



Technologia i rozwiązania

# Java EE 6

## Tworzenie aplikacji w NetBeans 7

Obowiązkowa wiedza każdego programisty!

Helion



David R. Heffelfinger

[PACKT  
PUBLISHING]

Tytuł oryginału: Java EE 6 Development with NetBeans 7

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-246-8936-1

Copyright © 2011 Packt Publishing

First published in the English language under the title 'Java EE 6 Development with NetBeans'.

© Helion 2014.

All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

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

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

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

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

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:  
<ftp://ftp.helion.pl/przyklady/jave6n.zip>

Drogi Czytelniku!  
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres  
<http://helion.pl/user/opinie/jave6n>  
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

<b>O autorze</b>	<b>7</b>
<b>O recenzentach</b>	<b>9</b>
<b>Przedmowa</b>	<b>11</b>
<b>Rozdział 1. Wprowadzenie do środowiska NetBeans</b>	<b>15</b>
Wprowadzenie	15
Pobieranie środowiska NetBeans	16
Instalowanie środowiska NetBeans	19
Pierwsze uruchomienie środowiska NetBeans	26
Konfigurowanie środowiska NetBeans pod kątem programowania w Javie EE	27
Instalowanie pierwszej aplikacji	35
Wskazówki dotyczące wydajnego programowania w środowisku NetBeans	38
Podsumowanie	49
<b>Rozdział 2. Tworzenie aplikacji sieciowych z wykorzystaniem serwetów i technologii JSP</b>	<b>51</b>
Tworzenie pierwszej aplikacji	52
Programowanie serwetów	72
Zabezpieczanie aplikacji sieciowych	81
Fragmenty JSP	93
Podsumowanie	96
<b>Rozdział 3. Wzbogacanie stron JSP za pomocą biblioteki JSTL i niestandardowych znaczników</b>	<b>97</b>
Podstawowe znaczniki JSTL	98
SQL-owe znaczniki JSTL	107
Modyfikowanie danych w bazie za pomocą znacznika <code>&lt;sql:update&gt;</code>	114
Ostatnie uwagi na temat biblioteki JSTL	122
Niestandardowe znaczniki JSP	122
Podsumowanie	129

<b>Rozdział 4. Tworzenie aplikacji sieciowych z wykorzystaniem platformy JavaServer Faces 2.0</b>	<b>131</b>
Wprowadzenie do platformy JSF	132
Tworzenie pierwszej aplikacji JSF	132
Tworzenie szablonów faceletów	152
Komponenty złożone	159
Podsumowanie	164
<b>Rozdział 5. Tworzenie eleganckich aplikacji sieciowych z wykorzystaniem biblioteki PrimeFaces</b>	<b>165</b>
Pierwszy projekt utworzony z wykorzystaniem biblioteki PrimeFaces	165
Stosowanie komponentów PrimeFaces w aplikacjach JSF	169
Widoki z zakładkami	173
Interfejsy oparte na kreatorze	178
Dodatkowe informacje	183
Podsumowanie	183
<b>Rozdział 6. Interakcja z bazami danych za pomocą interfejsu Java Persistence API</b>	<b>185</b>
Tworzenie pierwszej encji JPA	186
Automatyczne generowanie encji JPA	200
Relacje między encjami	209
Generowanie aplikacji JSP na podstawie encji JPA	215
Podsumowanie	221
<b>Rozdział 7. Implementowanie warstwy biznesowej za pomocą ziaren sesyjnych</b>	<b>223</b>
Wprowadzenie do ziaren sesyjnych	224
Tworzenie ziaren sesyjnych w środowisku NetBeans	224
Dostęp do ziarna z poziomu klienta	233
Zarządzanie transakcjami w ziarnach sesyjnych	238
Programowanie aspektowe z wykorzystaniem interceptorów	239
Usługi zegara w ziarnach EJB	242
Generowanie ziaren sesyjnych na podstawie encji JPA	244
Podsumowanie	248
<b>Rozdział 8. Interfejs API CDI</b>	<b>249</b>
Wprowadzenie do CDI	249
Kwalifikatory	256
Stereotypy	260
Typy do wiązania interceptorów	263
Podsumowanie	267

<b>Rozdział 9. Przesyłanie komunikatów za pomocą usług JMS i ziaren sterowanych komunikatami</b>	<b>269</b>
Wprowadzenie do interfejsu JMS	270
Tworzenie projektu i zasobów JMS	271
Przetwarzanie komunikatów JMS za pomocą ziaren sterowanych komunikatami	280
Podsumowanie	283
<b>Rozdział 10. Usługi sieciowe SOAP oparte na interfejsie JAX-WS</b>	<b>285</b>
Wprowadzenie do usług sieciowych	285
Tworzenie prostej usługi sieciowej	286
Udostępnianie ziaren EJB jako usług sieciowych	298
Podsumowanie	305
<b>Rozdział 11. Usługi sieciowe RESTful oparte na interfejsie JAX-RS</b>	<b>307</b>
Generowanie usług sieciowych typu RESTful na podstawie istniejącej bazy danych	308
Testowanie usług sieciowych typu RESTful	314
Tworzenie klienta usług sieciowych typu RESTful	319
Podsumowanie	325
<b>Dodatek A. Debugowanie aplikacji dla firm za pomocą debugera środowiska NetBeans</b>	<b>327</b>
Debugowanie aplikacji dla firm	327
Podsumowanie	333
<b>Dodatek B. Wykrywanie problemów z wydajnością za pomocą profilera środowiska NetBeans</b>	<b>335</b>
Profilowanie aplikacji	335
Podsumowanie	339
<b>Skorowidz</b>	<b>341</b>



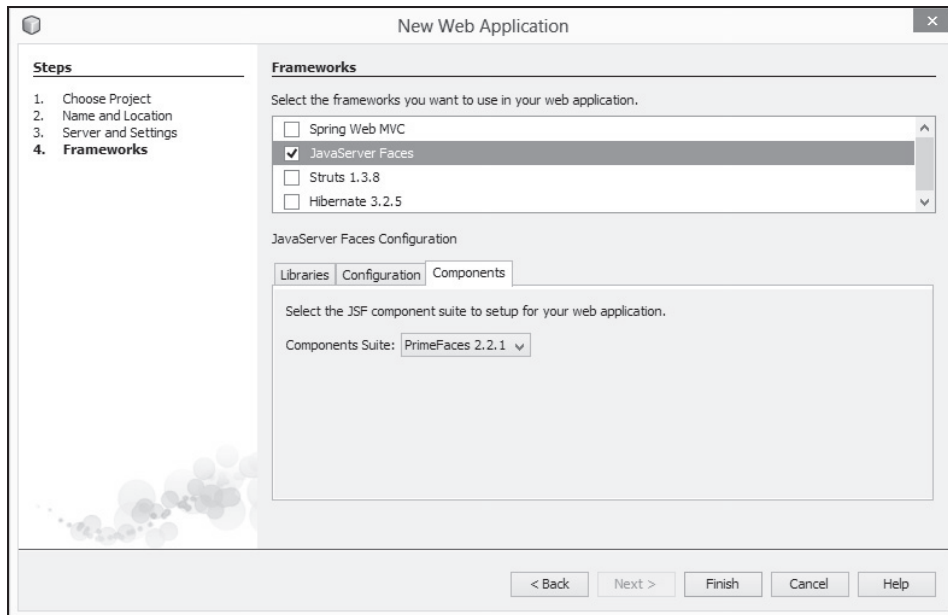
# Tworzenie eleganckich aplikacji sieciowych z wykorzystaniem biblioteki PrimeFaces

Jedną z zalet platformy JSF jest to, że umożliwia bardzo łatwe tworzenie niestandardowych komponentów. Dlatego powstało kilka bibliotek komponentów o otwartym dostępie do kodu źródłowego. Jedną z nich jest biblioteka PrimeFaces. Umożliwia ona łatwe tworzenie eleganckich aplikacji sieciowych. W wersji 7.0 środowiska NetBeans dostępna jest wbudowana biblioteka PrimeFaces.

---

## Pierwszy projekt utworzony z wykorzystaniem biblioteki PrimeFaces

Aby zastosować bibliotekę PrimeFaces w projekcie, trzeba w standardowy sposób utworzyć aplikację sieciową w Javie. Gdy wybierzesz platformę JavaServer Faces, kliknij zakładkę *Components*, a następnie wybierz pozycję *PrimeFaces 2.2.1* jako pakiet komponentów (rysunek 5.1).



Rysunek 5.1. Tworzenie aplikacji z biblioteką PrimeFaces

Po utworzeniu projektu środowisko NetBeans doda potrzebne biblioteki. Na stronach JSF dostępny wtedy będzie mechanizm automatycznego uzupełniania projektu dla znaczników biblioteki PrimeFaces.

