

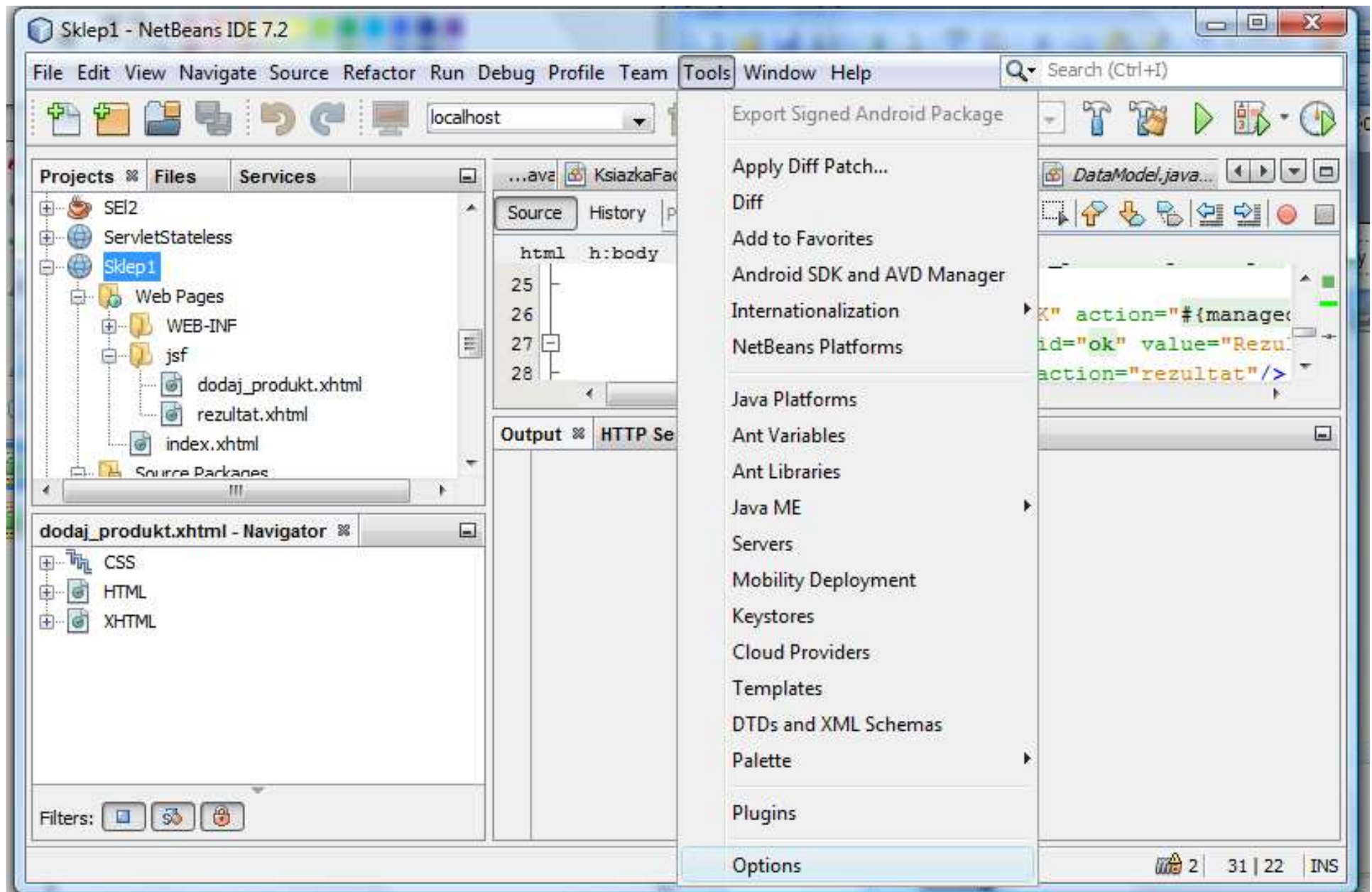
Budowa prostej aplikacji wielowarstwowej

