces 96

Rozdział 5. Komponenty złożone — tematy zaawansowane i przykłady 97
Atrybuty komponentu złożonego 97
Wywoływanie zarządzanego ziarna 98
Walidacja wartości komponentu złożonego 98
Przykładowa aplikacja compositecomponentlogin 99
Plik komponentu złożonego 99
Strona wykorzystująca komponent 100
Zarządzane ziarno 100
Uruchomienie przykładu compositecomponentlogin 101

Rozdział 6. Tworzenie własnych komponentów UI i innych obiektów 103
Określanie, czy potrzebny jest własny komponent lub renderer 105
Kiedy użyć własnego komponentu? 105
Kiedy zastosować własny renderer? 107
Kombinacje komponentów, rendererów i znaczników 107
Analiza przykładu z mapą obrazu 108
Dlaczego mam korzystać z technologii JavaServer Faces do implementacji mapy obrazu? 109
Działanie zrenderowanego kodu HTML 109
Omówienie strony faceletu 110
Konfiguracja danych modelu 111
Podsumowanie klas mapy obrazu 113
Kroki niezbędne do utworzenia własnego komponentu 113
Tworzenie własnych klas komponentów 114
Określenie rodziny komponentu 117
Przeprowadzenie kodowania 117
Przeprowadzenie dekodowania 119
Umożliwienie właściwościom komponentu przyjmowania wyrażeń 120
Zapis i przywracanie stanu 121
Przekazanie renderowania do renderera 122
Tworzenie klasy renderera 123
Określenie rodzaju renderera 124
Implementacja klasy nasłuchiwania zdarzeń 125
Implementacja klasy nasłuchiwania zdarzeń zmiany wartości 125
Implementacja klas nasłuchujących akcji 126
Obsługa zdarzeń dla samodzielnie wykonanych komponentów 127
Definicja znacznika własnego komponentu w deskryptorze biblioteki znaczników 128
Użycie własnego komponentu 129
Utworzenie i użycie własnego konwertera 130
Tworzenie własnego konwertera 131
Użycie własnego konwertera 133
Utworzenie i użycie własnego walidatora 135
Implementacja interfejsu Validator 136
Określanie własnego znacznika 138
Użycie własnego walidatora 138
Wiązanie wartości i instancji komponentów z właściwościami zarządzanego ziarna 140
Powiązanie wartości komponentu z właściwością 141
Powiązanie wartości komponentu z niejawnym obiektem 142

Powiązanie instancji komponentu z właściwością ziarna ... 144
Wiązanie konwerterów, walidatorów i obsługi zdarzeń z właściwościami zarządzanego ziarna ... 145

Rozdział 7. Konfiguracja aplikacji JavaServer Faces ... 147

Wykorzystanie adnotacji do konfiguracji zarządzanych ziaren ... 148
Korzystanie z zasięgów zarządzanych ziaren ... 148
Plik zasobu konfiguracji aplikacji ... 149
Kolejność plików zasobów konfiguracji aplikacji ... 151
Konfiguracja zarządzanych ziaren ... 152
Użycie elementu managed-bean ... 153
Inicjalizacja właściwości przy użyciu elementu managed-property ... 155
Inicjalizacja odwzorowań i list ... 160
Rejestracja komunikatów aplikacji ... 160
Użycie FacesMessage do utworzenia komunikatu ... 162
Dostęp do komunikatów o błędach ... 162
Korzystanie z walidatorów domyślnych ... 163
Rejestracja własnego walidatora ... 164
Rejestracja własnego konwertera ... 164
Konfiguracja reguł nawigacyjnych ... 165
▼ Konfiguracja reguły nawigacyjnej ... 167
Niejawne reguły nawigacyjne ... 168
Rejestracja własnego renderera w zestawie rendererów ... 168
Rejestracja własnego komponentu ... 170
Podstawowe wymagania stawiane aplikacjom JavaServer Faces ... 171
Konfiguracja aplikacji przy użyciu deskryptora wdrożenia ... 172
Konfiguracja etapu projektu ... 175
Dołączanie klas, stron i innych zasobów ... 176

Rozdział 8. Przesył plików do serwera w technologii Java Servlet ... 177

Adnotacja @MultipartConfig ... 177
Metody getParts i getPart ... 178
Przykładowa aplikacja fileupload ... 179
Architektura przykładowej aplikacji ... 179
Uruchomienie przykładu fileupload ... 182

Rozdział 9. Umiędzynarodowienie i lokalizacja aplikacji webowych ... 185

Klasy umiędzynarodowienia platformy Javy ... 185
Lokalizacja komunikatów i etykiet ... 186
Określanie dostępnych języków i regionów ... 187
Określenie paczki zasobów ... 187
Pobranie komunikatów w odpowiednim języku ... 188
Formatowanie dat i czasu ... 189
Kodowanie i zestawy znaków ... 189
Zestawy znaków ... 189
Kodowanie znaków ... 190

Część III. Usługi sieciowe 193

Rozdział 10. JAX-RS — tematy zaawansowane oraz przykład 195

Adnotacje dla pól i właściwości ziarna w klasach zasobów 195
Wydobycie parametrów ścieżki 196
Wydobycie parametrów zapytania 197
Wydobycie danych formularza 197
Wydobycie typu Javy dla żądania lub odpowiedzi 198
Podzasoby i dobór zasobów w trakcie działania usługi 198
Metody podzasobów 199
Lokalizator podzasobu 199
Integracja JAX-RS z technologią EJB i CDI 200
Warunkowe żądania HTTP 201
Negocjacja treści w trakcie działania usługi 202
Użycie JAX-RS z JAXB 204
Wykorzystanie obiektów Javy do modelowania własnych danych 205
Rozpoczynanie pracy od definicji schematu XML 207
Użycie formatu JSON wraz z JAX-RS i JAXB 209
Przykładowa aplikacja customer 210
Omówienie elementów przykładowej aplikacji 210
Klasy encji Customer i Address 210
Klasa CustomerService 213
Klasy CustomerClientXML i CustomerClientJSON 215
Modyfikacja przykładu, by generował klasy encji z istniejącego schematu 217
Uruchomienie przykładu customer 219

Część IV. Komponenty EJB 225

Rozdział 11. Przykład ziarna sterowanego komunikatami 227

Omówienie przykładu simplemessage 227
Klient aplikacji simplemessage 228
Klasa ziarna sterowanego komunikatami 228
Metoda onMessage 229
Uruchomienie przykładu simplemessage 231
Obiekty administrowane związane z przykładem simplemessage 231
▼ Uruchomienie przykładu simplemessage w środowisku NetBeans IDE 231
▼ Uruchomienie przykładu simplemessage przy użyciu narzędzia Ant 232
Usunięcie administrowanych obiektów z przykładu simplemessage 233

Rozdział 12. Korzystanie z osadzonego kontenera komponentów EJB 235

Omówienie osadzonego kontenera komponentów EJB 235
Tworzenie aplikacji z EJB z możliwością osadzania 235
Uruchamianie osadzonych aplikacji 236
Tworzenie kontenera EJB 236

Wyszukiwanie referencji do ziaren sesyjnych ... 237
Wyłączanie kontenera EJB ... 238
Przykładowa aplikacja standalone ... 238
▼ Uruchomienie przykładowej aplikacji standalone ... 239

Rozdział 13. Wywoływanie metod asynchronicznych ziaren sesyjnych ... 241
Wywoływanie metod asynchronicznych ... 241
Tworzenie asynchronicznych metod biznesowych ... 242
Wywoływanie metod asynchronicznych z poziomu klientów EJB ... 243
Przykładowa aplikacja async ... 244
Architektura przykładowej aplikacji async ... 244
Uruchomienie przykładu async ... 245