Jeśli wybierzesz PrimeFaces jako pakiet komponentów dla projektu JSF, środowisko NetBeans utworzy przykładowy projekt używający komponentów PrimeFaces. Kod wygenerowanego pliku wygląda tak:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:p="http://primefaces.prime.com.tr/ui"
  xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    <h:form>
      <p:commandButton value="Hello from PrimeFaces"
        onclick="dlg1.show();" type="button"
      />
      <p:dialog header="PrimeFaces Dialog" widgetVar="dlg1"
        width="500">
        For more information visit <a href="http://primefaces.
```



```

        org">http://primefaces.org</a>
    </p:dialog>
</h:form>
</h:body>
</html>

```

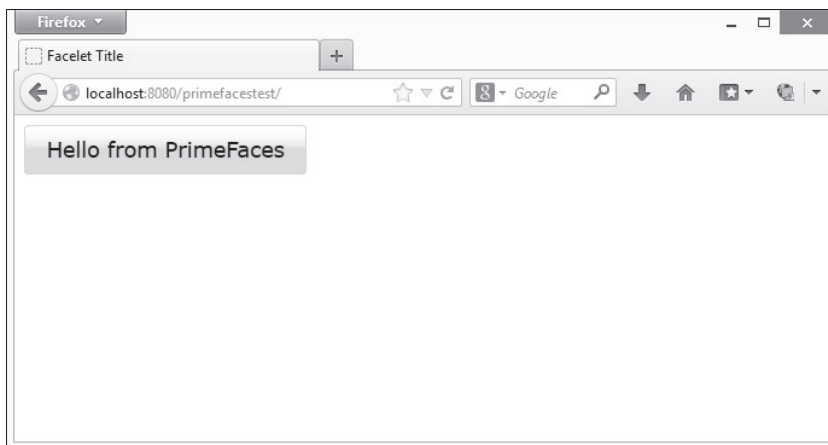
Strona ta wygląda prawie jak zwykła strona faceletu. Wyjątkiem jest kilka komponentów charakterystycznych dla biblioteki PrimeFaces.

Zwróć uwagę na to, że przestrzeń nazw biblioteki PrimeFaces (`xmlns:p="http://primefaces.prime.com.tr/ui"`) jest automatycznie dodawana do znacznika `<html>`. Ta przestrzeń nazw jest niezbędna do stosowania komponentów PrimeFaces na stronach. Zgodnie z konwencją znaczniki PrimeFaces mają przedrostek `p`.

Pierwszy komponent PrimeFaces na przedstawionej stronie to `<p:commandButton>`. Jest on podobny do standardowego komponentu przycisku polecenia JSF, jednak ma pewne dodatkowe zalety (na przykład wygląda elegancko bez konieczności ręcznego dodawania arkusza stylów CSS).

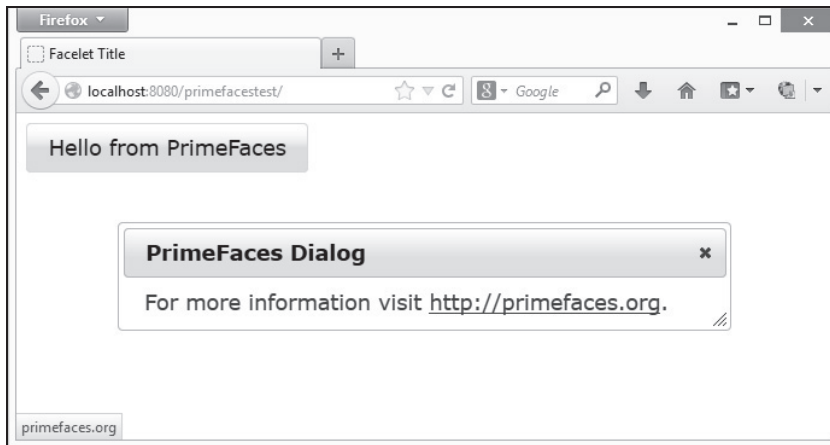
Inny komponent PrimeFaces na przykładowej stronie to `<p:dialog>`. Komponent ten jest wyświetlany jako okno dialogowe, które może znajdować się nad innymi komponentami strony. Za pomocą wartości atrybutu `widgetVar` można uzyskać dostęp do tego komponentu w innych komponentach ze strony. Okno dialogowe udostępnia w tym celu działający po stronie klienta interfejs API JavaScriptu. Najczęściej używane funkcje tego interfejsu API to `show()` i `hide()`. Służą one do wyświetlania i ukrywania okna dialogowego na stronie. Ten interfejs API wykorzystano w atrybucie `onlick` wspomnianego wcześniej przycisku polecenia.

Gdy uruchomisz aplikację, zobaczysz automatycznie wygenerowaną stronę (rysunek 5.2).



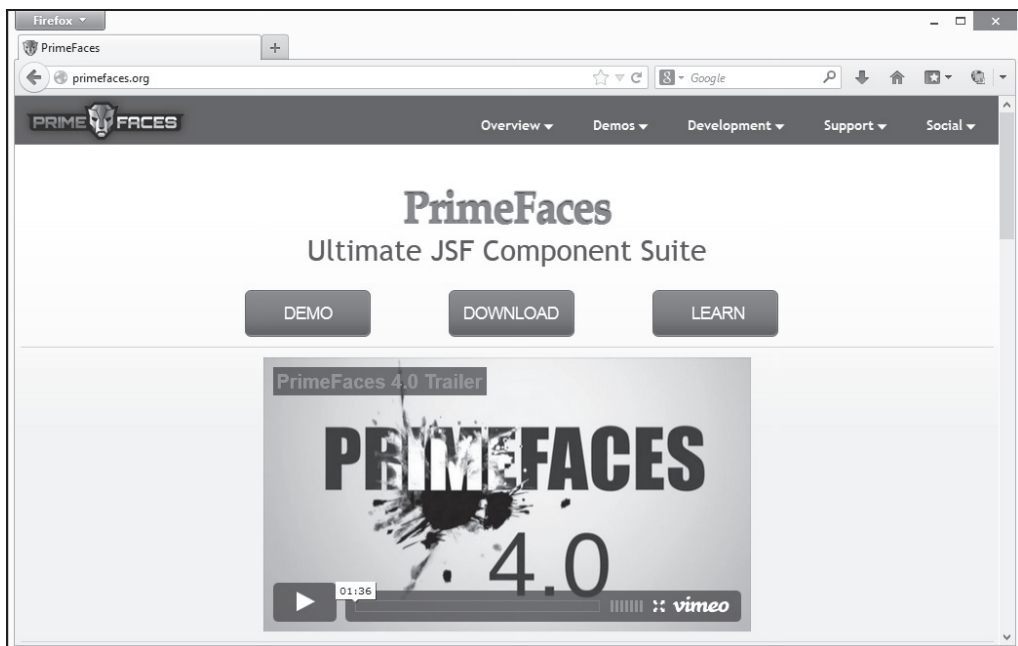
Rysunek 5.2. Automatycznie wygenerowana strona

Po kliknięciu przycisku pojawi się okno dialogowe (rysunek 5.3).



Rysunek 5.3. Okno dialogowe wyświetlane w odpowiedzi na kliknięcie przycisku

Gdy klikniesz odnośnik w oknie dialogowym, przejdziesz do witryny biblioteki PrimeFaces (rysunek 5.4).



Rysunek 5.4. Witryna poświęcona bibliotece PrimeFaces

Jak widać, biblioteka PrimeFaces umożliwia tworzenie eleganckich aplikacji sieciowych małym nakładem sił. Dalej zobaczysz, jak wykorzystać niektóre komponenty PrimeFaces, aby znacznie ułatwić i uprościć proces tworzenia aplikacji sieciowych.

# Stosowanie komponentów PrimeFaces w aplikacjach JSF

Z tego podrozdziału dowiesz się, jak stosować proste komponenty PrimeFaces, aby ułatwić sobie pracę nad tworzeniem aplikacji JSF. Poniżej znajduje się kod prostej strony do wprowadzania danych na temat klientów:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:p="http://primefaces.prime.com.tr/ui"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets">
  <ui:composition template="./template.xhtml">
    <ui:define name="top">
      <h2>Wprowadzanie danych o klientach</h2>
    </ui:define>

    <ui:define name="content">
      <h:form>
        <p:messages/>
        <p:panel header="Wprowadzanie danych">
          <h:panelGrid columns="2">
            <h:outputLabel for="firstName" value="Imię"
              styleClass="requiredLb1"/>
            <h:inputText id="firstName" label="Imię"
              value="#{customer.firstName}"
              required="true"/>
            <h:outputLabel for="middleName" value="Drugie imię"
              styleClass="optionalLb1"/>
            <h:inputText id="middleName" label="Drugie imię"
              value="#{customer.middleName}"/>
            <h:outputLabel for="lastName" value="Nazwisko"
              styleClass="requiredLb1"/>
            <h:inputText id="lastName" label="Nazwisko"
              value="#{customer.lastName}"
              required="true"/>
            <h:outputLabel for="birthDate" value="Data urodzenia"
              styleClass="optionalLb1"/>
            <p:calendar id="birthDate"
              value="#{customer.birthDate}"
              showOn="button"
              inputStyle="width:100px;"
              navigator="true"/>
          </h:panelGrid>
        </p:panel>
      </h:form>
    </ui:define>
  </ui:composition>
</html>
```