Laboratorium 1

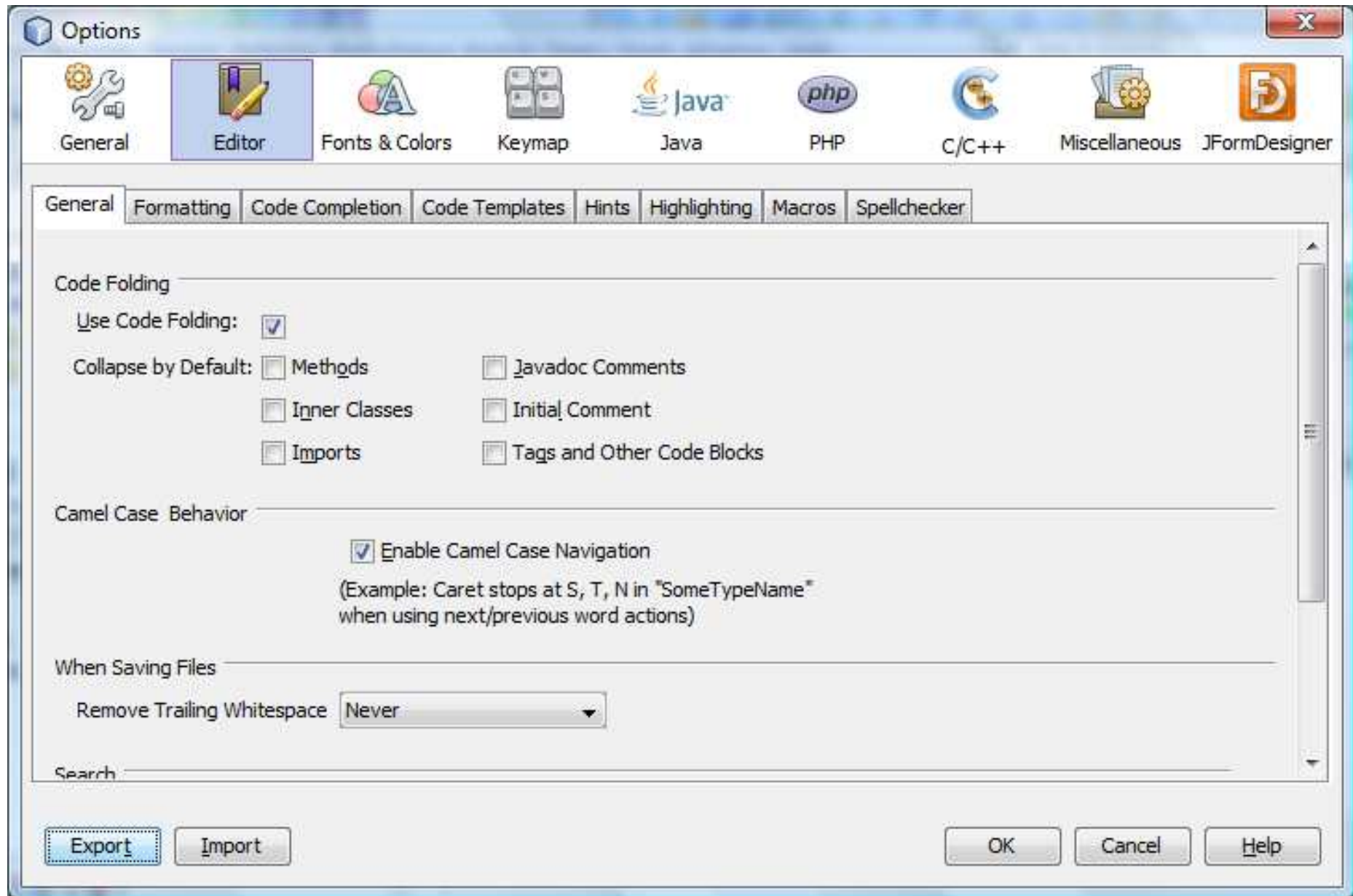
Programowanie komponentowe

Zofia Kruczkiewicz

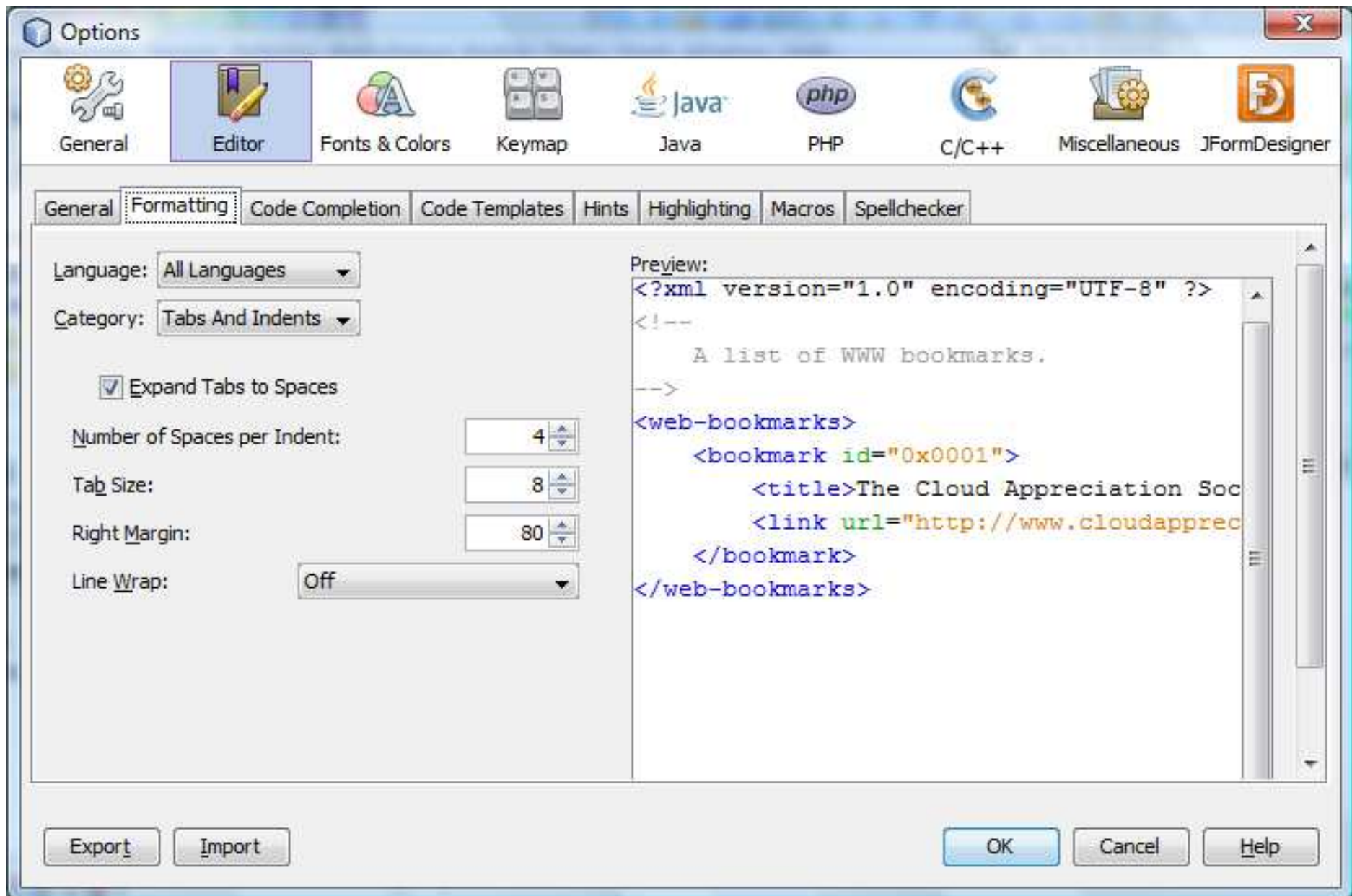
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



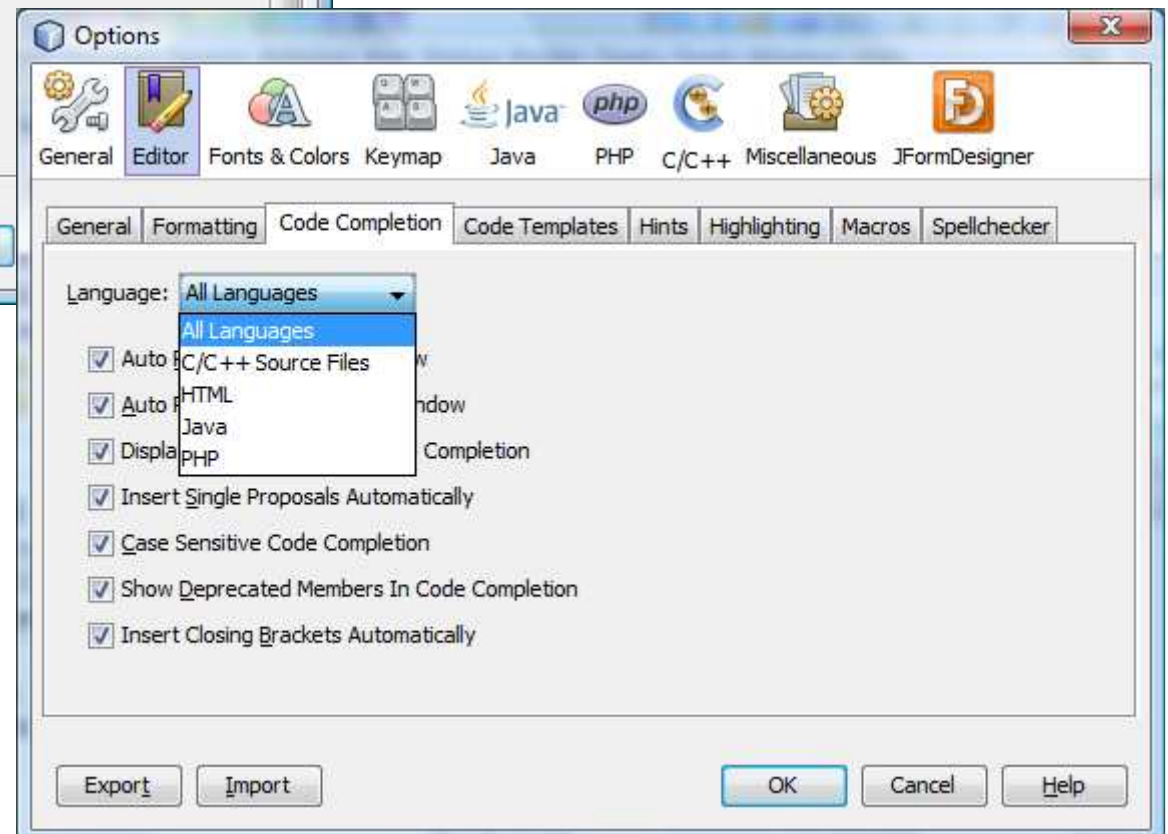
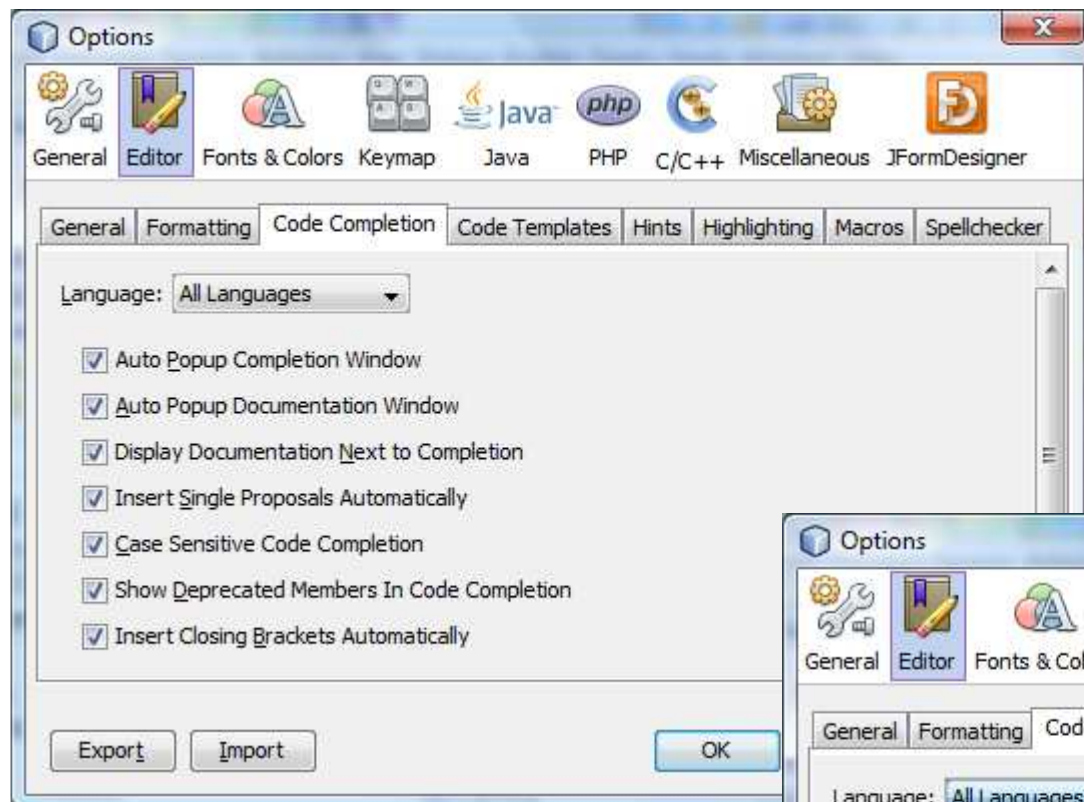
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