Część V. Konteksty oraz wstrzykiwanie zależności na platformie Java EE ... 249

Rozdział 14. Konteksty oraz wstrzykiwanie zależności na platformie Java EE — tematy zaawansowane ... 251
Wykorzystanie alternatyw w aplikacjach CDI ... 251
Specjalizacje ... 252
Wykorzystanie w aplikacjach CDI metod produkujących, pól produkujących i metod usuwających ... 253
Użycie metody produkującej ... 254
Użycie pól produkujących do generowania zasobów ... 255
Metody usuwające ... 255
Użycie predefiniowanych ziaren w aplikacjach CDI ... 256
Wykorzystanie zdarzeń w aplikacjach CDI ... 257
Definiowanie zdarzeń ... 257
Użycie metod obserwatorów do obsługi zdarzeń ... 257
Zgłaszanie zdarzeń ... 258
Użycie interceptorów w aplikacjach CDI ... 259
Użycie dekoratorów w aplikacjach CDI ... 261
Użycie stereotypów w aplikacjach CDI ... 262

Rozdział 15. Uruchamianie zaawansowanych przykładów dotyczących kontekstów i wstrzykiwania zależności ... 265
Przykład encoder — użycie alternatyw ... 265
Interfejs Coder i jego implementacje ... 266
Strona faceletu i zarządzane ziarno ... 266
Uruchomienie przykładowej aplikacji encoder ... 268
Przykład producermethods — użycie metody produkującej do wyboru implementacji ziarna ... 270
Składniki przykładu producermethods ... 270
Uruchomienie przykładowej aplikacji producermethods ... 272
Przykład producerfields — użycie pól produkujących do generowania zasobów ... 273
Pole produkujące w przykładzie producerfields ... 273
Ziarno sesyjne i encja producerfields ... 274

Zarządzane ziarno i strony faceletów ... 276
Uruchomienie przykładowej aplikacji producerfields ... 277
Przykład billpayment — zdarzenia i interceptory ... 279
Klasa zdarzenia PaymentEvent ... 279
Klasa nasłuchiwania zdarzeń PaymentHandler ... 280
Strony faceletów i zarządzane ziarno przykładu billpayment ... 280
Klasa interceptora LoggedInterceptor ... 283
Uruchomienie przykładowej aplikacji billpayment ... 284
Przykład decorators — dekorowanie ziarna ... 285
Elementy przykładu decorators ... 285
Uruchomienie przykładowej aplikacji decorators ... 286

Część VI. Java Persistence API ... 289

Rozdział 16. Tworzenie i używanie tekstowych zapytań z kryteriami ... 291
Wprowadzenie do zapytań tekstowych w Criteria API ... 291
Tworzenie zapytań tekstowych ... 292
Wykonywanie zapytań tekstowych ... 293

Rozdział 17. Sterowanie współbieżnym dostępem do danych encji przy użyciu blokad ... 295
Omówienie blokowania encji i współbieżności ... 295
Użycie blokad optymistycznych ... 296
Tryby blokad ... 297
Ustawienie trybu blokady ... 298
Użycie blokad pesymistycznych ... 298

Rozdział 18. Wykorzystanie pamięci cache drugiego poziomu w aplikacjach Java Persistence API ... 301
Wprowadzenie do pamięci cache drugiego poziomu ... 301
Określanie, czy encje można umieścić w cache ... 302
Określanie ustawień trybu cache w celu poprawy wydajności ... 303
Ustawienie trybu pobierania i zapisu w cache ... 303
Sterowanie cache drugiego poziomu w sposób programowy ... 305

Część VII. Bezpieczeństwo ... 307

Rozdział 19. Bezpieczeństwo w Javie EE — tematy zaawansowane ... 309
Korzystanie z certyfikatów cyfrowych ... 309
Tworzenie certyfikatu serwera ... 310
Dodanie użytkowników do dziedziny certyfikatu ... 313
Użycie innego certyfikatu serwera w serwerze GlassFish ... 313
Mechanizmy uwierzytelniania ... 314
Uwierzytelnianie klienta ... 314
Uwierzytelnianie wzajemne ... 314

Logowanie przy użyciu formularza w aplikacjach JavaServer Faces 318
Użycie j_security_check w formularzach JavaServer Faces 318
Użycie zarządzanego ziarna do uwierzytelniania w aplikacjach JavaServer Faces 319
Uwierzytelnianie za pomocą JDBC Realm 321
▼ Konfiguracja dziedziny uwierzytelniania JDBC 321
Zabezpieczanie zasobów HTTP 325
Zabezpieczenie klientów aplikacji 328
Użycie modułów logowania 328
Użycie logowania programowego 329
Zabezpieczanie aplikacji typu EIS 329
Logowanie zarządzane przez kontener 330
Logowanie zarządzane przez komponent 330
Konfiguracja bezpieczeństwa adapterów zasobów 331
▼ Odwzorowanie zarządcy aplikacji na zarządcę EIS 332
Konfiguracja bezpieczeństwa przy użyciu deskryptorów wdrożenia 333
Określanie w deskryptorze wdrożenia zasad bezpieczeństwa dla prostego uwierzytelniania 333
Wskazanie w deskryptorze wdrożenia niedomyślnego odwzorowania zarządzającego na rolę 334
Dalsze informacje na temat bezpieczeństwa 334

Część VIII. Technologie wspomagające Javę EE 335

Rozdział 20. Zagadnienia technologii JMS 337

Wprowadzenie do JMS API 337
Czym jest przekazywanie komunikatów? 337
Czym jest JMS API? 338
Kiedy mogę użyć JMS API? 338
W jaki sposób JMS API współpracuje z platformą Javy EE? 339
Podstawowe koncepcje JMS API 340
Architektura JMS API 340
Dziedziny komunikatów 341
Konsumpcja komunikatów 343
Model programistyczny JMS API 344
Obiekty administracyjne JMS 344
Połączenia JMS 347
Sesje JMS 347
Producenty komunikatów JMS 348
Konsumenty komunikatów JMS 349
Komunikaty JMS 351
Przeglądarki kolejek JMS 353
Obsługa wyjątków JMS 353
Tworzenie wydajnych aplikacji JMS 354
Użycie prostych mechanizmów trwałości 355
Użycie zaawansowanych mechanizmów niezawodności 359

Wykorzystanie JMS API w aplikacjach Javy EE 363
Użycie adnotacji @Resource w komponentach webowych i EJB 364
Użycie ziaren sesyjnych do produkcji i synchronicznego otrzymywania komunikatów 364
Wykorzystanie ziaren sterowanych komunikatami do asynchronicznego otrzymywania komunikatów 365
Zarządzanie transakcjami rozproszonymi 368
Korzystanie z JMS API w klientach aplikacji i komponentach webowych 370
Dodatkowe informacje na temat JMS 370

Rozdział 21. Przykłady wykorzystania JMS 371

Tworzenie prostych aplikacji JMS 371
Prosty przykład synchronicznego otrzymywania komunikatów 372
Prosty przykład asynchronicznego otrzymywania komunikatów 381
Prosty przykład przeglądania komunikatów w kolejce 386
Uruchomienie klientów JMS na wielu systemach 391
Usunięcie wdrożenia i wyczyszczenie systemu po przykładach JMS 397
Tworzenie rozbudowanych aplikacji JMS 397
Przykład potwierdzania komunikatu 397
Przykład trwałej subskrypcji 400
Przykład transakcji lokalnej 402
Aplikacja stosująca JMS API w połączeniu z ziarnem sesyjnym 407
Tworzenie komponentów aplikacji dla przykładu clientsessionmdb 407
Tworzenie zasobów dla przykładu clientsessionmdb 410
Uruchomienie przykładu clientsessionmdb 410
Aplikacja stosująca JMS API w połączeniu z encją 411
Opis przykładowej aplikacji clientmdbentity 412
Tworzenie komponentów przykładowej aplikacji clientmdbentity 413
Tworzenie zasobów dla przykładu clientmdbentity 415
Uruchomienie przykładu clientmdbentity 416
Przykład aplikacji konsumującej komunikaty z zewnętrznego serwera 419
Opis modułów przykładu consumeremote 419
Tworzenie komponentów modułu dla przykładu consumeremote 420
Tworzenie zasobów dla przykładu consumeremote 421
Użycie dwóch serwerów aplikacji dla przykładu consumeremote 421
Uruchomienie przykładu consumeremote 421
Przykład aplikacji wdrażającej ziarno sterowane komunikatami na dwóch serwerach 425
Opis modułów przykładu sendremote 425
Tworzenie komponentów modułów dla przykładu sendremote 426
Tworzenie zasobów dla przykładu sendremote 428
▼ Uruchomienie wdrożenia na zdalnym serwerze 428
▼ Użycie dwóch serwerów aplikacji dla przykładu sendremote 429
Uruchomienie przykładu sendremote 429
▼ Uruchomienie przykładu sendremote w środowisku NetBeans IDE 430
▼ Uruchomienie przykładu sendremote przy użyciu narzędzia Ant 432
▼ Wyłączenie wdrażania na zdalnym systemie 434