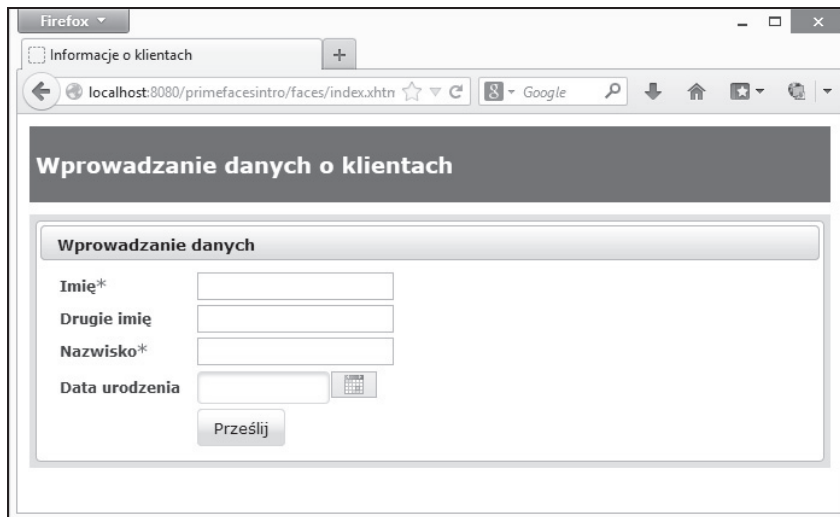
```

        <p:commandButton
            value="Prześlij"
            action="#{customerController.
                saveCustomer}"
            ajax="false"/>
    </h:panelGrid>
</p:panel>
</h:form>
</ui:define>
</ui:composition>
</html>

```

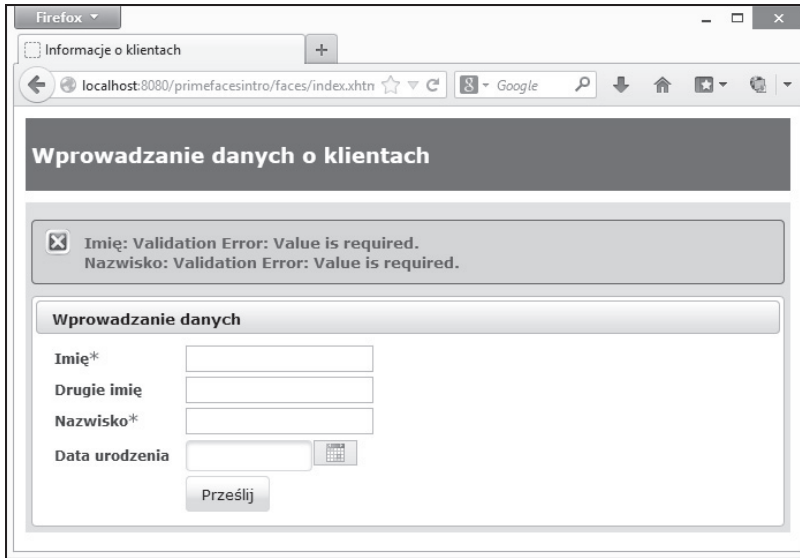
W przykładzie wykorzystano dostępny w środowisku NetBeans mechanizm generowania szablonów faceletu. Dzięki temu można „za darmo” uzyskać bardzo ciekawe style CSS. Więcej informacji o szablonach faceletów w środowisku NetBeans znajdziesz w poprzednim rozdziale.

Gdy uruchomisz projekt, przedstawiony kod zostanie wyświetlony w przeglądarce tak jak na rysunku 5.5.



Rysunek 5.5. Strona do wprowadzania danych w przeglądarce

Pierwszy nowy komponent PrimeFaces zastosowany na tej stronie to `<p:messages>`. Można go wykorzystać jako zamiennik standardowego komponentu JSF `<h:messages>`. Zaletą znacznika `<p:messages>` w porównaniu ze znacznikiem `<h:messages>` jest to, że komunikaty o błędach w znaczniku `<p:messages>` są domyślnie formatowane w elegancki sposób (rysunek 5.6).



Rysunek 5.6. Sformatowane komunikaty o błędach w znaczniku `<p:messages>`

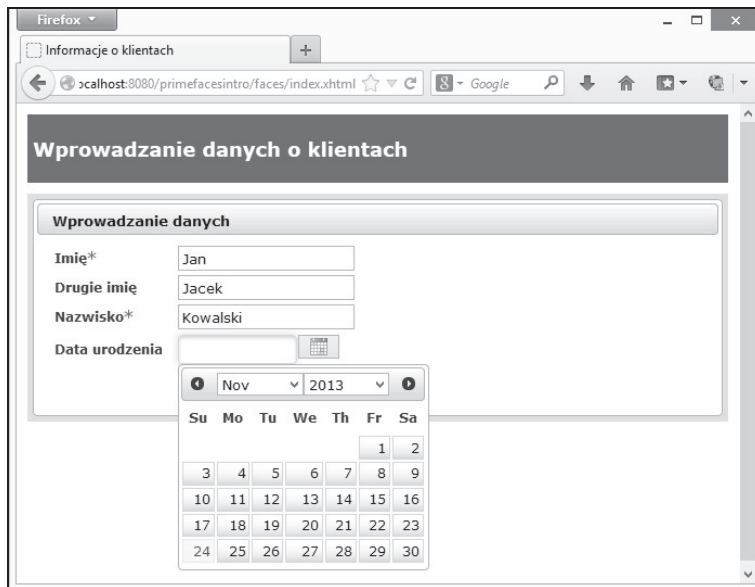
Podobnie jak w przypadku komponentu `<p:messages>`, biblioteka PrimeFaces udostępnia komponent `<p:message>`, który można potraktować jako zamiennik standardowego komponentu JSF `<h:message>` (w przykładzie nie zastosowano komponentu `<p:message>`). Znacznik `<p:message>`, podobnie jak `<h:message>`, należy stosować do wyświetlania błędów związanych ze sprawdzaniem poprawności obok pól z nieprawidłowymi danymi, a nie w górnej części strony.

Inny komponent PrimeFaces wykorzystany w przykładzie to `<p:calendar>`. Kalendarz ten można wykorzystać do wprowadzania dat. Gdy użytkownik kliknie ikonę wygenerowaną dla tego komponentu, pojawi się atrakcyjny widżet z kalendarzem (rysunek 5.7). Użytkownik może w nim wybrać datę, klikając ją.

Komponent kalendarza biblioteki PrimeFaces można w szerokim zakresie modyfikować. Domyślnie jest wyświetlany jako pole tekstowe, a gdy użytkownik je kliknie, pojawia się widżet z kalendarzem. W przykładzie atrybut `showOn` ma wartość `button`. To powoduje, że obok pola tekstowego widoczna jest ikona kalendarza. Jest to wizualna wskazówka informująca o specjalnym charakterze pola. Pozwala to także wprowadzić datę ręcznie, jeśli użytkownik woli zastosować to podejście.

Domyślnie listy miesięcy i lat są niedostępne, co utrudnia wprowadzanie dat z odległej przeszłości i przyszłości (domyślnym dniem jest zawsze dzień bieżący). Aby rozwiązać ten problem, można ustawić właściwość `navigator` na `true`, jak zrobiono w przykładowym kodzie.

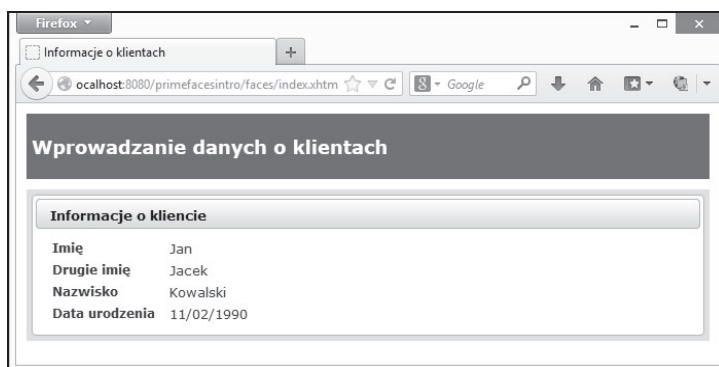
Ponadto można kontrolować wygląd pola tekstowego. Umożliwiają to atrybuty `inputStyle` i `inputStyleClass` komponentu `<p:calendar>`. Wartością atrybutu `inputStyle` musi być poprawny wewnątrzwierszowy kod CSS, natomiast w atrybucie `inputStyleClass` trzeba podać nazwę klasy CSS zdefiniowanej w jednym z arkuszy stylów CSS.