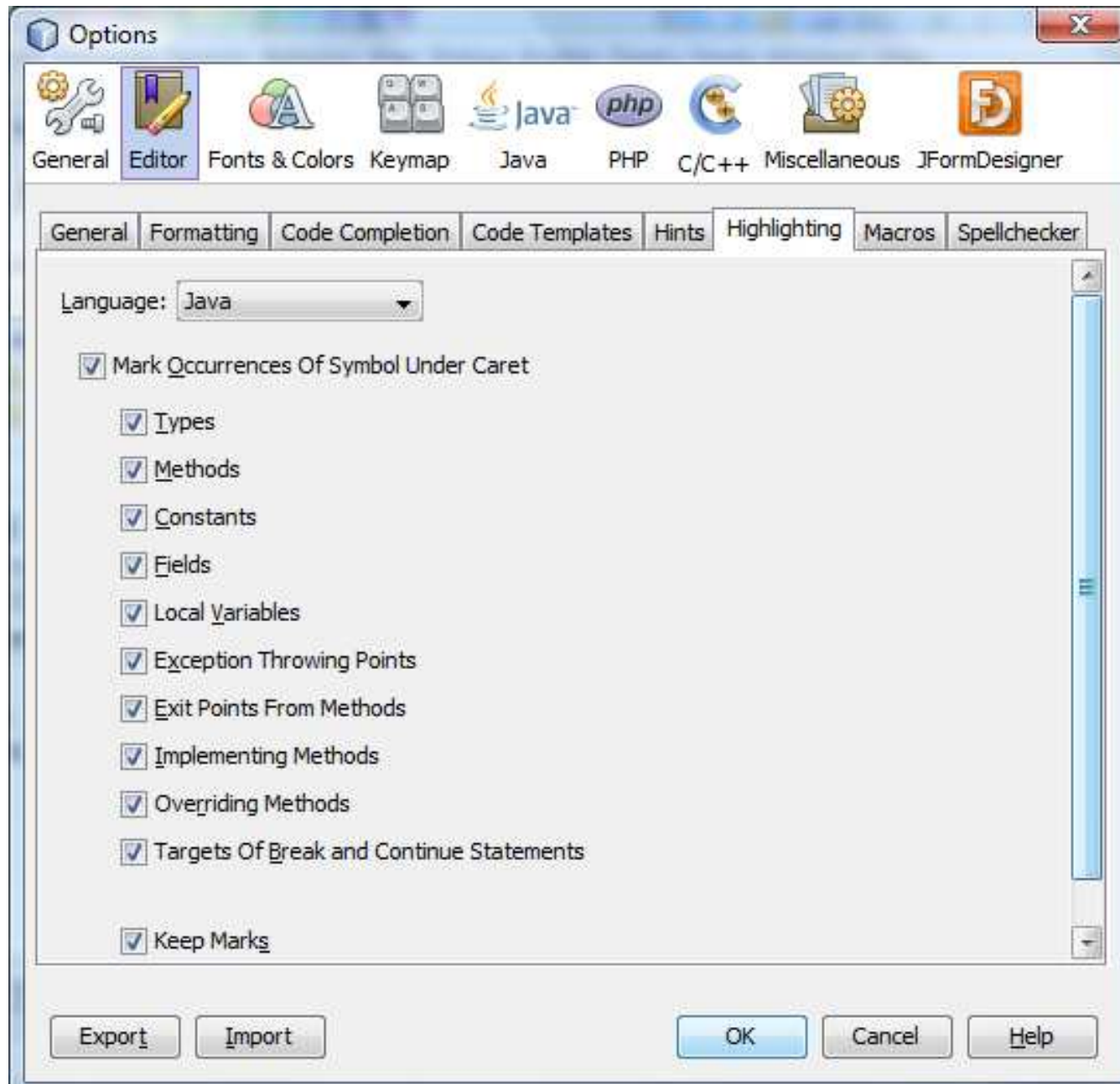
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



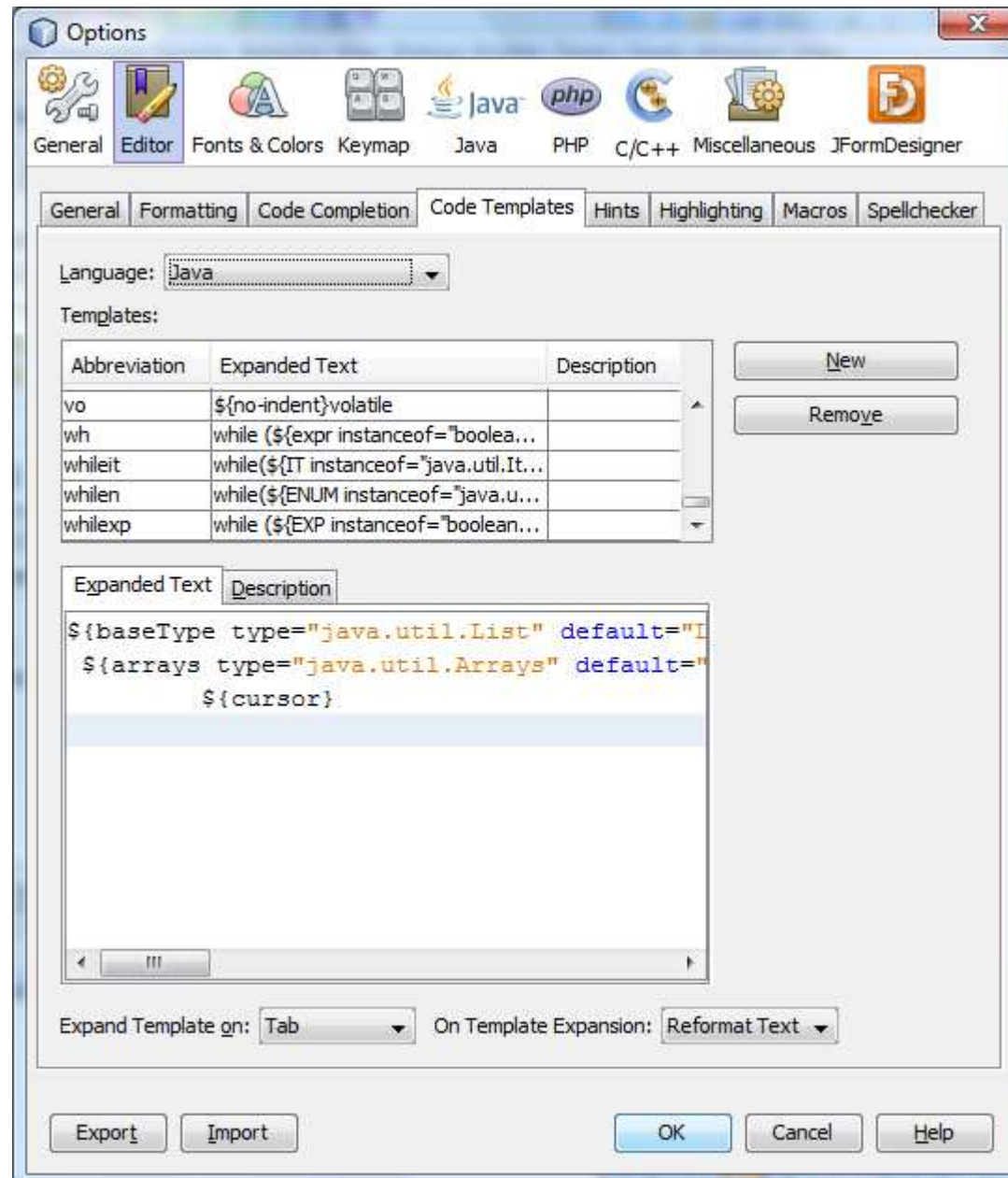
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



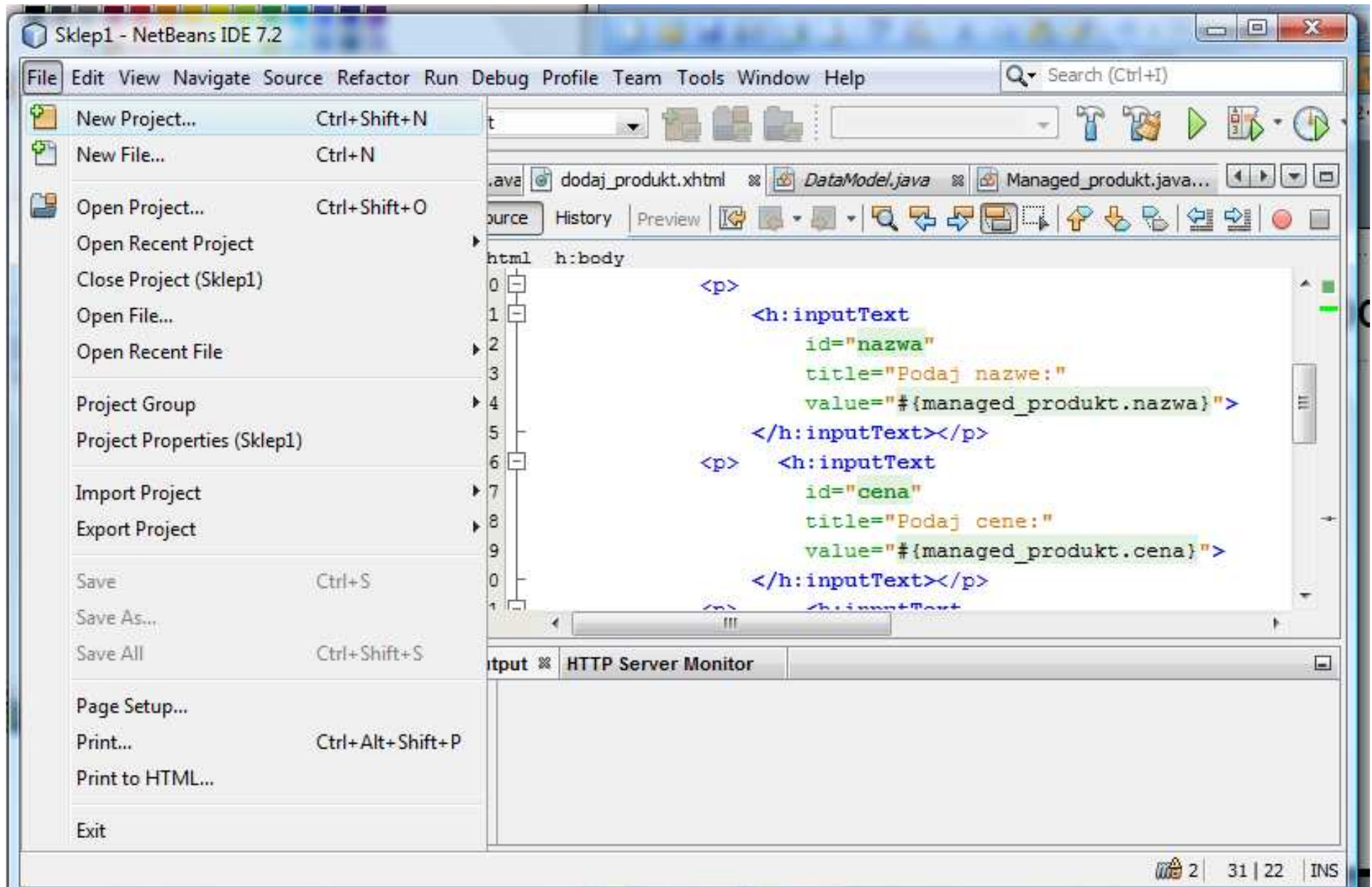
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