Rozdział 22. Mechanizm Bean Validation — tematy zaawansowane 435
Tworzenie własnych ograniczeń 435
Użycie wbudowanych ograniczeń do tworzenia własnych ograniczeń 435
Dostosowywanie komunikatów walidatorów 436
Paczka zasobów ValidationMessages 436
Grupowanie ograniczeń 437
Dostosowanie kolejności walidacji grup 437

Rozdział 23. Wykorzystanie interceptorów Javy EE 439
Wprowadzenie do interceptorów 439
Klasy interceptorowe 440
Cykl życia interceptorów 440
Interceptory i CDI 441
Użycie interceptorów 441
Wywołania metod przechwytujących 441
Przechwytywanie zdarzeń wywołań zwrotnych cyklu życia 443
Przechwytywanie zdarzeń upłynięcia czasu 444
Przykładowa aplikacja interceptor 445
Uruchomienie przykładu interceptor 446

Rozdział 24. Przykład z adapterem zasobów 449
Adapter zasobów 449
Ziarno sterowane komunikatami 450
Aplikacja webowa 450
Uruchomienie przykładu mailconnector 450
▼ Przygotowania przed wdrożeniem przykładu mailconnector 450
▼ Zbudowanie, spakowanie i wdrożenie przykładu mailconnector w środowisku NetBeans IDE 451
▼ Zbudowanie, spakowanie i wdrożenie przykładu mailconnector przy użyciu narzędzia Ant 452
▼ Uruchomienie przykładu mailconnector 452

Część IX. Przykładowe scenariusze użycia 453

Rozdział 25. Przykładowy scenariusz użycia — księgarnia Duke'a 455
Projekt i architektura księgarni Duke'a 455
Interfejs aplikacji 456
Encja Book korzystająca z Java Persistence API 456
Komponenty EJB wykorzystywane w przykładzie 457
Strony faceletów i zarządzane ziarna używane w przykładowej aplikacji 457
Własne komponenty i inne własne obiekty wykorzystywane w księgarni Duke'a 459
Pliki właściwości używane przez aplikację 459
Deskryptory wdrożeń użyte w przykładowej aplikacji 460

Uruchomienie aplikacji księgarnia Duke'a 461
▼ Budowanie i wdrażanie aplikacji w środowisku NetBeans IDE 461
▼ Budowanie i wdrażanie aplikacji przy użyciu narzędzia Ant 461
▼ Uruchomienie aplikacji 462

Rozdział 26. Przykładowy scenariusz użycia — ćwiczenia Duke'a 463
Projekt i architektura aplikacji 463
Interfejs główny 465
Encje Java Persistence API wykorzystywane przez interfejs główny 465
Komponenty EJB wykorzystywane w interfejsie głównym 466
Pliki faceletów wykorzystywane w interfejsie głównym 466
Klasy pomocnicze używane w interfejsie głównym 467
Pliki właściwości 468
Deskryptory wdrożenia wykorzystywane przez aplikację 469
Interfejs administracyjny 469
Komponenty EJB wykorzystywane przez interfejs administracyjny 469
Pliki faceletów wykorzystywane przez interfejs administracyjny 470
Uruchomienie przykładowej aplikacji ćwiczenia Duke'a 470
Konfiguracja serwera GlassFish 470
Uruchomienie aplikacji ćwiczenia Duke'a 471

Rozdział 27. Przykładowy scenariusz użycia — las Duke'a 475
Projekt i architektura aplikacji las Duke'a 476
Projekt events 478
Projekt entities 479
Projekt dukes-payment 481
Projekt dukes-resources 481
Projekt sklepu Duke'a 481
Projekt wysyłka Duke'a 486
Budowanie i wdrażanie aplikacji las Duke'a 488
Zadanie przygotowawcze 489
▼ Zbudowanie i wdrożenie aplikacji las Duke'a w środowisku NetBeans IDE 489
▼ Zbudowanie i wdrożenie aplikacji las Duke'a przy użyciu narzędzia Ant 490
Uruchamianie aplikacji las Duke'a 491
▼ Rejestracja jako klient sklepu 491
▼ Zakup produktów 491
▼ Zatwierdzenie wysyłki produktu 492
▼ Utworzenie nowego produktu 492

Skorowidz 495

◆ ◆ ◆ 8

ROZDZIAŁ 8

Przesył plików do serwera w technologii Java Servlet

Obsługa przesyłania plików do serwera to jeden z wymogów stawianych wielu aplikacjom internetowym. Przed udostępnieniem technologii Servlet 3.0 implementacja przesyłania plików do serwera wymagała użycia zewnętrznych bibliotek lub złożonego przetwarzania danych wejściowych. Wersja 3.0 specyfikacji Java Servlet pomaga zapewnić ogólne i elastyczne rozwiązanie tego problemu. Specyfikacja Servlet 3.0 od samego początku obsługuje przesyłanie plików, więc każdy kontener webowy zgodny ze specyfikacją może obsługiwać żądania wieloczęściowe i udostępniać załączniki MIME w obiekcie `HttpServletRequest`.

Nowa adnotacja — `javax.servlet.annotation.MultipartConfig` — służy do wskazania, że serwlet, przy którym wystąpi, oczekuje żądań wykorzystujących typ MIME `multipart/form-data`. Serwlety z adnotacją `@MultipartConfig` mogą pobierać komponenty `Part` żądania typu `multipart/form-data`, wywołując metodę `request.getPart(String name)` lub `request.getParts()`.

W rozdziale:

- Adnotacja @MultipartConfig,
- Metody getParts i getPart,
- Przykładowa aplikacja fileupload.

Adnotacja @MultipartConfig

Adnotacja `@MultipartConfig` obsługuje następujące atrybuty opcjonalne.

- `location` — pełna ścieżka do folderu w systemie plików. Atrybut `location` **nie obsługuje** ścieżek względnych (określających położenie względem kontekstu aplikacji). Wskazany folder służy do zapisywania plików tymczasowych w trakcie przetwarzania części żądania lub gdy wielkość pliku przekroczy rozmiar wskazany w atrybucie `fileSizeThreshold`. Lokalizacją domyślną jest `""`.
- `fileSizeThreshold` — rozmiar pliku w bajtach, po przekroczeniu którego plik będzie tymczasowo przechowywany na dysku twardym. Domyślny rozmiar to 0 bajtów.

- `MaxFileSize` — maksymalny rozmiar plików przyjmowanych przez serwer w bajtach. Jeśli rozmiar przesyłanego do serwera pliku jest większy od wskazanej wartości, kontener webowy zgłosi wyjątek (`IllegalStateException`). Domyślnie rozmiar nie jest w żaden sposób ograniczony.
- `maxRequestSize` — maksymalny rozmiar całego żądania typu `multipart/form-data` podany w bajtach. Kontener webowy powinien zgłosić wyjątek, jeśli łączny rozmiar wszystkich przesyłanych plików przekroczy wskazany próg. Domyślnie rozmiar nie jest w żaden sposób ograniczony.

Przykładowa adnotacja `@MultipartConfig` mogłaby wyglądać następująco:

```
@MultipartConfig(location="/tmp", fileSizeThreshold=1024*1024,
maxFileSize=1024*1024*5, maxRequestSize=1024*1024*5*5)
```

Zamiast umieszczać w adnotacji `@MultipartConfig` wszystkie wartości na sztywno, można skorzystać z odpowiedniego elementu zagnieżdżonego wewnątrz elementu konfigurującego serwlet w pliku *web.xml*. Oto przykład:

```
<multipart-config>
   <location>/tmp</location>
   <max-file-size>20848820</max-file-size>
   <max-request-size>418018841</max-request-size>
   <file-size-threshold>1048576</file-size-threshold>
</multipart-config>
```

Metody getParts i getPart

Serwlety w wersji 3.0 obsługują w obiekcie `HttpServletRequest` dwie dodatkowe metody:

- `Collection<Part> getParts()`,
- `Part getPart(String name)`.