Rysunek 5.7. Komponent kalendarza umożliwia wybranie daty

Ostatnim nowym komponentem PrimeFaces użytym na przykładowej stronie jest `<p:commandButton>`. Domyślnie wyświetla on przycisk, który zgłasza ajaksowe żądanie aktualizacji części strony. Nie jest wtedy konieczne odświeżanie całej strony. Komponent ten można wykorzystać jako zamiennik standardowego komponentu JSF `<h:commandButton>`. W tym celu właściwość `ajax` należy ustawić na `false`, tak jak w przykładowym kodzie. Zaletą stosowania komponentu `<p:commandButton>` zamiast `<h:commandButton>` jest to, że ma on domyślnie elegancki wygląd. Dzięki temu nie trzeba tworzyć niestandardowych stylów CSS dla przycisków.

Gdy klikniesz przycisk na przykładowej stronie, przejdziesz do strony z potwierdzeniem (rysunek 5.8).



Rysunek 5.8. Strona z potwierdzeniem

Na stronie z potwierdzeniem nie ma żadnych nowych komponentów PrimeFaces, dlatego strona ta nie wymaga omówienia. Znajdziesz ją w dostępnym do pobrania kodzie dla tego rozdziału (plik *confirmation.xhtml*).

## Widoki z zakładkami

W formularzach HTML często znajduje się dużo pól, przez co formularze mogą być bardzo długie. Programiści nieraz dzielą formularze na zakładki, dzięki czemu strona jest mniej przytłaczająca dla użytkowników. Zwykle tworzenie stron z zakładkami wymaga stosowania sztuczek z języków HTML i JavaScript. Jednak biblioteka PrimeFaces udostępnia komponent `<p:tabView>`, który można wykorzystać do łatwego generowania zakładek. W poniższym przykładzie pokazano, jak korzystać z tego komponentu.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:p="http://primefaces.prime.com.tr/ui"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:f="http://java.sun.com/jsf/core">
<ui:composition template="./template.xhtml">
  <ui:define name="top">
    <h2>Wprowadzanie danych o klientach</h2>
  </ui:define>
  <ui:define name="content">
    <h:form>
      <p:messages/>
      <h:panelGrid columns="1" style="width: 100%">
        <p:tabView>
          <p:tab title="Dane osobowe">
            <h:panelGrid columns="2">
              <h:outputLabel for="firstName"
                value="Imię"
                styleClass="requiredLb1"/>
              <h:inputText id="firstName"
                label="Imię"
                value="#{customer.firstName}"
                required="true"/>
              <h:outputLabel for="middleName"
                value="Drugie imię"
                styleClass="optionalLb1"/>
              <h:inputText id="middleName"
                label="Drugie imię"
                value="#{customer.middleName}"/>
            </h:panelGrid>
          </p:tab>
        </p:tabView>
      </h:form>
    </ui:define>
  </ui:composition>
</html>
```

```

<h:outputLabel for="lastName"
  value="Nazwisko"
  styleClass="requiredLb1"/>
<h:inputText id="lastName" label="Nazwisko"
  value="#{customer.
  lastName}"
  required="true"/>
<h:outputLabel for="birthDate"
  value="Data urodzenia"
  styleClass="optionalLb1"/>
<p:calendar id="birthDate"
  value="#{customer.
  birthDate}"
  showOn="button"
  inputStyle="width:100px;"
  navigator="true"/>
</h:panelGrid>
</p:tab>
<p:tab title="Adres">
  <h:panelGrid columns="2">
    <h:outputLabel for="line1" value="Wiersz 1."
      styleClass="requiredLb1"/>
    <h:inputText id="line1"
      value="#{customer.
      addrLine1}"
      required="true"/>
    <h:outputLabel for="line2" value="Wiersz 2."
      styleClass="optionalLb1"/>
    <h:inputText id="line2"
      value="#{customer.
      addrLine2}"/>
    <h:outputLabel for="city" value="Miasto"
      styleClass="requiredLb1"/>
    <h:inputText id="city"
      value="#{customer.
      addrCity}"
      required="true"/>
    <h:outputLabel for="state"
      value="Województwo"
      styleClass="requiredLb1"/>
    <h:selectOneMenu id="state"
      required="true"
      value="#{customer.addrState}">
      <f:selectItem itemValue=""
        itemLabel=""/>
      <f:selectItem itemValue="LB"
        itemLabel="Lubelskie"/>
      <f:selectItem itemValue="MZ"
        itemLabel="Mazowieckie"/>

```



```

        <f:selectItem itemValue="OP"
            itemLabel="Opolskie"/>
        <f:selectItem itemValue="PM"
            itemLabel="Pomorskie"/>
        <!-- Uwagi na zwięzłość przykładu pozostałe
            województwa zostały pominięte -->
    </h:selectOneMenu>
    <h:outputLabel for="zip" value="Kod"
        styleClass="requiredLb1"/>
    <h:inputText id="zip"
        value="#{customer.
            addrZip}"
        required="true"/>
    </h:panelGrid>
</p:tab>
<p:tab title="Numery telefonu">
    <h:panelGrid columns="2">
        <h:outputLabel for="homePhone"
            value="Domowy"/>
        <p:inputMask id="homePhone"
            mask="(999)-999-999"
            value="#{customer.
                homePhone}"
            size="12"
            styleClass="optionalLb1"/>
        <h:outputLabel for="mobilePhone"
            value="Komórkowy"/>
        <p:inputMask id="mobilePhone"
            mask="(999)-999-999"
            value="#{customer.
                mobilePhone}"
            size="12"
            styleClass="optionalLb1"/>
        <h:outputLabel for="workPhone"
            value="Służbowy"/>
        <p:inputMask id="workPhone"
            mask="(999)-999-999"
            value="#{customer.
                workPhone}"
            size="12"
            styleClass="optionalLb1"/>
    </h:panelGrid>
</p:tab>
</p:tabView>
<p:commandButton
    value="Prześlij"
    action="#{customerController.saveCustomer}"
    ajax="false"/>
</h:panelGrid>

```

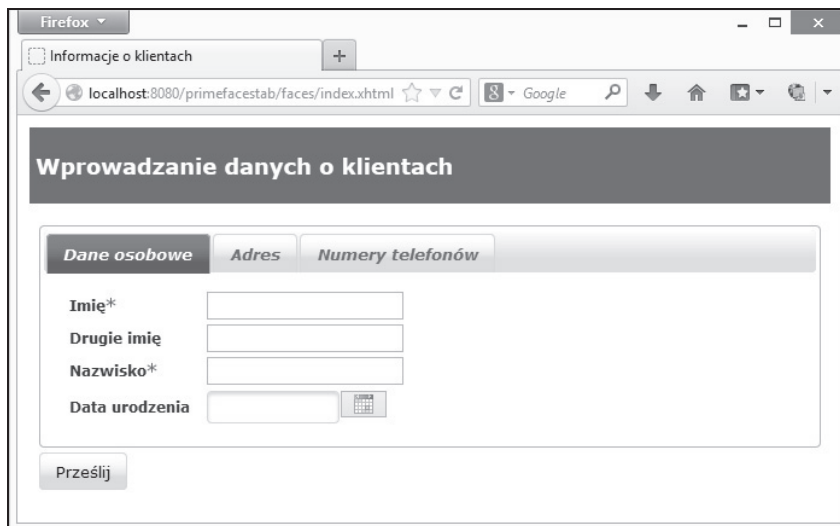
```