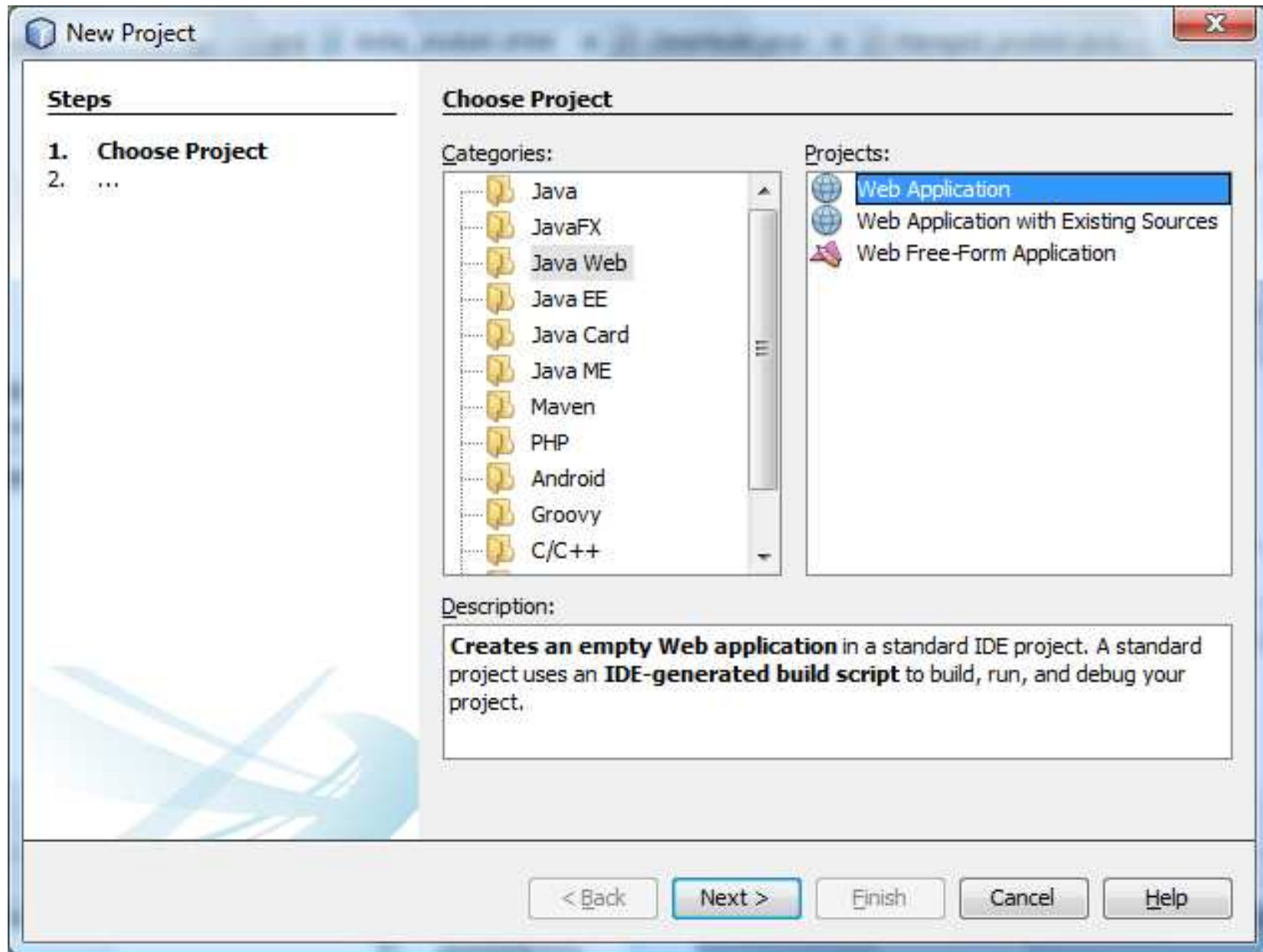
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



Zakładanie projektu typu Web Application (**File/New Project**)



Zakładanie projektu typu Web Application (Java Web/Web Application i Next)



Zakładanie projektu typu Web Application: **Project Name: Sklep_1**, należy wybrać Project Location

New Web Application

Steps

1. Choose Project
- 2. Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name: Sklep_1

Project Location: E:\JSF\JavaPK

Project Folder: E:\JSF\JavaPK\Sklep_1

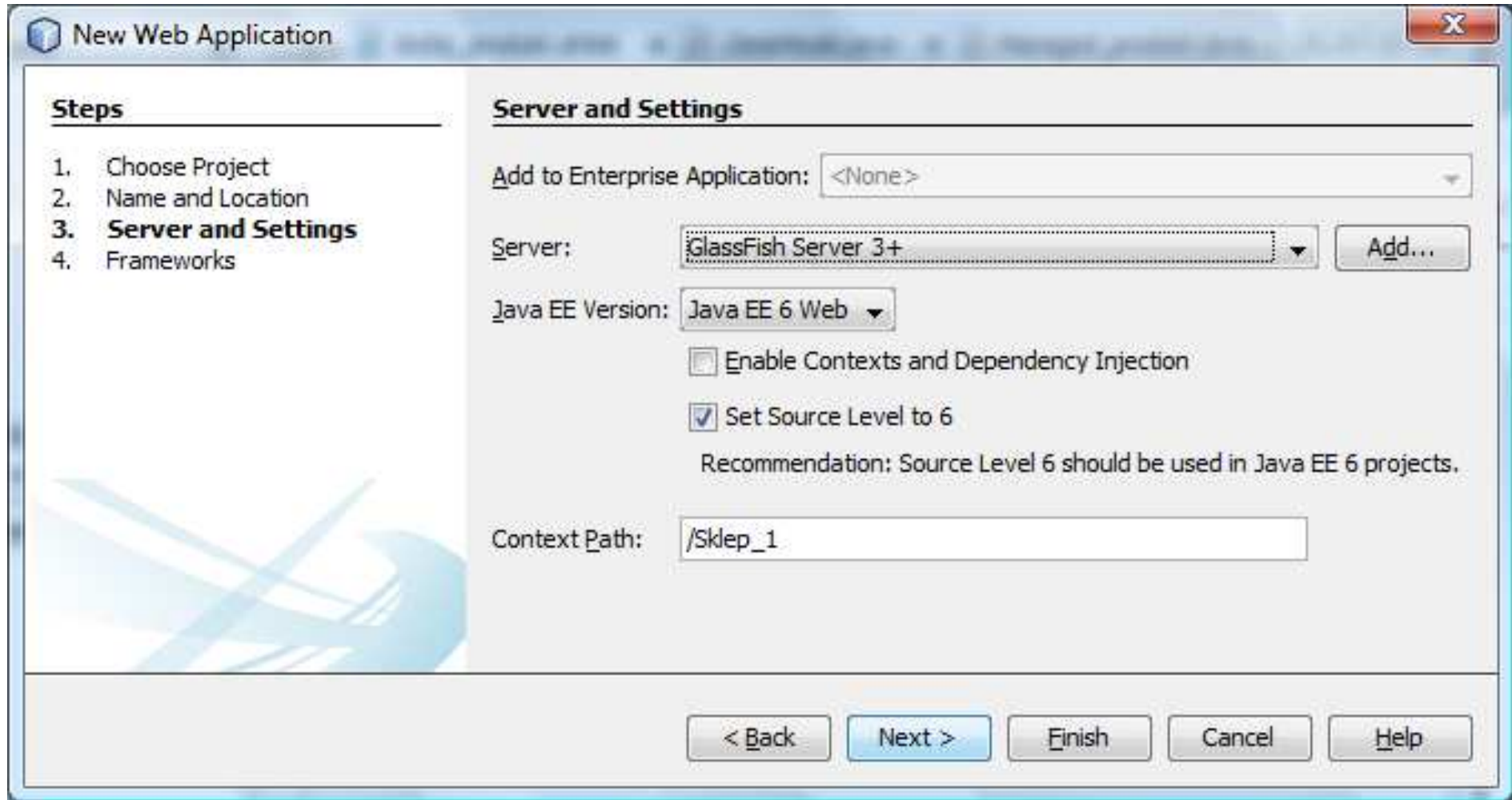
Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help)