Metoda `request.getParts()` zwraca kolekcję ze wszystkimi obiektami `Part`. Jeśli użytkownik przesyła więcej niż jeden plik (strona zawierała kilka pól wejściowych typu `file`), zwracanych jest kilka obiektów `Part`. Ponieważ obiekty `Part` posiadają nazwy, do pobrania konkretnego obiektu `Part` można posłużyć się metodą `getPart(String name)`. Ewentualnie można skorzystać z metody `getParts()`, która zwraca obiekt `Iterable<Part>`, a następnie użyć iteratora, by przejść w pętli przez wszystkie obiekty `Part`.

Interfejs `javax.servlet.http.Part` nie jest skomplikowany i deklaruje metody umożliwiające introspekcję każdej z części. Metody dotyczą:

- pobrania nazwy, rozmiaru i typu pliku związanego z obiektem `Part`,
- pobrania nagłówków wysłanych z plikiem,
- usunięcia przesłanego pliku,
- zapisania przesłanego pliku na dysku twardym.

Przykładowo interfejs `Part` definiuje metodę `write(String filename)` do zapisu pliku pod określoną nazwą. Plik można zapisać w folderze wskazywanym przez atrybut `location` z adnotacji `@MultipartConfig` lub, tak jak w przypadku aplikacji *fileupload*, w lokalizacji wskazanej przez pole formularza.

Przykładowa aplikacja fileupload

W przykładowej aplikacji *fileupload* pokazujemy, w jaki sposób zaimplementować funkcjonalność przesyłu plików do serwera i jak jej użyć.

Bardziej rozbudowany przykład przesyłania plików obrazów i ich zapisu w bazie danych znaleźć można w przykładzie z lasem Duke'a.

Architektura przykładowej aplikacji

Przykładowa aplikacja *fileupload* korzysta z pojedynczego serwletu i formularza HTML umożliwiającego przesyłanie plików do serwera.

Strona HTML składa się z prostego formularza z dwoma polami: *Plik* i *Lokalizacja docelowa*. Pole wejściowe typu `file` umożliwia użytkownikowi wskazanie w przeglądarce internetowej pliku z własnego komputera. Plik jest następnie przesyłany do serwera jako część żądania typu `POST`. Pojawienie się pola wejściowego typu `file` nakłada na formularz dwa wymagania:

- atrybut `enctype` musi zostać ustawiony na wartość typu `multipart/form-data`,
- przesył danych musi odbywać się w trybie `POST`.

Po określeniu formularza w odpowiedni sposób wszystkie związane z nim dane zostaną przesłane do serwera w specjalnym formacie. Serwlet obsługuje żądanie, przetwarza nadchodzące dane i wydobywa ze strumienia zawartość plików. *Lokalizacja docelowa* wskazuje miejsce zapisu pliku na serwerze. Naciśnięcie przycisku *Prześlij plik* na dole formularza spowoduje wysłanie danych do serwletu, który zapisze plik we wskazanej lokalizacji.

Formularz HTML znajduje się w pliku ***tut-install****/examples/web/fileupload/web/index.html* i ma następującą postać:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Przesyłanie plików</title>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   </head>
   <body>
      <form method="POST" action="upload" enctype="multipart/form-data" >
         Plik:
         <input type="file" name="file" id="file" /> <br/>
         Lokalizacja docelowa:
         <input type="text" value="/tmp" name="destination"/>
```

```
        <br/>
        <input type="submit" value="Prześlij plik" name="upload"
        ↪id="upload" />
      </form>
   </body>
</html>
```

Żądanie typu `POST` jest niezbędne, by przesłać do serwera dane w formacie wieloczęściowym, mogącym prawidłowo obsłużyć przesyłanie plików. Żądanie typu `GET` przesyła do serwera jedynie nagłówki i adres URL (być może z dodatkowymi parametrami). Żądanie typu `POST` przesyła dodatkowo treść komunikatu. Dzięki temu możliwy jest przesył do serwera danych w dowolnym formacie i dowolnej długości. Bardzo często pole nagłówka z żądania `POST` zawiera informację o typie MIME przesyłanej treści.

Przesyłając formularz, przeglądarka internatowa łączy wszystkie składowe w jeden strumień danych. Poszczególne części noszą nazwy pól wejściowych i są oddzielone specjalnymi separatorami nazywanymi `boundary`.

Oto wygląd danych przesyłanych z formularza przykładu *fileupload* po wybraniu pliku *teksty.txt* jako pliku do przesłania do serwera i wskazaniu folderu *tmp* jako lokalizacji docelowej:

```
POST /fileupload/upload HTTP/1.1
Host: localhost:8080
Content-Type: multipart/form-data;
boundary=---------------------------263081694432439
Content-Length: 441
-----------------------------263081694432439
Content-Disposition: form-data; name="file"; filename="teksty.txt"
Content-Type: text/plain
Dane z przykładowego pliku
-----------------------------263081694432439
Content-Disposition: form-data; name="destination"
/tmp
-----------------------------263081694432439
Content-Disposition: form-data; name="upload"
Prześlij plik
-----------------------------263081694432439--
```

Plik *FileUploadServlet.java* z kodem serwletu znajduje się w folderze ***tut-install****/examples/web/fileupload/src/java/fileupload/*. Serwlet rozpoczyna się fragmentem:

```
@WebServlet(name = "FileUploadServlet", urlPatterns = {"/upload"})
@MultipartConfig
public class FileUploadServlet extends HttpServlet {
   private final static Logger LOGGER =
           Logger.getLogger(FileUploadServlet.class.getCanonicalName());
```

Adnotacja `@WebServlet` wykorzystuje właściwość `urlPatterns`, by wskazać ścieżkę obsługiwaną przez serwlet.

Adnotacja `@MultipartConfig` wskazuje, iż serwlet oczekuje przesyłania żądań o typie MIME `multipart/form-data`.

Metoda `processRequest` pobiera z żądania docelową lokalizację oraz plik, a następnie używa metody `getFileName`, by pobrać nazwę przesyłanego pliku. W następnym kroku metoda tworzy obiekt `FileOutputStream` i kopiuje plik

do docelowej lokalizacji. Fragment obsługi błędów wyłapuje i obsługuje najczęstsze przyczyny nieodnalezienia pliku. Oto kod metod `processRequest` i `getFileName`:

```
protected void processRequest(
         HttpServletRequest request,
         HttpServletResponse response) throws ServletException,
         ↪IOException {
   response.setContentType("text/html;charset=UTF-8");

    // Tworzy komponent ścieżki, by określić miejsce zapisu pliku.
   final String path = request.getParameter("destination");
   final Part filePart = request.getPart("file");
   final String fileName = getFileName(filePart);

   OutputStream out = null;
   InputStream filecontent = null;
   final PrintWriter writer = response.getWriter();

   try {
      out = new FileOutputStream(
                new File(path + File.separator + fileName));
      filecontent = filePart.getInputStream();

      int read = 0;
      final byte[] bytes = new byte[1024];

      while ((read = filecontent.read(bytes)) != -1) {
         out.write(bytes, 0, read);
      }
      writer.println("Nowy plik " + fileName + " utworzony w " + path);
      LOGGER.log(Level.INFO, "Plik {0} został przesłany do {1}",
                new Object[] { fileName, path });
   } catch (FileNotFoundException fne) {
      writer.println("Albo nie wskazałeś pliku do przesłania, albo "
             + "próbujesz go zapisać w nieistniejącej lub niedostępnej "
             + "lokalizacji.");
      writer.println("<br/> BŁĄD: " + fne.getMessage());

      LOGGER.log(Level.SEVERE, "Problemy w trakcie przesyłu pliku.
      ↪Błąd: {0}",
                new Object[] { fne.getMessage() });
   } finally {
      if (out != null) {
         out.close();
      }
      if (filecontent != null) {
         filecontent.close();
      }
      if (writer != null) {
         writer.close();
      }
   }
}

private String getFileName(final Part part) {
   final String partHeader = part.getHeader("content-disposition");
   LOGGER.log(Level.INFO, "Nagłówek części = {0}", partHeader);
   for (String content : part.getHeader("content-
   ↪disposition").split(";")) {
      if (content.trim().startsWith("filename")) {
```