    </h:form>
  </ui:define>
</ui:composition>
</html>

```

Jak widać w tym przykładzie, nadrzędnym komponentem interfejsu z zakładkami jest `<p:tabView>`. Wewnątrz niego trzeba umieścić jeden lub kilka komponentów `<p:tab>`. Każdy komponent `<p:tab>` zawiera tu pola na dane wejściowe. Pola te pojawiają się w odpowiednich zakładkach. Komponent `<p:tab>` ma atrybut `title`, którego wartość jest wyświetlana jako nagłówek zakładki.

Gdy uruchomisz projekt, zobaczysz, jak działa komponent zakładki (rysunek 5.9).



Rysunek 5.9. Zakładki na stronie

Po kliknięciu poszczególnych zakładek zobaczysz umieszczone na nich komponenty (rysunek 5.10).

Uważni Czytelnicy zapewne zauważyli, że w trzeciej zakładce pojawił się nowy komponent PrimeFaces — `<p:inputMask>`. Pozwala on uniemożliwić użytkownikom wprowadzenie źle sformatowanych danych. Tu komponent ten wykorzystano dla każdego pola na numer telefonu. Działanie tego komponentu przedstawia rysunek 5.11.

Jak widać na rysunku, gdy użytkownik kliknie komponent `inputMask`, automatycznie wyświetlany jest oczekiwany format. Użytkownik musi tylko uzupełnić puste miejsca, aby dane były poprawnie sformatowane. Tu oczekiwany jest numer telefonu w formacie `(xxx)xxx-xxx`, gdzie `x` to cyfra w systemie dziesiętnym. Przy definiowaniu masek cyfra `9` reprezentuje dowolną cyfrę, litera `a` reprezentuje dowolną literę, a gwiazdka `(*)` — dowolną cyfrę lub literę.

The screenshot shows a Firefox browser window with the address bar displaying 'localhost:8080/primefacestab/faces/index.xhtml'. The page title is 'Informacje o klientach'. The main heading is 'Wprowadzanie danych o klientach'. Below the heading, there are three tabs: 'Dane osobowe', 'Adres', and 'Numery telefonów'. The 'Adres' tab is currently selected. The form contains the following fields:

- Wiersz 1.\*
- Wiersz 2.
- Miasto\*
- Województwo\* (dropdown menu)
- Kod\*

At the bottom left of the form is a 'Prześlij' button.

Rysunek 5.10. Komponenty z zakładki Adres

This screenshot shows the same web browser window, but now the 'Numery telefonów' tab is selected. The form fields are populated with the following values:

- Domowy: (771)-326-543
- Komórkowy: ( )- - -
- Służbowy: (empty)

The 'Prześlij' button remains at the bottom left.

Rysunek 5.11. Działanie komponentu `<p:inputMask>`

Tu w celu formatowania danych przypisano do atrybutu `mask` komponentu `<p:inputMask>` wartość `(999)-999-999`, co pozwoliło uzyskać pożądaną maskę.

Komponent `<p:inputMask>` pozwala więc wymuszać poprawne formatowanie danych bez stosowania walidatorów JSE.

Pora wrócić do omawiania zakładki. Gdy użytkownik kliknie widoczny w dolnej części strony przycisk *Prześlij*, dane z wszystkich zakładek zostaną przesłane i potraktowane jak informacje z jednego znacznika `<h:form>`. Od tego momentu działa standardowy dla JSF proces przetwarzania żądań.

## Interfejsy oparte na kreatorze

Inną (obok zakładek) popularną techniką dzielenia długich formularzy jest stosowanie kreatorów. Są one przydatne, gdy użytkownik musi uzupełnić pola w określonej kolejności. W poprzednim przykładzie nie można wymusić na użytkownika wprowadzenia adresu przed podaniem numeru telefonu. Dlatego nie można sprawdzić, czy wprowadzone numery pasują do miejsca zamieszkania podanego w adresie. Aby rozwiązać ten problem, można wykorzystać interfejsy oparte na kreatorze. Pożądaný efekt można łatwo uzyskać za pomocą komponentu PrimeFaces `<p:wizard>`. Poniżej pokazano, jak zastosować ten komponent.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:p="http://primefaces.prime.com.tr/ui"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:f="http://java.sun.com/jsf/core">
  <ui:composition template="./template.xhtml">
    <ui:define name="top">
      <h2>Wprowadzanie danych o użytkownikach</h2>
    </ui:define>

    <ui:define name="content">
      <h:form>
        <h:panelGrid columns="1" style="width: 100%">
          <p:wizard>
            <p:tab title="Dane osobowe"
              id="personalInfo">
              <p:panel header="Dane osobowe">
                <p:messages/>
                <h:panelGrid columns="2">

                  <h:outputLabel for="firstName"
                    value="Imię"
                    styleClass="requiredLbl"/>
                  <h:inputText id="firstName"
                    label="Imię"
                    value="#{customer.
                      firstName}"
                    required="true"/>
```

```

        <h:outputLabel for="middleName"
            value="Drugie imię"
            styleClass="optionalLb1"/>
        <h:inputText id="middleName"
            label="Drugie imię"
            value="#{customer.
                middleName}"
        />
        <h:outputLabel for="lastName"
            value="Nazwisko"
            styleClass="requiredLb1"/>
        <h:inputText id="lastName"
            label="Nazwisko"
            value="#{customer.
                lastName}"
            required="true"/>
        <h:outputLabel for="birthDate"
            value="Data urodzenia"
            styleClass="optionalLb1"/>
        <p:calendar id="birthDate"
            value="#{customer.
                birthDate}"
            showOn="button"
            inputStyle="width:100px;"
            navigator="true"/>
    </h:panelGrid>
</p:panel>
</p:tab>
<!-- Zakładkę z numerami telefonów pominięto
z uwagi na zwięzłość przykładu -->
<p:tab title="Potwierdzenie" id="confirmation">
    <p:messages/>
    <h:panelGrid columns="2">
        <h:outputText value="Imię"
            styleClass="optionalLb1"/>
        <h:outputText id="firstNameTxt"
            value="#{customer.
                firstName}" />
        <h:outputText value="Drugie imię"
            styleClass="optionalLb1"/>
        <h:outputText id="middleNameTxt"
            value="#{customer.
                middleName}" />
        <h:outputText value="Nazwisko"
            styleClass="optionalLb1"/>
        <h:outputText id="lastNameTxt"
            value="#{customer.
                lastName}" />
        <h:outputText value="Data urodzenia"
            styleClass="optionalLb1"/>
    </h:panelGrid>
</p:tab>

```

```

        <h:outputText id="birthDateTxt"
            value="#{customer.
                formattedBirthDate}" />
        <h:outputText value="Telefon domowy"
            styleClass="optionalLbl"/>
        <h:outputText id="homePhoneTxt"
            value="#{customer.
                homePhone}" />
        <h:outputText value="Telefon komórkowy"
            styleClass="optionalLbl"/>
        <h:outputText id="mobilePhoneTxt"
            value="#{customer.
                mobilePhone}" />
        <h:outputText value="Telefon służbowy"
            styleClass="optionalLbl"/>
        <h:outputText id="workPhoneTxt"
            value="#{customer.
                workPhone}" />
        <h:panelGroup/>
        <h:inputHidden value="#{customer.
            addrState}" />
        <p:commandButton id="submitButton"
            value="Prześlij"
            actionListener=
                "#{customerController.saveCustomer}"
            ajax="false"/>
    </h:panelGrid>
</p:tab>
</p:wizard>
</h:panelGrid>
</h:form>
</ui:define>
</ui:composition>
</html>

```

Niektóre strony („zakładki”) kreatora zostały pominięte w kodzie z uwagi na konieczność zachowania zwięzłości tego przykładu. Kompletną wersję znajdziesz w dostępnym do pobrania kodzie do tego rozdziału.

Jak widać, biblioteka PrimeFaces sprawia, że generowanie interfejsów opartych na kreatorze jest proste. Wystarczy zastosować komponent `<p:wizard>`, a następnie dodać komponent `<p:tab>` dla każdego etapu kreatora. W każdym komponencie `<p:tab>` można umieścić standardowy kod JSF.

W ostatniej zakładce znajduje się komponent `<p:commandButton>` służący do przesyłania danych na serwer. Wartość atrybutu `actionListener` tego komponentu to metoda z ziarna zarządzanego `CustomerController`.

```

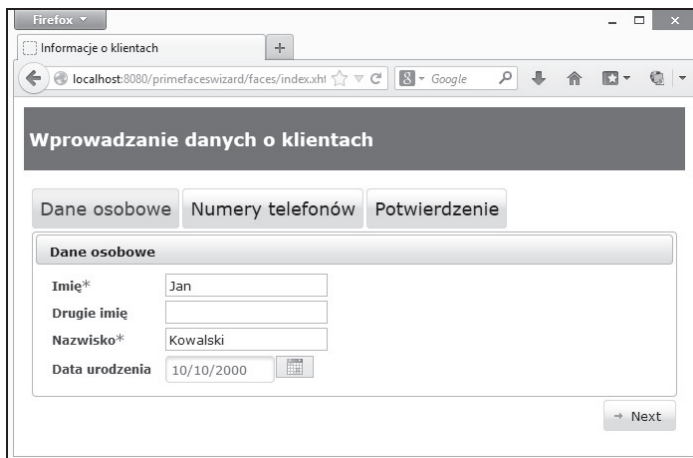
package com.ensode.primefacesdemo.managedbeans;
import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.event.ActionEvent;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
@ManagedBean
@SessionScoped
public class CustomerController implements Serializable {
    /** Tworzy nowy obiekt typu CustomerController */
    public CustomerController() {
    }

    public void saveCustomer(ActionEvent actionEvent) {
        // W rzeczywistej aplikacji należałoby zapisać dane.
        // W tym przykładzie kod tylko wyświetla komunikat.
        FacesMessage facesMessage = new FacesMessage(
            "Dane zostały zapisane");
        facesMessage.setSeverity(FacesMessage.SEVERITY_INFO);
        FacesContext.getCurrentInstance().addMessage(null,
            facesMessage);
    }
}