< Back **Next >** Finish Cancel Help

Zakładanie projektu typu Web Application: **Server: GlassFish 3+, Java EE
Version: Java EE 6 Web i Next**

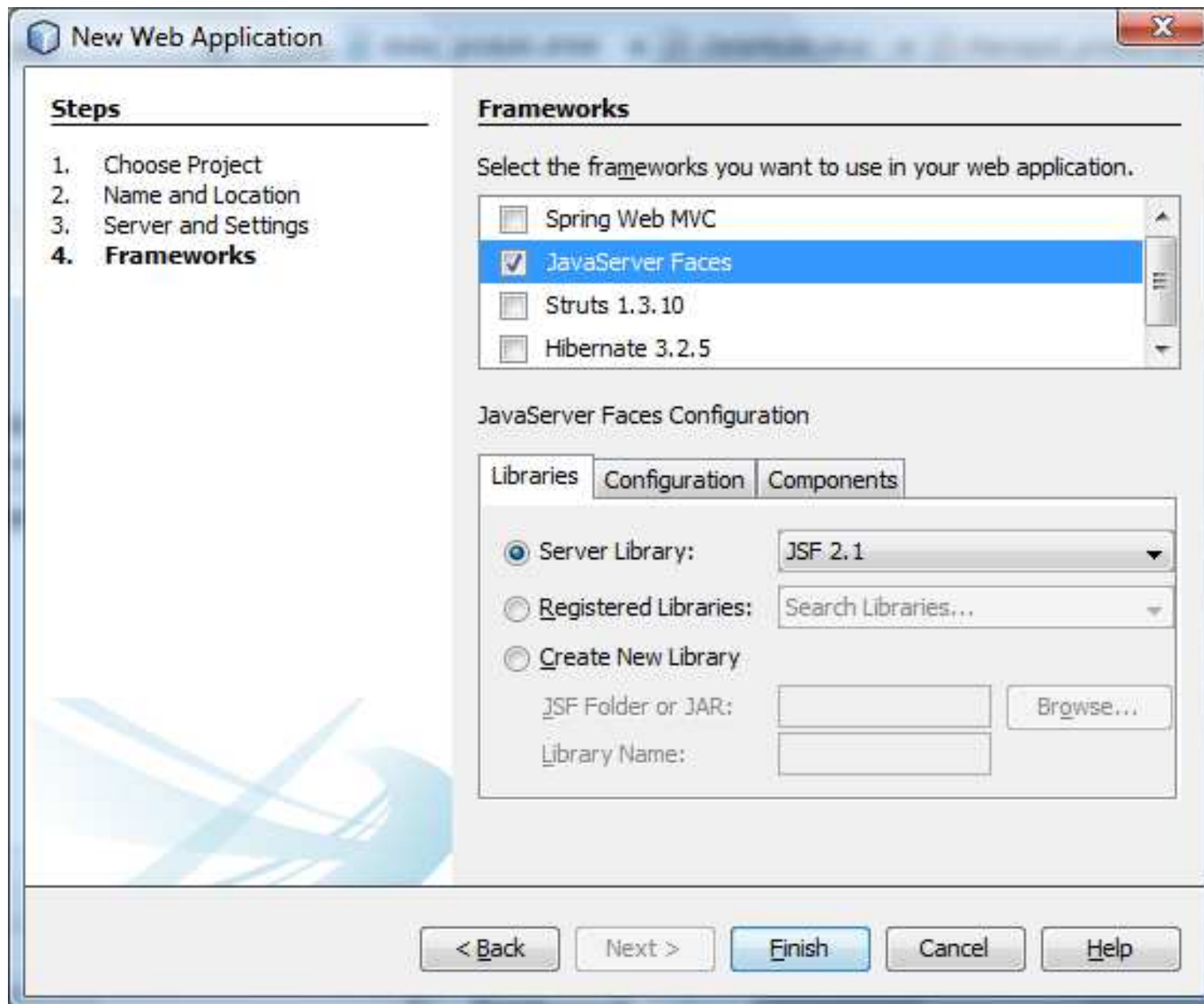


The screenshot shows the 'New Web Application' wizard in an IDE. The window title is 'New Web Application'. On the left, a 'Steps' pane lists four steps: 1. Choose Project, 2. Name and Location, 3. **Server and Settings** (the current step), and 4. Frameworks. The main area is titled 'Server and Settings' and contains the following configuration options:

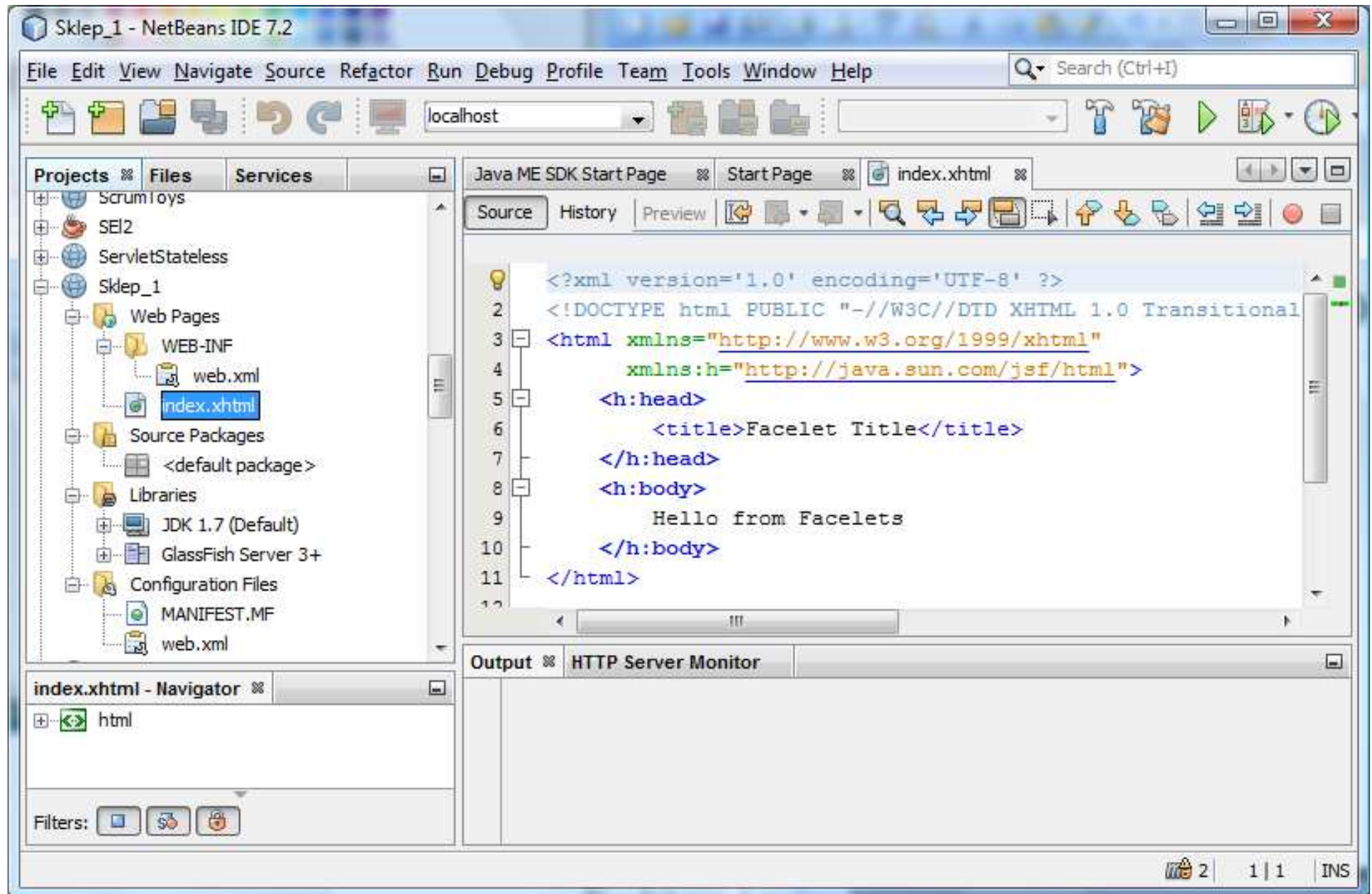
- 'Add to Enterprise Application:' dropdown menu with '<None>' selected.
- 'Server:' dropdown menu with 'GlassFish Server 3+' selected and an 'Add...' button to its right.
- 'Java EE Version:' dropdown menu with 'Java EE 6 Web' selected.
- Checkbox for 'Enable Contexts and Dependency Injection' (unchecked).
- Checkbox for 'Set Source Level to 6' (checked).
- Text below the checked checkbox: 'Recommendation: Source Level 6 should be used in Java EE 6 projects.'
- 'Context Path:' text box containing '/Sklep_1'.