```
            return content.substring(content.indexOf('=') +
            ↪1).trim().replace("\"", "");
        }
    }
    return null;
}
```

Uruchomienie przykładu fileupload

Do uruchomienia przykładowej aplikacji *fileupload* można użyć albo środowiska NetBeans IDE, albo narzędzia Ant.

▼ Budowanie, pakowanie i wdrażanie przykładu fileupload w środowisku NetBeans IDE

1. Z menu *File* wybierz polecenie *Open Project*.
2. Po otwarciu okna dialogowego *Open Project* przejdź do folderu ***tut-install***/*examples/web*.
3. Zaznacz folder *fileupload*.
4. Kliknij przycisk *Open Project*.
5. Przejdź do zakładki *Projects*. Kliknij prawym przyciskiem myszy *fileupload* i z menu wybierz polecenie *Deploy*.

▼ Budowanie, pakowanie i wdrażanie przykładu fileupload przy użyciu narzędzia Ant

1. W oknie terminalu przejdź do poniższej lokalizacji:

   tut-install/*examples/web/fileupload/*

2. Wpisz polecenie:

   ```
   ant
   ```

3. Wpisz polecenie:

   ```
   ant deploy
   ```

▼ Uruchomienie przykładu fileupload

1. W przeglądarce internetowej wpisz następujący adres URL:

   ```
   http://localhost:8080/fileupload
   ```

   Pojawi się strona *Przesyłanie plików*.

2. Kliknij przycisk *Wybierz plik* w oknie przeglądarki internetowej.

3. Wybierz plik do przesłania i kliknij przycisk *Otwórz*.

   Nazwa wybranego pliku pojawi się w polu wejściowym *Plik*. Jeśli plik nie zostanie wybrany, serwer zgłosi wyjątek.

4. W polu *Lokalizacja docelowa* wpisz nazwę folderu.

   Wskazywany folder musi istnieć i serwer musi mieć prawo zapisania w nim plików. Jeśli folder nie zostanie wskazany lub podany folder nie będzie istniał (albo serwer nie będzie miał odpowiednich uprawnień), serwer zgłosi wyjątek.

5. Kliknij przycisk *Prześlij plik*, by przesłać plik i zapisać go w folderze wskazanym w polu *Lokalizacja docelowa*.

   Pojawi się komunikat informujący o zapisaniu pliku we wskazanym folderze.

6. Przejdź do folderu podanego w polu *Lokalizacja docelowa* i sprawdź, czy plik rzeczywiście się tam znajduje.

Skorowidz

A

Abstract Window Toolkit, *Patrz:* AWT
Adapter zasobów, *Patrz:* zasób adapter
adnotacja, 21, 148, 333
  @AroundInvoke, 441
  @Chosen, 271
  @Consumes, 202
  @DefaultValue, 197
  @Disposes, 255
  @FacesConverter, 164
  @FacesRenderer, 169
  @FacesValidator, 164
  @FormParam, 197
  @Interceptor, 260
  @Interceptors, 442, 444
  @ManagedBean, 152
  @ManagedBean, 148
  @MessageDriven, 366
  @MultipartConfig, 177, 180
  @Named, 152
  @NamedQuery, 298
  @Observes, 257
  @PathParam, 196
  @PostCreate, 443
  @Produces, 202, 271
  @RequestScoped, 271
  @Resource, 364
  @Specializes, 253
  @WebServlet, 180
  javax.annotation.Resource, 256
  javax.decorator.Decorator, 261
  javax.ejb.Asynchronous, 242
  javax.enterprise.inject.Alternative, 252
  javax.faces.bean.ManagedBean, 148
  javax.interceptor.Interceptors, 441
  javax.persistence.Cacheable, 302
  javax.persistence.Version, 296
  javax.ws.rs.core.Context, 198
  JAX-RS, 195
  metadanych interceptora, 440
adres
  IP, 309
  URI, 195, 325
Ajax, 41, 64, 81, 82
alternatywa, 251, 252, 253, 265
Ant, 101, 176, 232, 247, 269, 272, 278, 284, 287, 375, 379, 385, 389, 395, 399, 401, 406, 411, 417, 423, 432, 447, 452
Apache Ant, 51, 54
API, 46, 47
  Java Naming and Directory Interface, *Patrz:* JNDI
  Javy, 22
  Javy EE 6, 37
  JavaMail, 45
  Swing, 26
  JMS, *Patrz:* JMS
aplet, 25, 26, 27, 31, 33
aplikacja, 57
  CDI, 257, 259, 261
  cykl życia, 64
  EIS, 329
  enterprise, 21, 23, 31
  JavaServer Faces, 147, 171, 173, 175
  JEE, 24, 33
  JMS, 354, 397
  kliencka, 25, 26, 27, 31, 35, 47
  klient, 21, 227, 340
  konfiguracja, 172
  lokalizacja, 185, 186
  osadzona, 236
  paczka, 176
  publikuj-subskrybuj, 342
  punkt-punkt, 342
  umiędzynarodowienie, 185, 186, 436
  uruchamianie, 182
  webowa, 450
  wielowarstwowa, 23, 24

application assemblers, *Patrz:* budowniczy aplikacji
architektura
  Java EE Connector, 35
  JMS, *Patrz:* JMS
  komponentów JavaBeans, 27
  konektora Javy EE, 340
  REST, 195
  usług implementowanych, 23
atrybut
  attribute, 164
  binding, 134, 145, 149
  converter, 133, 134
  converterId, 134
  converterMessage, 163
  default, 98
  disabled, 84
  eager, 149
  event, 84, 86
  execute, 84, 86, 87, 89, 90
  fileSizeThreshold, 177
  forClass, 133
  immediate, 85, 87
  listener, 85, 87
  location, 177
  MaxFileSize, 178
  maxRequestSize, 178
  method-signature, 98
  name, 98
  onclick, 110, 130
  onerror, 85, 88
  onevent, 85, 87
  onmouseout, 110, 130
  onmouseover, 110, 130
  przyjmowanie wyrażeń, 120
  render, 85, 89, 90
  rendered, 144
  required, 98
  requiredMessage, 163
  type, 98
  validatorId, 139
  validatorMessage, 163
  value, 141
automatyczne zatwierdzanie, 43

B

Bean Validation, 23, 44, 135, 435
bezpieczeństwo, 25, 46, 235, 309, 311, 313, 314, 316, 318, 319, 321, 325, 328, 329, 331, 334
  konfiguracja, 333
biblioteka
  Ajax, 83
  JavaServerFaces, 83
  javax.faces, 91
  znaczników, 128
    JavaServer Pages, 42
blokada
  Java Persistence API, 295
  metod HTTP, 326, 327
  odczytu, 295
  optymistyczna, 295, 296, 297
  pesymistyczna, 296, 297, 298, 299
  tryby, 297, 298
  zapisu, 295
budowniczy aplikacji, 35, 37
business-tier, *Patrz:* warstwa logiki biznesowej

C

CDI, 23, 44, 152, 195, 200, 251, 257, 259, 261, 265, 441, 455
cel, *Patrz:* komunikat cel
certyfikat cyfrowy, 309, 310, 314, 317
  tworzenie, 310, 311, 313
client-tier, *Patrz:* warstwa klienta
Contexts and Dependency Injection for the Java, *Patrz:* CDI
CRUD, 210
Curl, 222

D

debugger, 59
debugowanie, 59, 60
dekodowanie, 107, 123
dekorator, 261, 285
Dependency Injection for Java, *Patrz:* DI
deployment descriptor, *Patrz:* deskryptor wdrożenia
deserializer, 49
deskryptor
  biblioteki znaczników, *Patrz:* TLD
  wdrożenia, 21, 33, 34, 316, 321, 439
    EJB, 35
    JAVY EE, 34
    modułu EJB, 35
    środowiska uruchomieniowego, 34
    zasady bezpieczeństwa, 333, 334
  wdrożenia aplikacji, 37, 237
  wdrożenia aplikacji klienckiej, 35

deskryptorz wdrożenia, 303
DI, 23, 44