```

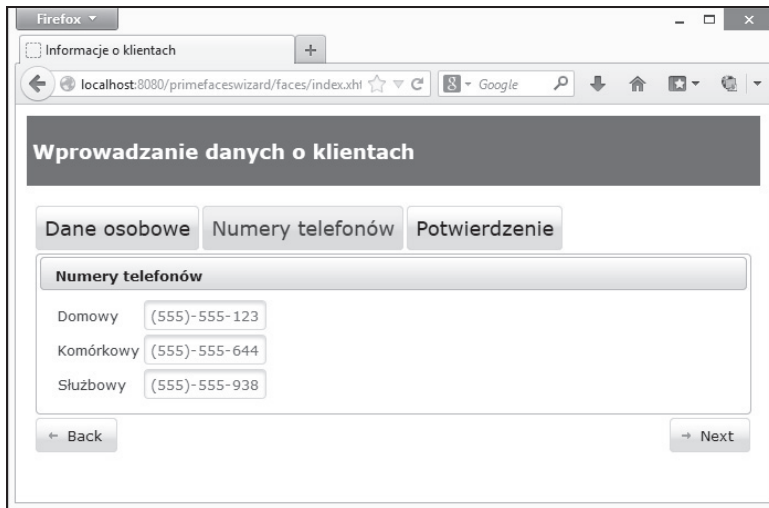
Metoda `saveCustomer()` tylko ustawia komunikat wyświetlany jako potwierdzenie na stronie. W rzeczywistej aplikacji oczywiście należałoby zapisać dane w bazie.

Na tym etapie można zobaczyć, jak działa komponent kreatora (rysunek 5.12).



Rysunek 5.12. Działanie komponentu kreatora

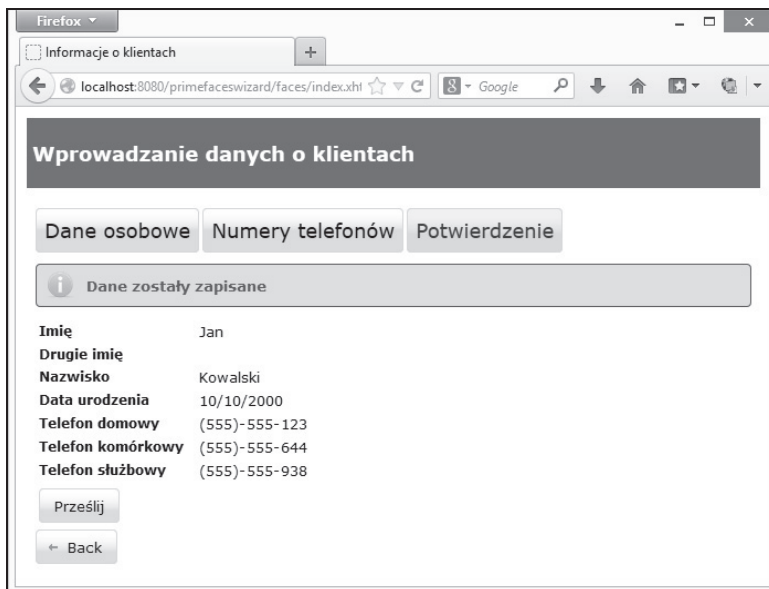
Zauważ, że komponent kreatora automatycznie dodaje przycisk *Next* w prawym dolnym rogu strony (rysunek 5.13). Kliknięcie tego przycisku powoduje przejście do następnej strony kreatora.



Rysunek 5.13. Przycisk Next pozwala przechodzić między stronami kreatora

Jak można się domyślić, komponent kreatora generuje zarówno przycisk *Next*, jak i przycisk *Back*.

Po przejściu w kreatorze do ostatniej strony i kliknięciu przycisku *Prześlij* pojawi się komunikat z potwierdzeniem wygenerowany przez metodę `saveCustomer()` z ziarna zarządzanego `CustomerController`. Komunikat ten jest wyświetlany przez komponenty `<p:messages>`, które zapewniają elegancki styl tekstu bez konieczności jawnego stosowania stylów CSS (rysunek 5.14).



Rysunek 5.14. Komponent `<p:messages>` zapewnia elegancki styl komunikatu



## Dodatkowe informacje

W tym rozdziale przedstawiono tylko podstawowe informacje na temat możliwości biblioteki PrimeFaces. Pokaz ilustrujący wszystkie komponenty z tej biblioteki znajdziesz na stronie <http://www.primefaces.org/showcase/ui/home.jsf>. Więcej informacji na temat tej biblioteki zawiera witryna <http://www.primefaces.org>.

## Podsumowanie

Ten rozdział zawiera wprowadzenie do biblioteki komponentów JSF PrimeFaces dostępnej w środowisku NetBeans od wersji 7.0. Pokazano tu, w jaki sposób biblioteka PrimeFaces umożliwia łatwe tworzenie eleganckich aplikacji z obsługą Ajaksa. Jednym z ważnych opisanych tu komponentów biblioteki PrimeFaces jest komponent zakładek, który umożliwia wygodne dzielenie stron na zakładki. Dowiedziałeś się też, jak stosować komponent kreatora z biblioteki PrimeFaces, pozwalający na łatwe tworzenie w aplikacjach sieciowych interfejsów opartych na kreatorach.



# Skorowidz

## A

adnotacja @  
  ActivationConfigProperty, 283  
  AroundInvoke, 240  
  EJB, 301  
  Entity, 193, 206  
  FacesValidator, 151  
  Id, 193  
  InterceptorBinding, 263  
  Interceptors, 240, 241  
  ManagedBean, 145  
  MessageDriven, 283  
  Named, 253  
  NamedQueries, 207  
  Path, 314, 319  
  Qualifier, 256  
  RequestScoped, 145  
  Schedule, 243  
  Stateless, 314  
  Table, 207  
  TransactionAttribute, 238  
  ViewScoped, 145  
  WebServlet, 73  
adres URL, 77  
aktualizowanie danych w bazie, 116  
AOP, Aspect Oriented Programming, 248  
aplikacje  
  CDI, 251  
  debugowanie, 327  
  JSF, 132, 217, 251  
  JSP, 215  
  profilowanie, 335  
  sieciowe, 51, 131  
  z kwalifikatorami, 260

architektura JMS, 270  
atrybuty dyrektywy strony, 57  
automatyczna obsługa zasobów, 159  
automatyczne generowanie  
  encji, 200  
  klas, 296  
  komunikatów, 136, 149  
  stron, 167

## B

baza danych, 308  
  aktualizowanie danych, 116  
  modyfikowanie danych, 114  
  schemat APP, 190  
  usuwanie danych, 119  
  wstawianie danych, 114  
bezpieczeństwo  
  aplikacji, 86  
  serwera, 89  
bezstanowe ziarno sesyjne, 298, 314  
biblioteka  
  JSTL, 97, 122, 126  
  PrimeFaces, 165  
  znaczników niestandardowych, 127

## C

CDDL, Common Development and Distribution License, 21  
CDI, Contexts and Dependency Injection, 249

**D**

DAO, Data Access Object, 195  
 debugger, 327–333  
 deskryptor
 

- glassfish-web.xml, 90
- wdrażania, 73, 85

 docelowa lokalizacja JMS, 272  
 dodawanie
 

- adnotacji @Interceptors, 241
- deskryptora wdrażania, 85, 108
- grup bezpieczeństwa, 90
- instrukcji INSERT, 114
- metod do ziaren, 232, 248
- nowej strony JSP, 67
- pliku klienta, 294
- pól, 62
- projektu biblioteki, 234
- referencji, 109
- ról zabezpieczeń, 86
- serwletu, 72
- szablonu faceletu, 154
- utrwalanych pól, 194
- użytkowników, 92
- znaczników, 113

 dokument JSR, 208  
 domena bezpieczeństwa, security realm, 81  
 dostęp do
 

- obiektów JSTL, 104
- stron, 87
- ziarna, 233

 dostępne
 

- metody obiektu, 70
- obiekty, 69

 dyrektywa strony JSP, 56  
 działanie typu do wiązania interceptorów, 266  
 dzielenie formularzy, 178