At the bottom of the wizard, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

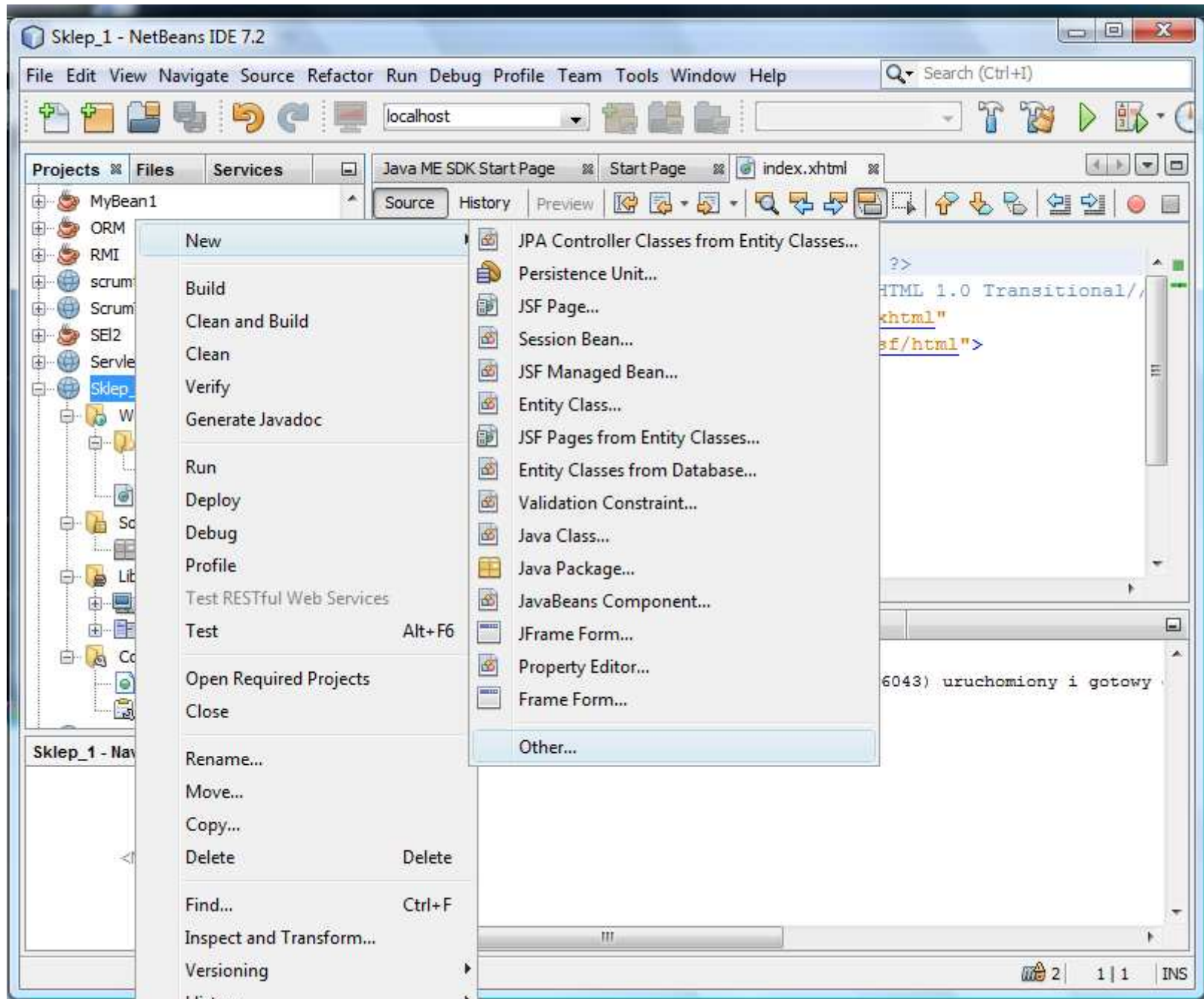
Zakładanie projektu typu Web Application- zaznaczyć checkbox **JavaServer Faces** i **Finish**



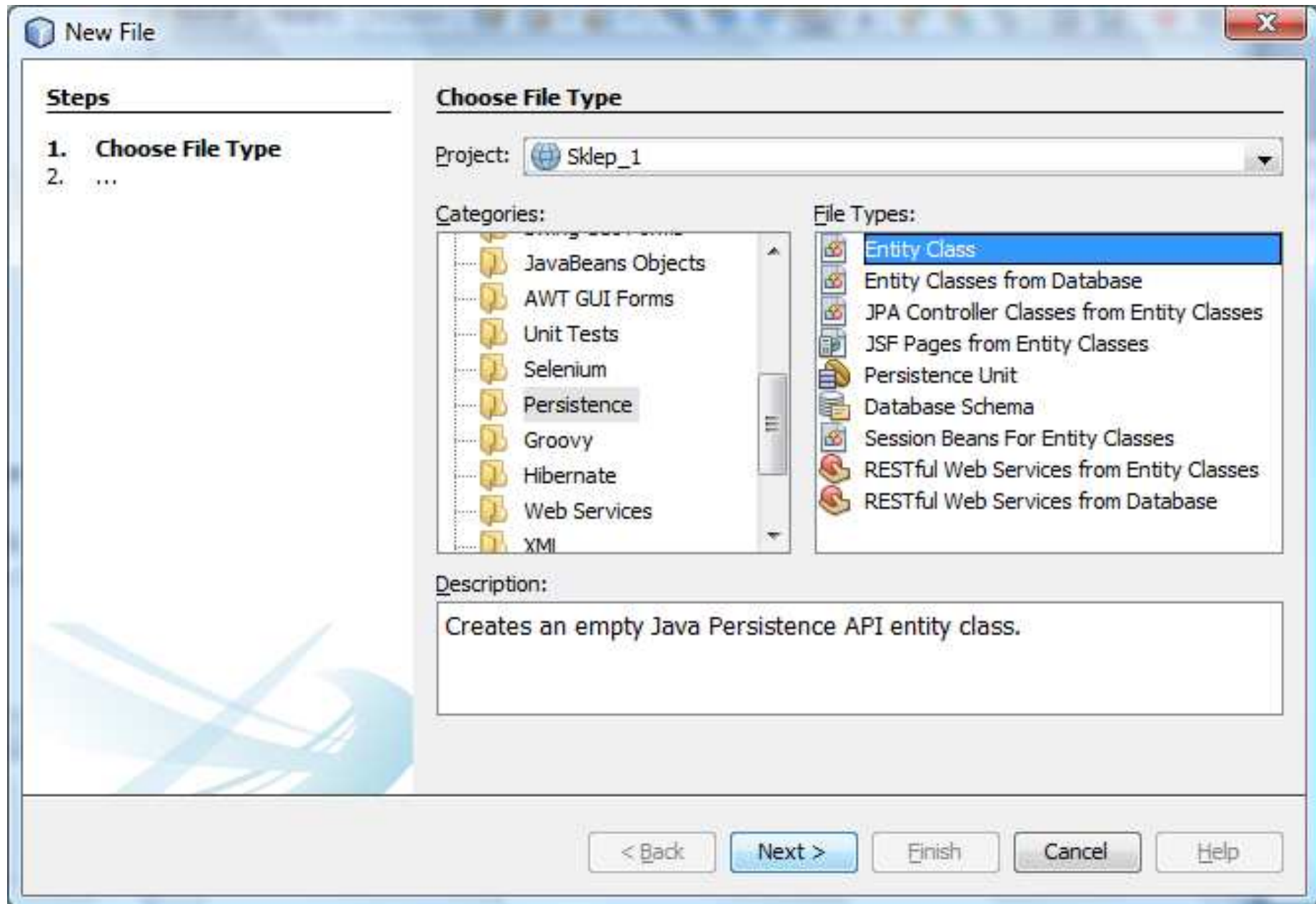
Zakładanie projektu typu Web Application – projekt ze stroną startową **index.xhtml**



Dodanie klasy typu **Entity** przechowującej dane produktu i wyznaczającej cenę brutto (**New/Other**)



Dodanie klasy typu **Entity** przechowującej dane produktu i wyznaczającej cenę brutto – **Persistence/ Entity Class i Next**



Dodanie klasy typu **Entity** przechowującej dane produktu i wyznaczającej cenę brutto-
Class Name: Produkt, Package: jpa, usunąć zaznaczenie **Create Persistence Unit**

New Entity Class

Steps

1. Choose File Type
- 2. Name and Location**
3. Provider and Database

Name and Location

Class Name: Produkt

Project: Sklep_1

Location: Source Packages

Package: jpa

Created File: E:\JSF\JavaPK\Sklep_1\src\java\jpa\Produkt.java

Primary Key Type: Long

The project does not have a persistence unit. You need a persistence unit to persist entity classes.