Document Object Model, *Patrz:* DOM
DOM, 82, 89, 92
domena
  bezpieczeństwa, 321
  publikacja-subskrypcja, 341, 342
  reguł bezpieczeństwa, 321
  typu punkt-punkt, 341
dostawca
  encji, 205
  JMS, 338, 341
  komponentu aplikacji, 36, 37
  narzędzi, 36, 45
  oprogramowania Java EE, 36
  trwałości, 295, 301, 302
dziedzina uwierzytelniania JDBC, 321

E

EIS, 24, 329, *Patrz też:* warstwa danych
EJB, 25, 28, 31, 33, 39, 47, 200, 235
EL, 41
element
  alternative, 252
  converter-class, 165
  from-action, 167
  from-outcome, 167
  http-method-omission, 327
  list-entries, 154, 156, 158
  locale-config, 187
  lockMode, 298
  managed-bean, 152, 153, 154
  managed-bean-class, 154
  managed-bean-name, 154
  managed-bean-scope, 154, 155
  managed-property, 154, 155, 157
  map-entries, 154, 156, 157
  navigation-case, 167
  navigation-rule, 166, 167
  navigation-rules, 168
  null-value, 156
  renderer-class, 169
  renderer-type, 169
  render-kit-id, 169
  resource-bundle, 161, 163, 187
  shared-cache-mode, 303
  to-view-id, 167
  validator-class, 164
  validator-id, 164, 165
  value, 156
e-mail, 449
encja
  dostawca, *Patrz:* dostawca encji
  dostęp do danych współbieżny, 295
  menedżera, 273
Enterprise Archive, *Patrz:* EAR
Enterprise Information System, *Patrz:* EIS
Enterprise JavaBeans, *Patrz:* EJB
Enterprise Resource Planning, *Patrz:* ERP
etykieta, 83
  lokalizacja, 186
  umiędzynarodowienie, 186

F

Fabryka połączeń, 341, 344, 345, 346, 360, 375, 428, 449
facelet, 63, 91, 168, 266, 276, 280, 285, 450, 455, 457
Facelets, 41
FacesMessage, 162
format
  EAR, 33, 35
  JAR, 33, 236
  WAR, 33, 35, 171
formularz, 197, 276
  logowania, 318
Full Profile, *Patrz:* profil pełen

G

GUI, 26

H

hasło, 99, 311, 314, 321
HTML, 26, 128, 450,
HTTP, 32, 64, 196, 199, 326, 327
HTTPS, 310

I

integrator systemów, 45
interceptor, 259, 261, 279, 283, 285, 439
  cykl życia, 440
  powiązania, 260, 285, 286
  upłynięcia czasu, 444, 445
  zdarzenia cyklu życia, 443
interfejs, 285
  CallbackHandler, 328
  ConnectionFactory, QueueConnectionFactory, 345

interfejs
dostawcy usług, *Patrz:* SPI
java.security.Principal, 256
java.util.concurrent.Future, 241
javax.faces.component.ActionSource, 125
javax.faces.convert.Converter, 131
javax.faces.event.ActionListener, 125
javax.faces.event.ValueChangeListener, 125, 126
javax.jms.MessageListener, 229, 366
javax.persistence.Cache, 305
javax.servlet.http.Part, 178
javax.transaction.UserTransaction, 256, 368
javax.validation.Validator, 256
javax.validation.ValidatorFactory, 256
javax.ws.rs.core.HttpHeaders, 198
javax.ws.rs.core.UriInfo, 198
JNDI, 30
lokalny, 229
MessageListener, 229
nasłuchiwania komunikatów, 229
StateHelper, 121
TopicConnectionFactory, 345
użytkownika, 41
graficzny, *Patrz:* GUI
zdalny, 229
ISO 8859, 190

J

JAAS, 49, 328
JACC, 45
JAF, 47
JASPIC, 23, 46
Java API for XML Processing, *Patrz:* JAXP
Java Architecture for XML Binding, *Patrz:* JAXB
Java Archive, *Patrz:* format JAR
Java Authentication And Authorization Service, *Patrz:* JAAS
Java Authentication Service Provider Interface for Containers, *Patrz:* JASPIC, *Patrz:* JASPIC
Java Authorization Contract for Containers, *Patrz:* JACC
Java Community Process, *Patrz:* JCP
Java Database Connectivity, *Patrz:* JDBC API
Java EE 6 Software Development Kit, 51
Java EE Connector Architecture, *Patrz:* JCA
Java Message Service, *Patrz:* JMS, *Patrz:* JMS
Java Naming and Directory Interface, *Patrz:* interfejs JNDI
Java Persistence API, *Patrz:* JPA
Java Platform Standard Edition, *Patrz:* JDK
Java Servlet, 25, 40
Java Specification Request, *Patrz:* JSR
Java Transaction API, *Patrz:* JTA
JavaBeans, 27
JavaBeans Activation Framework, *Patrz:* JAF
JavaServer Faces, 23, 25, 27, 41, 63, 81, 103, 109, 171, 173, 449, 455
cykl życia, 64
cykl życia elementów, 63
konfiguracja, 147
obsługa błędów, 88
JavaServer Pages, *Patrz:* JSP, *Patrz:* JSP
JavaServer Pages Standard Tag Library, *Patrz:* JSTL
JAXB, 48, 204, 209
JAXP, 48
JAX-RS, 43, 195, 200, 209
JCA, 45
JCP, 21
JDBC, 255, 321
JDBC API, 46
JDK, 51
język
domyślny, 187
wyrażeń Expression Language, *Patrz:* EL
zapytań, 43
znaczników, 26
JMS, 227, 231, 337, 338, 340, 363, 370, 371
połączenie, 347
sesja, 347
wyjątki, 353
JNDI, 47, 238
JPA, 42
JSON, 82, 209
JSP, 25, 27, 42
JSR, 21
JSTL, 42
JTA, 43

K

keytool, 310, 311
klasa, 148
abstrakcyjna, 228, 262
celów, 439
com.sun.appserv.security.ProgrammaticLogin, 329
dekoratora, 261
dynamiczna, 291

encji, 274
interceptora, 260, 283
interceptorowa, 439, 440, 441
interceptorów, 439
javax.ejb.embedded.EJBContainer, 236
javax.faces.component.UIComponent, 103
javax.faces.component.UIComponentBase, 114, 117
javax.faces.context.ResponseWriter, 118
javax.faces.event.ActionEvent, 125
javax.faces.render.Renderer, 103, 104
javax.ws.rs.core.Variant, 203
komponentu, 148, *Patrz:* komponent klasa
konwertera, 148
końcowa, 228
metamodelu, 291
nasłuchiwania dla zdarzenia akcji, 111
nasłuchiwania zdarzeń, 280
nasłuchująca, 125, 145
OutputStream, 191
PrintWriter, 191
publiczna, 228
renderera, 148
statyczna, 291
walidatora, 148
zasobów, 198
zdarzenia, 279
ziarna, 279
ziarna sterowanego komunikatami, 228
klasa nasłuchująca, 104
klient
aplikacji, *Patrz:* aplikacja klient
cienki, *Patrz:* klient przeglądarkowy
JMS, 341
na wielu systemach, 391
przeglądarkowy, 25, 26, 27, 29
klucz
magazyn, 311, 313, 317
prywatny, 310
publiczny, 310, 314
kod Cezara, 266
kodowanie, 107, 123
kolejka, 227, 228, 342
przeglądarka, *Patrz:* przeglądarka kolejek
komponent, 105, 169
aplikacji dostawca, *Patrz*: dostawca komponentu aplikacji
aplikacji klienckiej, 31
drzewo, 90
EJB, 39, 340, 363
Enterprise JavaBeans, 25
grupowanie, 90
instancja, 140, 144
interfejsu użytkownika, 103
Java Servlet, 25
JavaBeans, 27
JavaServer Faces, 23, 25
JavaServer Pages, 25
Javy EE, 25
klasa, 114
mapy obrazu, 108, 109, 113, 455
modułów, 420
o silnym typowaniu, 251
obsługa zdarzeń, 127
rejestracja, 170
rodzina, 117
samouczka Javy EE 6, 51, 53
sesyjny, 40
singleton, 40
sterowany komunikatami, 40, 227