**E**

EAR, Enterprise Application, 224  
 EJB, Enterprise JavaBeans, 223  
 ekran
 

- instalatora, 25
- startowy, 26

 encje JPA, 186, 192, 215, 244  
 etap projektu JSF, 135

**F**

fabryka połączeń, 274  
 facelet, 136, 251  
 format
 

- JSON, 317
- XML, 317

 formatowanie kodu, 60  
 formularz, 58, 65  
 fragmenty JSP, 93

**G**

generowanie
 

- aplikacji JSP, 215
- encji JPA, 200, 203, 205
- interfejsów, 178
- klas, 296, 311
- klasy kontrolerów, 196
- kluczy głównych, 193, 218
- kodu
  - do wysyłania komunikatów, 276
  - klienta, 296, 319
  - kwalifikatora, 256
  - nowego typu, 264
  - stereotypu, 262
  - usługi sieciowej, 287, 299, 302, 305
- kwerend JPQL, 207
- metod, 291
- serwletów, 73
- stron, 167
- stron JSF, 215
- szablonów, 152
- szablonów faceletu, 170
- tabeli, 61
- usług sieciowych, 308
- ziaren sesyjnych, 231, 244

 getter, 145, 195  
 GPL, GNU Public License, 21  
 grupa bezpieczeństwa, 90

**H**

hierarchia bieżącej klasy, 45

## I

IDE, Integrated Development Environment, 15  
 identyfikator  
   JNDI, 224, 276  
   URI, 98, 307  
 ikona  
   błąd kompilacji, 48  
   implementacja interfejsu, 49  
   ostrzeżenie, 48  
   przesłanie metod, 48  
 implementowanie  
   klienta, 317  
   strony błędu logowania, 84  
   uwierzytelniania, 82  
   warstwy biznesowej, 223  
 informacje  
   o bazie danych, 201  
   o przywracaniu pamięci, 338  
 instalowanie aplikacji, 35  
 instalowanie środowiska, 19  
   Linux, 20  
   Mac OS X, 19  
   Microsoft Windows, 19  
   Solaris, 20  
 instrukcja  
   DELETE, 120, 121  
   INSERT, 114  
   UPDATE, 119  
 integrowanie środowiska, 27  
   z serwerem aplikacji, 27  
   z systemem RDBMS, 30  
 interceptor, 239, 263  
 interfejs  
   API Servlet, 51  
   CDI, 249  
   graficzny, 288  
   javax.jms.Message, 278  
   javax.jms.MessageListener, 283  
   javax.servlet.jsp.jstl.sql.Result, 110  
   JAX-RS, 307  
   JAX-WS, 285  
   JMS, 270  
   JPA, 185  
   JTA, 192  
   lokalny, 230  
   zdalny, 230

## J

J2EE, 223, 237  
 Java EE, 16  
 Java ME, 16  
 Java SE, 16  
 JAXB, Java API for XML Binding, 203  
 JAX-RS, 307  
 JDK, Java Development Kit, 19  
 język  
   JPQL, 207  
   UEL, 252  
 JMS, Java Messaging Service, 269  
 JNDI, Java Naming and Directory Interface, 109  
 JPA, Java Persistence API, 16, 185  
 JPQL, Java Persistence Query Language, 207  
 JSF, JavaServer Faces, 54, 131  
   biblioteka PrimeFaces, 165  
   komponenty złożone, 159  
   nowy projekt, 133  
   poprawność danych, 148  
   szablony, 152  
 JSON, JavaScript Object Notation, 317  
 JSP, Java Server Pages, 51  
 JSR, Java Specification Request, 208  
 JSTL, Java Standard Tag Library, 97  
 JTA, Java Transaction API, 192

## K

kalendarz, 171  
 katalog  
   instalacyjny, 23  
   resources, 140  
 klasa  
   AbstractClass, 312  
   ApplicationConfig, 318  
   interceptora, 240  
   ValidatorException, 151  
 klasy  
   fasady, 311  
   kontrolerów, 195  
 klient, 237  
   szablonu, 158  
   usługi sieciowej, 293, 319  
 klucz główny, 193  
 kod klienta usługi sieciowej, 297

- kolejka komunikatów, 272
- komponent kreatora, 181
- komponenty
  - PrimeFaces, 165, 183
  - złożone, 159
- komunikacja z bazą, 187
- komunikat
  - JMS, 271, 276
  - o błędzie, 149, 151, 171
  - SOAP, 289
- konfigurowanie
  - bezpieczeństwa aplikacji, 86, 89, 90
  - dostępu do stron, 87
  - fabryki połączeń, 275
  - klasy kontrolera, 197
  - klienta, 320
  - kodu do wysyłania komunikatów, 277
  - kolejki komunikatów, 273
  - komponentu, 160
  - kwalifikatora, 257
  - metody, 232
  - nowego ziarna, 281
  - odwzorowań, 205
  - operacji, 289
  - operacji wstawiania danych, 114
  - połączenia, 190
  - procesu generowania ziaren, 245
  - serwletu, 74
  - środowiska, 27
  - usługi sieciowej, 301, 304
  - ziarna, 144
  - ziarna sesyjnego, 229
  - znacznika
    - <c:choose>, 102
    - <c:forEach>, 105
    - <c:if>, 99
    - <sql:query>, 111
    - <sql:update>, 117, 120
  - konsola administracyjna serwera, 90, 91
  - kontener serwletów, 22
  - kontroler, 72
  - kreator
    - Create Persistence Unit, 187
    - interfejsu, 180
    - klas encji, 187
    - New Entity Classes from Database, 207
    - New JSF Managed Bean, 145
      - usług sieciowych, 299
    - kwalifikator CDI, 256
    - kwerendy nazwane, 207

**L**

  - licencja
    - CDDL, 21
    - GPL, 21
  - lista
    - obiektów, 69
    - szablonów kodu, 44

**Ł**

  - łańcuch znaków, 323

**M**

  - mechanizm MTOM, 290
  - metoda
    - GET, 59
    - getProjectStage(), 136
    - getServletContext(), 77
    - getSession(), 77
    - main(), 278
    - POST, 59
    - saveCustomer(), 181
    - sendJMSMessageToMyQueue(), 278
  - metody interfejsu javax.servlet.jsp.jstl.sql.Result, 110
  - model, 72
  - modyfikowanie
    - danych w bazie, 114
    - deskryptora wdrażania, 89
    - kodu, 57
    - skryptletu, 79
    - strony, 58, 138
    - usługi sieciowej, 288
  - MVC, Model-View-Controller, 72

**N**

  - narzędzia diagnostyczne, 136
  - nawigacja
    - dynamiczna, 142
    - statyczna, 142

nazwa grupy, 90  
nazwy ról zabezpieczeń, 90  
NetBeans  
  instalowanie aplikacji, 35  
  instalowanie środowiska, 19  
  konfigurowanie środowiska, 27  
  pierwsze uruchomienie, 26  
  pobieranie środowiska, 16  
  wydajne programowanie, 38

nowa

  baza danych, 201  
  encja, 216  
  operacja, 289  
  pula połączeń, 189  
  rola, 87  
  strona, 66  
  usługa sieciowa, 300

nowe

  operacje usługi, 290  
  ziarno, 281  
  ziarno sesyjne, 229

nowy

  deskryptor wdrażania, 109  
  egzemplarz encji, 219  
  fragment JSP, 95  
  projekt, 36  
  projekt JSF, 133  
  szablon faceletu, 153

## O

obiekty

  DAO, 195  
  na stronach JSP, 69  
  POJO, 186, 298  
  SurveyData, 76

obsługa

  interfejsu CDI, 250  
  widoków, 251, 253

okno

  Local Variables, 333  
  usługi sieciowej, 302  
  VM Telemetry, 337

określanie pakietu zasobów, 310

opcja

  Connect Using..., 33  
  Create Database..., 200

  Deploy, 101  
  Getters and Setters, 145  
  Run File, 64  
  Test Resource Uri, 314  
operacje  
  CRUD, 307  
  usługi sieciowej, 295  
operatory JSTL, 104

## P

pakiet JDK, 19

pakiety środowiska NetBeans, 18

platforma JSF, 54, 131

plik

  glassfish-resources.xml, 276  
  web.xml, 90, 108, 135

pliki

  EAR, 224  
  JSP, 56, 68  
  WAR, 224  
  WSDL, 286, 293, 303  
  znaczników niestandardowych, 122

pobieranie

  danych, 138  
  danych z bazy, 109  
  referencji do obiektu, 235  
  środowiska, 16

POJO, Plain Old Java Object, 186, 298

poła do wprowadzania danych, 62

pole wyboru, 62, 63

połączenie z RDBMS, 32

profiler, 335–339

programowanie

  aspektowe, 239, 248, 263, 267  
  serwletów, 72

projekt

  Enterprise Application, 225, 271  
  Java Class Library, 227

protokół

  REST, 285  
  SOAP, 285

przenoszenie wspólnego kodu, 96

przesyłanie

  danych, 323  
  formularza, 64  
  komunikatów, 275–279

przycisk Remove Operation, 289  
 puła połączeń, 187, 188

## R

referencja do interfejsu zdalnego, 236  
 relacja  
   jeden do jednego, 211, 213  
   jeden do wielu, 211, 213  
   wiele do jednego, 211  
   wiele do wielu, 211, 213  
 relacje  
   dwukierunkowe, 214  
   jednokierunkowe, 214  
   między encjami, 209  
 REST, Representational State Transfer, 285, 307  
 role zabezpieczeń, 86