Create Persistence Unit

< Back Next > **Finish** Cancel Help

Wygenerowania klasa o nazwie Produkt typu Entity, reprezentująca obiektowy model danych aplikacji EE

The screenshot displays the NetBeans IDE 7.2 interface. The main editor window shows the source code for the file `Produkt.java`. The code defines a JPA Entity class named `Produkt` that implements the `Serializable` interface. The class includes a `serialVersionUID`, an `@Id` annotated `private Long id` field, and methods `getId()` and `setId(Long id)`.

```
1  /*...*/
5  package jpa;
6
7  import java.io.Serializable;
8  import javax.persistence.Entity;
9  import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12
13 /*...*/
17 @Entity
18 public class Produkt implements Serializable {
19     private static final long serialVersionUID = 1L;
20     @Id
21     @GeneratedValue(strategy = GenerationType.AUTO)
22     private Long id;
23
24     public Long getId() {
25         return id;
26     }
27
28     public void setId(Long id) {
```

The interface also shows the 'Projects' view on the left, with the project structure for 'Sklep_1' expanded to show the 'jpa' package containing 'Produkt.java'. The 'Members View' for 'Produkt' shows the `getId() : Long` method. The 'Output' window at the bottom shows the status of the Java DB Database Process and the HTTP Server Monitor.

Wygenerowany kod klasy **Produkt**

```
package jpa;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Produkt implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
}
```

@Override

```
public int hashCode() {  
    int hash = 0;  
    hash += (id != null ? id.hashCode() : 0);  
    return hash;  
}
```

@Override

```
public boolean equals(Object object) {  
    // TODO: Warning - this method won't work in the case the id fields are not set  
    if (!(object instanceof Produkt)) {  
        return false;  
    }  
    Produkt other = (Produkt) object;  
    if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {  
        return false;  
    }  
    return true;  
}
```

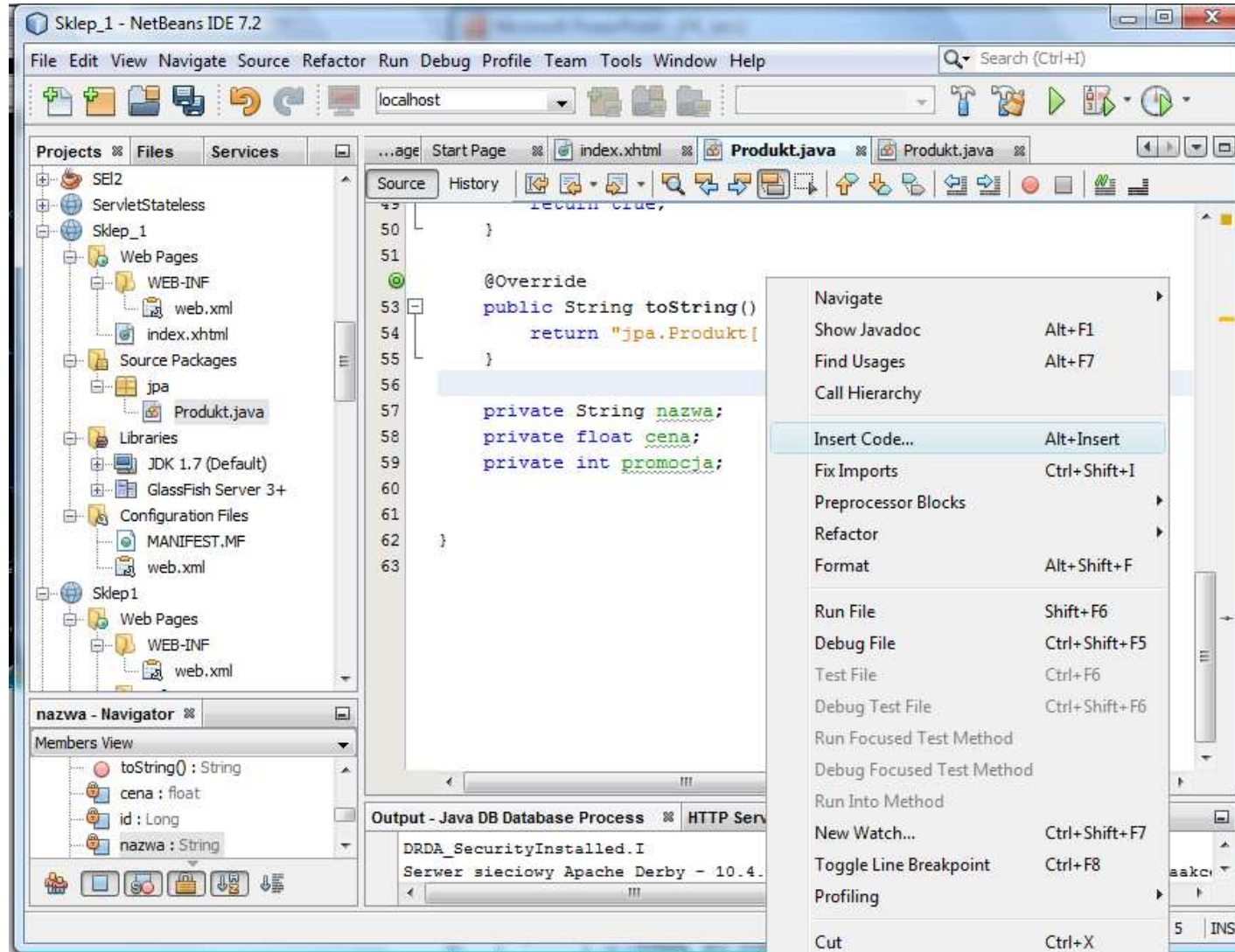
@Override

```
public String toString() {  
    return "jpa.Produkt[ id=" + id + " ]";  
}
```

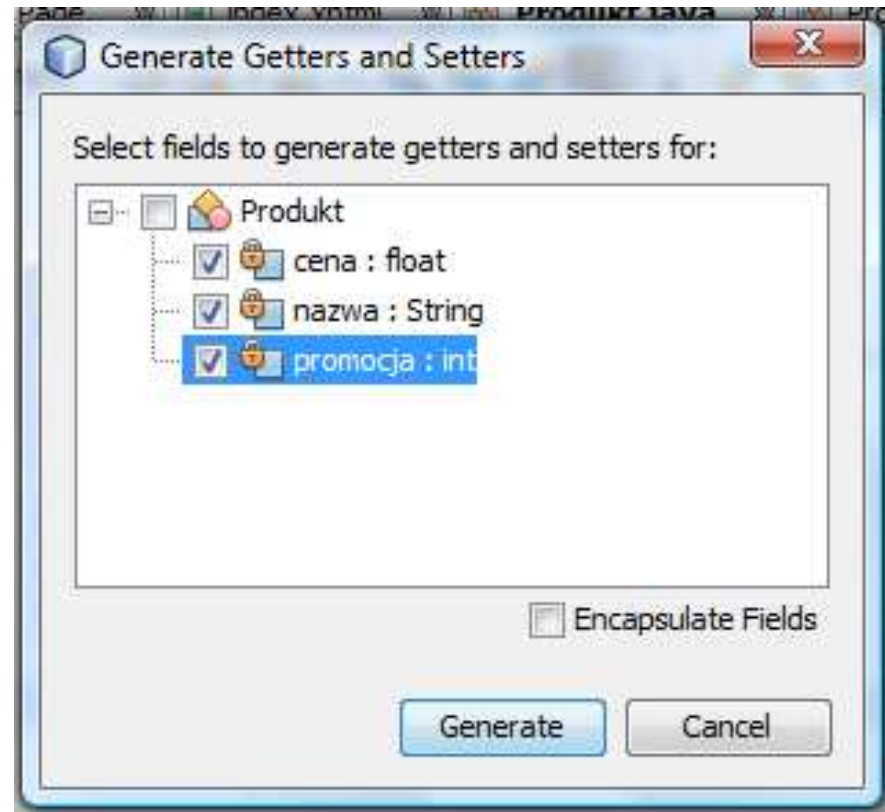
```
}
```

private String nazwa;
private float cena;
private int promocja;

Uzupełnienie kodu klasy Produkt - należy dodać atrybuty podane z lewej strony i po kliknięciu prawym klawiszem myszy na okno edytora wybrać pozycję **Insert Code/ Getter and Setter...**