tworzenie, 103, 104, 105, 106, 107, 108, 113, 114, 129, 413
wartość, 140, 141, 142
webowy, 25, 27, 28, 47, 340, 363
widoku, 64
zapis stanu, 121, 175
zarządzany
adnotacją, 41
wpisem, 41
złożony, 97, 99
komunikat, 160, 162, 227, 228, 339, 341
asynchroniczny, 227, 340, 343, 344, 381, 382, 385
błędu, 162
cel, 337, 344, 346, 349, 375
tworzenie, 358
tymczasowy, 358
domyślny, 358, 436
dziedzina, 341, 342
JMS, 351
konsument, 347, 349
nagłówek, 351
producent, 347, 348
selektor, 350
typ, 352
właściwości, 351
kolejka, 342, 346, 358, 375, 386
konsument, 344, 358, 366
konsumowanie, 419
lokalizacja, 186
łączenie, 412

komunikat
nadawca, 337, 342
nasłuchiwanie, 349, 366
niezawodność, 355
odbiorca, 337, 342, 343, 346, 381
PERSISTENT, 354
połączenie, 344
potwierdzanie, 355, 356, 366, 397
priorytet, 357
producent, 344
przekazywanie, 337
publikuj-subskrybuj, 342, 343
punkt-punkt, 342, 343
sesja, 344
synchroniczny, 340, 344, 364, 372, 375, 376, 379
temat, 342, 346, 358, 375
trwałość, 356
tworzenie, 435
umiędzynarodowienie, 186
walidacyjny, 436
wybór języka, 188
wygasanie, 358
konsola administracyjna, 56
konstruktor publiczny, 229
kontener, 30, 330
apletów, 31
aplikacji klienckich, 31
EJB, 21, 241
wyłączanie, 238
Enterprise JavaBeans, 31
komponentów EJB, 235
osadzony, 235
kontrakt autoryzacji, 45
webowy, 21, 31, 177
kontrolka, 110
konwerter, 104, 145, 162
daty i czasu, 189
tworzenie, 130, 131, 133
kryptografia, 310

L

LDAP, 321
lista, 158, 160
logika
aplikacji, 23
biznesowa, 21, 23, 24, 28
logowanie, 328
oparte na formularzu, 318
programowe, 329
zarządzane przez komponent, 329, 330
zarządzane przez kontener, 329, 330
lokalizacja, 185, 186
lokalizator podzasobu, 198, 199

M

Managed Bean, 43
Managed Beans, 22
mapa obrazu, 108, 109, 113, 455
menu rozwijane, 131
Message-Oriented Middleware, *Patrz:* MOM
metadane odwzorowania obiektowo-relacyjnego, 43
metamodel, 291
metoda, 166
@PostConstruct, 366
asynchroniczna, 241, 242, 243, 244
build, 203
Cache.contains, 305
Cache.evict, 306
Cache.evictAll, 306
CLIENT-CERT, 316
commit, 361
Connection.createSession, 369
createDurableSubscriber, 360
createEJBContainer, 237
createSession, 355, 369
DateFormat.getDateInstance int, locale, 189
decode, 119
encodeBegin, 117, 118
encodeChildren, 117
encodeEnd, 117, 118
EntityManager.createQuery, 293
EntityManager.find, 298
EntityManager.flush, 295
EntityManager.lock, 298
EntityManager.refresh, 298
EntityManager.setProperty, 304
finalize, 229
get, 27
getAsObject, 131, 132
getAsString, 132, 133
getLocale, 187
getPart, 178
getRendererType, 124
getResultList, 293
getSingleResult, 293
interceptora, 439

interceptorowa, 439
cyklu życia, 444
upłynięcia czasu, 444, 445
InvocationContext.proceed, 442
javax.enterprise.event.Event.fire, 258
javax.faces.event.AjaxBehaviorListener.processAjaxBehavior, 87
jsf.ajax.request, 83, 86, 89, 92
kodująca, 118
nawigacji po ścieżce, 292
NumberFormat.getXXXInstance, 189
obserwatora, 257
transakcyjna, 258
warunkowa, 258
ograniczania wyników, 292
onMessage, 229, 230, 350, 365, 366
podzasobu, 198, 199
processAction, 126
processValueChange, 125
produkująca, 253, 254, 270, 271
przechwytująca, 441
Query.setLockMode, 298
request.getParts, 178
restoreState(FacesContext, Object), 121, 122
rollback, 361
saveState(FacesContext), 121, 122
selectVariant, 203
Session.createBrowser, 353
Session.createConsumer, 359
Session.createDurableSubscriber, 359
Session.createTemporaryQueue, 358
Session.createTemporaryTopic, 358
Session.recover, 356
set, 27
setMessageListener, 349, 366
setRollbackOnly, 229
startElement, 118
synchroniczna, 241
toString, 279
TypedQuery.setLockMode, 298
UserTransaction.begin, 369
UserTransaction.commit, 369
UserTransaction.rollback, 369
usuwająca, 255
uwierzytelniania, 316
validate, 136
visitTree, 90
write(String filename), 179
wywołania zwrotnego, 366
middle tier, *Patrz:* warstwa logiki
model obiektowy dokumentu, *Patrz:* DOM
moduł
adaptera zasobów, 35
aplikacji klienckiej, 35
EJB, 237
Javy EE, 35
webowy, 35
MOM, 339

N

namespace, *Patrz:* przestrzeń nazw
narzędzie
Ant, *Patrz:* Ant
keytool, *Patrz:* keytool
negocjacja treści, 202, 203
NetBeans IDE, 51, 52, 53, 101, 167, 173, 176, 207, 229, 231, 246, 268, 272, 278, 284, 286, 375, 383, 388, 394, 398, 400, 404, 410, 416, 422, 429, 430, 446, 451

O

obiekt
administracyjny, 341, 344, 375
administrowany
dla wielu systemów, 392
ConnectionFactory, 346
DataSource, 255
dostarczony przez system, 47
EntityManager, 255
HttpServletRequest, 177
Iterable, 178
java.util.Locale, 185
javax.persistence.TypedQuery, 293
List, 158, 160
Map, 160
MessageListener, 350
niejawny, 142, 143, 157
POJO, 22
QueueBrowser, 353
wstrzykiwanie, *Patrz:* wstrzykiwanie obiektu
zdarzenia, 257
zdefiniowany przez użytkownika, 47
obsługa błędów, 88
odpowiedź Ajax, 89
odwzorowanie, 160
ograniczenie, 437
tworzenie, 435
wbudowane, 435
oprogramowanie, 51

P

Paczka zasobów, 187, 188
pamięć cache
  drugiego poziomu, 301, 302, 303, 305
  odczyt nieaktualny, 301
  pobieranie, 304
  zapis, 304
Plain Old Java Object, *Patrz:* obiekt POJO
plik
  EAR, 33, 35
  JAR, 33, 236
  keystore.jks, 311, 313
  konfiguracyjny, 285
  przesyłanie do serwera, 177
  TLD, 128
  WAR, 33, 35, 171
  zasobów JavaScript, 91
  zasobu konfiguracji aplikacji, 148, 149, 174
    kolejność, 151
poczta elektroniczna, 338, 449
podpis cyfrowy, 310
Point-To-Point, *Patrz:* PTP
POJO, 43
pole, 27
  produkujące, 254, 255, 273
pole tekstowe, 83
połączenie, *Patrz:* JMS połączenie
  SSL, 309
profil
  pełen, 22
  webowy, 22
protokół
  HTTPS, 310
  SOAP, *Patrz:* SOAP
przeglądarka
  internetowa, 26, 105
  klient, *Patrz:* klient przeglądarkowy
  kolejek, 347, 353
  logów, 59
przestrzeń nazw, 48, 129
przycisk, 83, 86
PTP, 342

R

reguła
  bezpieczeństwa, 321
  nawigacyjna, 165, 166, 167
    niejawna, 168
  serializacji, 190
Remote Method Invocation, *Patrz:* RMI
renderer, 105, 122
  HTML, 128
  rejestracja, 168
  tworzenie, 104, 107, 108, 114
  typ, 124
rendering, 106
  delegowanie, 123
renderowanie, 107, 117, 133, 169
  odpowiedzi, 90
REST, 43, 195, 204