## S

schemat  
   APP, 190  
   bazy danych, 34, 203  
 serwer  
   aplikacji, 19, 27  
   aplikacji GlassFish, 22, 54  
   Tomcat, 54  
 serwlet, 51, 72, 74  
 setter, 145, 195  
 siatka, 139  
 skrót, 82  
 skróty klawiaturowe, 43  
   Alt+Enter, 46  
   Alt+F12, 45  
   Alt+F7, 46  
   Alt+Insert, 45, 145  
   Ctrl+[, 45  
   Ctrl+E, 46  
   Ctrl+F12, 43  
   Ctrl+Shift+[, 45  
   Ctrl+Shift+I, 47  
   Ctrl+Shift+N, 52  
   Ctrl+Spacja, 38, 57  
   Shift+Alt+F, 46, 60  
   Shift+Alt+O, 46  
   Shift+F6, 64

skrypt create\_populate\_tables.sql, 202  
 skryptlety JSP, 71  
 SOAP, Simple Object Access Protocol, 285  
 sprawdzanie poprawności  
   danych, 148  
   ziaren, 208  
 SQL-owe znaczniki, 109  
 stereotyp CDI, 260  
 sterownik JDBC, 31, 189  
 stosowanie kwalifikatorów, 258  
 strona  
   błędu logowania, 84  
   confirmation.xhtml, 255  
   do wprowadzania danych, 57  
   domyślna, 38  
   logowania, 92  
   startowa aplikacji, 218  
   wyświetlająca dane, 66  
   z potwierdzeniem, 146, 172, 260  
 strony JSP, 56  
 system  
   JavaDB, 189, 190  
   RDBMS, 27, 31, 189  
 szablony, 152, 154  
 szablony kodu, 41, 42  
 szkielet projektu JSF, 135

## Ś

ścieżka resources, 319

## T

technologia JSP, 51  
 testowanie usługi sieciowej, 291, 314–317  
 Tomcat, 22  
 transakcje, 238  
 tryb potwierdzania, 279, 283  
 tworzenie  
   aplikacji, 166  
   aplikacji JSF, 132, 169  
   aplikacji sieciowych, 51, 131  
   bazy danych, 201  
   deskryptora wdrażania, 89  
   docelowej lokalizacji JMS, 272  
   encji JPA, 186



- fragmentu JSP, 93, 94
- klas kontrolerów, 195
- klasy encji, 186
- klasy interceptora, 240
- klienta szablonu, 155
- klienta usług sieciowych, 293, 295, 319
- kolejki komunikatów, 273
- komponentów złożonych, 160
- kwalifikatora, 256
- obiektów DAO, 195
- pliku ziarna, 280
- pliku znacznika JSP, 122
- połączenia JMS, 279
- projektu Enterprise Application, 225
- projektu Java Class Library, 227
- puli połączeń, 189
- serwletów, 72
- stereotypu, 261
- strony wyświetlającej dane, 66
- strony z potwierdzeniem, 146
- szablonów facetów, 152
- usług sieciowych, 285, 298, 303, 308
- ziarna, 76
- ziarna sesyjnego, 224, 228
- ziarna zarządzanego, 143
- źródła danych, 189

typy do wiązania interceptorów, 263

## U

- udostępnianie
  - obiektu POJO, 298
  - ziaren EJB, 298, 301
- UEL, Unified Expression Language, 252
- URI, Uniform Resource Identifier, 307
- URI, Unique Resource Identifier, 98
- uruchamianie
  - aplikacji, 147
  - klienta, 237
  - projektu, 55
  - środowiska, 26
- usługi sieciowe
  - RESTful, 307, 311
  - SOAP, 285
- usługi zegara, 243

- ustawianie
  - pakietu, 309
  - właściwości pola wyboru, 63
  - właściwości przycisku, 64
  - właściwości serwletu, 73
  - właściwości strony JSP, 67
- usuwanie
  - danych z bazy, 119
  - operacji usługi, 289
- utrwalanie danych, 191, 194
- uwierzytelnianie użytkownika
  - certyfikat po stronie klienta, 82
  - oparte na formularzu, 82–85
  - podstawowe, 81
  - z wykorzystaniem skrótu, 82
- uzupełnianie
  - kodu, 38, 40
  - nazw, 38, 41
- używanie szablonów, 154

## W

- walidator niestandardowy, 150
- wartości adnotacji @TransactionAttribute, 238
- wiązanie interceptorów, 263, 265
- widok, 72
- właściwości
  - kolejki komunikatów, 274
  - pola wyboru, 63
  - przycisku, 64
  - serwletu, 73
  - strony JSP, 67
- wprowadzanie danych, 57
- wskazówki wizualne, 47
- wstawianie
  - formularzy, 59
  - tabeli, 60
- wybieranie
  - encji, 216, 245
  - klasy encji, 196
  - tabel, 204, 309
  - usługi dla klienta, 320
- wyciek pamięci, 338
- wydajne programowanie, 38, 335
- wyjątki
  - JMSEException, 282
  - ValidatorException, 151

wysyłanie komunikatów, 276

wyświetlanie

części strony, 98

stron, 88

wzorzec projektowy

DAO, 195

Fasada, 311

MVC, 72

## Z

zabezpieczanie aplikacji sieciowych, 81

zakładka Adres, 177

zakładki, 173, 176

załącznik komunikatu, 290

zarządzanie transakcjami, 238

zasięg ziarna, 79, 254

zasób REST, 310

zegar, 243

ziarno

CDI, 252

EJB, 223, 247, 298

encyjne, 185, 223

JavaBeans, 76

sesyjne, 223, 228

bezstanowe, 224

stanowe, 224

sterowane komunikatami, 242, 271, 280, 282

zarządzane, 143, 145

addressBean, 163

CustomerController, 180

zintegrowane środowisko programistyczne, IDE, 15

zmienna środowiskowa JAVA\_HOME, 22

znacznik

<c:choose>, 98, 101

<c:forEach>, 98, 104

<c:if>, 98

<cc:attribute>, 162

<cc:implementation>, 162

<class>, 265

<f:validator>, 151

<h:commandButton>, 142

<h:form>, 138, 178

<h:graphicImage>, 141

<h:label>, 141

<h:message>, 141

<h:outputLabel>, 141

<h:panelGrid>, 139

<h:panelGroup>, 142

<h:selectOneMenu>, 141

<interceptor>, 265

<p:calendar>, 171

<p:commandButton>, 167

<p:dialog>, 167

<p:inputMask>, 177

<p:messages>, 170

<p:tabView>, 173

<p:wizard>, 178

<sql:param>, 116

<sql:query>, 109

<sql:update>, 114, 117

<ui:composition>, 156

<ui:define>, 156

<ui:insert>, 156

znaczniki

JSTL, 98

niestandardowe JSP, 122

PrimeFaces, 167

SQL-owe, 107

## Ż

źródło danych, 111, 188

## Ż

żądanie GET, 78

żądanie POST, 78

# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

# Java EE 6

## Tworzenie aplikacji w NetBeans 7

Java EE to zbiór zaawansowanych technologii, pozwalających stworzyć nowoczesną aplikację o doskonałej architekturze. Jeżeli żądasz najwyższej wydajności, niezawodności, jakości oraz elastyczności, to Java EE jest prawdopodobnie najlepszym wyborem. Jeżeli chcesz się skupić na postawionym zadaniu i nie masz ochoty zmagać się z problemami z konfiguracją czy środowiskiem pracy, wybierz lidera wśród darmowych IDE. NetBeans zapewnia pełne wsparcie dla Javy EE, a praca z nim to czysta przyjemność.

W trakcie lektury tej wyjątkowej książki poznasz najlepsze techniki pracy z tym duetem. Zaznajomisz się z możliwościami Javy EE oraz przekonasz się, jak bardzo NetBeans może ułatwić Ci pracę. Na wstępie dowiesz się, jak zainstalować, dostosować do własnych potrzeb oraz przygotować do pracy środowisko NetBeans. Na kolejnych stronach zdobędziesz wiedzę związaną z serwetami, JSP oraz JSF. Ponadto nauczysz się korzystać z Java Persistence API oraz projektować i uruchamiać usługi sieciowe. Książka ta jest doskonałym źródłem wiedzy dla wszystkich programistów języka Java, chcących wykorzystać potencjał NetBeans w tworzeniu zaawansowanych aplikacji.

**Poznaj najlepsze techniki pracy z NetBeans!**

**helion.pl**  
księgarnia  
internetowa

Nr katalogowy: 18834

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/novosci>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

Informatyka w najlepszym wydaniu



Dzięki tej książce:

- poznasz możliwości Javy EE
- dostosujesz NetBeans do własnych preferencji
- wykorzystasz w pełni wsparcie NetBeans dla Javy EE
- zwiększysz swoją produktywność!

sięgnij po WIĘCEJ



KOD KORZYSCI

ISBN 978-83-246-8936-1



9 788324 689361

Cena: 59,00 zł