RMI, 338
rozgłaszanie, 87
runtime deployment descriptor, *Patrz:* deskryptor wdrożenia środowiska uruchomieniowego

S, Ś

SAAJ, 48
Secure Sockets Layer, *Patrz:* SSL
serializer, 49
Service Provider Interface, *Patrz:* SPI
serwer
  GlassFish, 49, 52, 54, 55, 56, 227, 309, 345
    narzędzia, 49, 50
  Javy EE, 31
  JEE, 24
  logi, 59
  lokalny, 425
  pocztowy, 449
serwlet, 23, 27, 31, 33, 40, 172, 179
  fragment kodu, *Patrz:* snippet
sesja, *Patrz:* JMS sesja
Simple Object Access Protocol, *Patrz:* SOAP
skalowalność, 23
snippet, 42
SOAP, 32
SOAP with Attachments API for Java, *Patrz:* SAAJ
specjalizacja, 252, 253
SPI, 46
SSL, 309, 314, 316
stereotyp, 262
strona
  faceletu, *Patrz:* facelet
  HTML, 450
  internetowa, 31, 33
    dynamiczna, 26
  statyczna, 81
  wykorzystująca komponent, 100

subskrypcja trwała, 343, 359, 360, 400
szyfr przesuwający, 266
środowisko
  nazewnictwa, 47
  produkcyjne, 309
  programistyczne, 309

T

tablica, 158
Thawte, 310
TLD, 128
transakcja, 354, 361
  lokalna, 359, 361, 362, 365, 402
  rozproszona, 368
  zarządzana przez kontener, 365, 368
  zarządzana przez ziarno, 365, 368, 369

U

umiędzynarodowienie, 185, 186, 436
Unicode, 190
Uniform Resource Identifier, *Patrz:* URI
URI, 195, 198
US-ASCII, 190
usługa
  sieciowa, 31, 204
    opis, *Patrz:* WSDL
    REST, 43, 195, 210
    tester, 221
  zegarowa upłynięcia czasu, 444
UTF-8, 190
uwierzytelnianie, 321, 328, 333
  bazujące na formularzu, 318, 328, 450
  klienta, 314, 328
  wzajemne, 314, 316, 317
użytkownik
  autoryzacja, 49
  autoryzowany, 30
  dodawanie, 321
  edycja, 321
  nazwa, 99
  tożsamość, 313
  uwierzytelnianie, 49, 318, 319, 321

V

vcomponent-family, 169
Verisign, 310, 317

W

walidacja, 135, 150, 437
walidator, 104, 145, 162
  domyślny, 163
  instancji ziaren, 256
  rejestracja, 164
  tworzenie, 135, 138, 435
warstwa
  danych, 24, 29
  klienta, 24
  logiki, 23
  logiki biznesowej, 24, 28
  webowa, 24, 27, 28, 61
wdrożeniowiec, 37
Web Archive, *Patrz:* format WAR
Web Profile, *Patrz:* profil webowy
Web Services Description Language, *Patrz:* WSDL
web-tier, *Patrz:* warstwa webowa
widok, 64
WSDL, 33
wstrzykiwanie
  kontekstów i zależności, 44
  obiektu, 253
  punkt delegowany, 261
  zależności, 21, 44, 235, 251, 265
  zasobu, 229, 255, 256
    celu, 346
    fabryki połączeń, 346
wypychanie danych, 295
wyrażenie, 120
  dowiązania wartości, 141
  metody, 292
  warunkowe, 292

X

XML, 26, 32, 82, 204
  schemat, 48, 207, 217
  WSDL, *Patrz:* WSDL

Z

zapytanie, 197
  metamodelowe, 291, 292
  silnie typowane, 291
  słabo typowane, 291

zapytanie
tekstowe, 291, 292, 293
dynamiczne, 291
statyczne, 291
tworzenie, 292
w trybie wypychania danych, 295
zasób
adapter, 35, 45, 331, 340, 449
MessageDrivenContext, 229
publikacja, 204
reprezentacja, 205
tworzenie, 375
usuwanie, 376
wstrzykiwanie, 229
zdarzenie, 86, 105, 257, 279
akcji, 125
DOM, 92
nasłuchiwanie, 280
wywołania zwrotnego cyklu życia, 443
zgłaszanie, 258
zmiany wartości, 125
zestaw znaków, *Patrz:* znak zestaw
ziarno, 140, 145, 251, 254
adnotacja, 148, *Patrz też:* adnotacja
alternatywa, *Patrz:* alternatywa
bezstanowe, 319
drzewo, 159
EJB, *Patrz:* EJB
encji, 229
o zasięgu aplikacji, 149
sesyjne, 229, 237, 241, 273, 319, 364, 365, 366, 367, 407, 408
bezstanowe, 236, 457
singletonowe, 457
specjalizacja, *Patrz:* specjalizacja
stereotyp, 262
sterowane komunikatami, 227, 228, 340, 350, 364, 365, 366, 367, 368, 407, 409, 412, 419, 426, 427, 449, 450
wersja pełna, 252
wersja uproszczona, 252
zarządzane, 98, 100, 141, 273, 276, 280, 285, 319, 450
konfiguracja, 152
właściwości, 159
znacznik
area, 110
composite:attribute, 97, 98
converter, 165
default-locale, 161
f:actionListener, 111
f:ajax, 83, 84, 85, 86, 87, 90, 94
f:converter, 134
f:loadBundle, 163
f:validateBean, 99
f:validateRegex, 99
f:validateRequired, 99
f:validator, 138, 139
fmt:setLocale, 187
h:dataTable, 455
h:outputScript, 91
input, 110
javax.faces.view.facelets.Tag, 104
komponentu, 140, 149
message, 162, 188
messages, 162, 188
procedura obsługi, 108
renderer, 169
supported-locale, 161
tworzenie, 104, 108, 128, 138
walidacja, 128
walidatorów, 98
XML, 32
znacznika composite:attribute, 97, 98
znak
kodowanie, 190
zestaw, 190

Ż

żądanie, 87, 105, 179
Ajax, 86, 89, 90
GET, 180, 196
warunkowe, 201
HTTP, 64, 196, 199
warunkowe, 201
inicjujące, 121
JavaServer Faces, 90
nagłówek, 195
początkowe, 64
POST, 180
PUT warunkowe, 201
URI, 325
zwrotne, 64

Bezcenne źródło informacji!

Java Enterprise Edition 6 (Java EE6) to nowa jakość w tworzeniu zaawansowanych aplikacji korporacyjnych. Wyjątkowo łatwa konfiguracja, lekki profil internetowy oraz zdecydowanie prostsze pakowanie aplikacji to tylko niektóre z zalet tej platformy. Tempo, w jakim zdobywa ona popularność, zaskakuje niejednego specjalistę. Obecnie jest najczęściej wybierana jako fundament nowych projektów. To znak, że doskonale sprawdziły się założenia przyjęte przez jej twórców. Przyszłość projektów to właśnie platforma Java EE6!

Drugi tom tego rewelacyjnego podręcznika porusza zaawansowane tematy związane z platformą Java EE6. W trakcie lektury poznasz zagadnienia związane z JSF, JAX-RS oraz JAXB. Dowiesz się, jak tworzyć niezawodne ziarna sterowane komunikatami, wstrzykiwać zależności oraz korzystać z elementów programowania aspektowego. Najwięcej emocji wzbudza rozdział poświęcony Java Persistance API. Mapowanie obiektowo-relacyjne to wciąż gorący temat, a jego poprawne wykorzystanie bardzo korzystnie wpłynie na Twoją aplikację. Błąd w tym miejscu może Cię kosztować sporo nerwów, dlatego powinieneś zaznajomić się z mechanizmami bezpieczeństwa w Java EE6 oraz z kontrolą poprawności danych. Książka ta jest kompletnym źródłem informacji o platformie Java EE6. Musisz ją przeczytać!

Dowiedz się:

- jak wstrzykiwać zależności
- jak wykorzystać potencjał JPA
- do czego stosować ziarna sterowane komunikatami
- jak stworzyć bezpieczną aplikację

Addison-Wesley
Pearson Education
ORACLE
cena: 89,00 zł
ISBN 978-83-246-7393-3
9 788324 673933

PEARSON