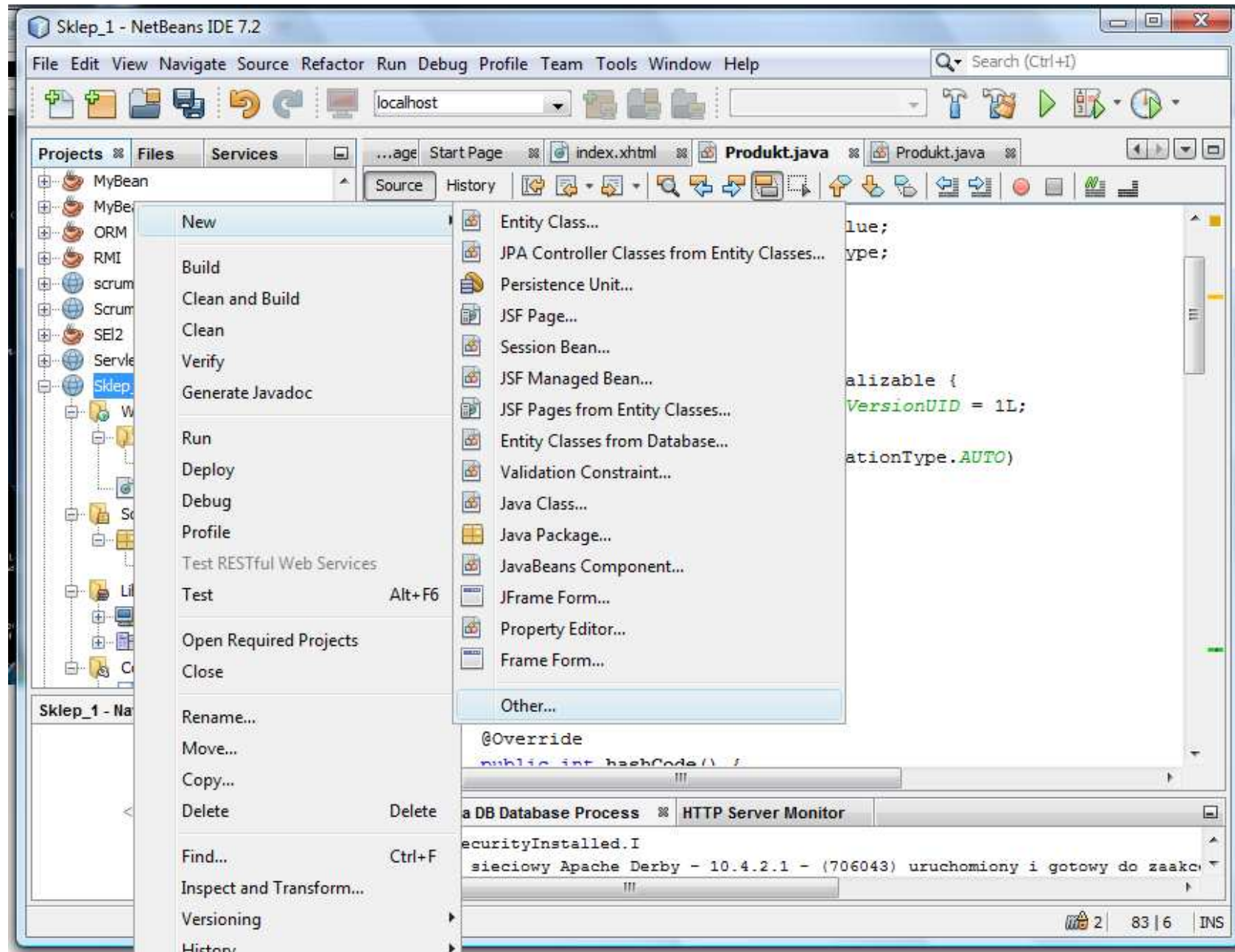
Zaznaczyć atrybuty do wygenerowania metod dostępu do atrybutów



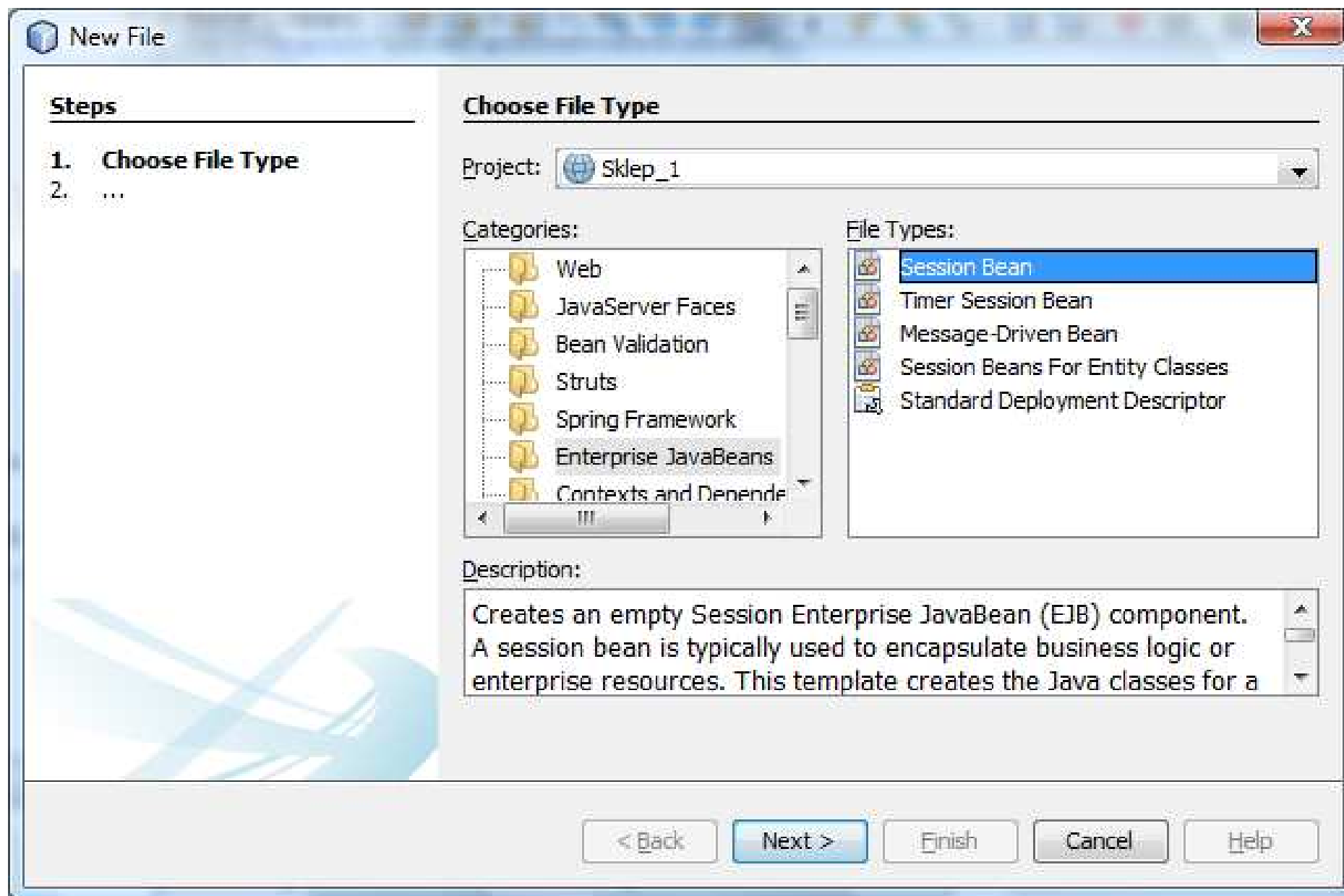
Oprócz wygenerowanych metod dodano metodę **cena_brutto()** wyznaczającą cenę brutto na podstawie ceny netto i promocji

```
public String getNazwa() {
    return nazwa;
}
public void setNazwa(String nazwa) {
    this.nazwa = nazwa;
}
public float getCena() {
    return cena;
}
public void setCena(float cena) {
    this.cena = cena;
}
public int getPromocja() {
    return promocja;
}
public void setPromocja(int promocja) {
    this.promocja = promocja;
}
public float cena_brutto ()
{
    float cena_brutto= cena*(1-(float)promocja/100);
    return cena_brutto;
}
```

Należy dodać ziarno EJB do przetwarzania obiektu typu Entity (Produkt) –
New/Other



Należy dodać ziarno EJB do przetwarzania obiektu typu Entity – **Enterprise JavaBean/ Session Bean i Next**



Należy dodać ziarno EJB do przetwarzania obiektu typu Entity –
**EJB Name: Fasada_warstwy_biznesowej, Package:
Warstwa_biznesowa**

New Session Bean

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

EJB Name: Fasada_warstwy_biznesowej

Project: Sklep_1

Location: Source Packages

Package: Warstwa_biznesowa

Session Type:

Stateless

Stateful

Singleton

Create Interface:

Local

Remote

< Back Next > **Finish** Cancel Help

Kod wygenerowany ziarna EJB do przetwarzania obiektu typu Entity

The screenshot displays the NetBeans IDE 7.2 interface. The main editor window shows the source code for the class `Fasada_warstwy_biznesowej.java`. The code is as follows:

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5  package Warstwa_biznesowa;
6
7  import javax.ejb.Stateless;
8
9  /**
10   *
11   * @author kruczkiewicz
12   */
13  @Stateless
14  public class Fasada_warstwy_biznesowej {
15
16      // Add business logic below. (Right-click in editor a
17      // "Insert Code > Add Business Method")
18
19  }
20
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar with various icons, and a project explorer on the left showing the project structure. The project explorer highlights the `Fasada_warstwy_biznesowej.java` file. The bottom status bar shows the page number 2, line 1, and column 1.

Uzupełniono kod klasy **Fasada_warstwy_biznesowej**: dodano atrybut produkt typu Produkt, metody typu set i get (za pomocą opcji **Insert Code** - (slajdy 20-21)) oraz metody (następny slajd): **utworz_produkt**, która tworzy produkt nadając mu dane pobrane z tablicy **dane** przekazanej do metody oraz **dane_produktu**, która zwraca atrybuty produktu oraz wartość ceny brutto w postaci tablicy elementów typu String

```
package Warstwa_biznesowa;
```

```
import javax.ejb.Stateless;
```

```
import jpa.Produkt;
```

```
@Stateless
```

```
public class Fasada_warstwy_biznesowej {
```

```
    // Add business logic below. (Right-click in editor and choose  
    // "Insert Code > Add Business Method")
```

```
    private Produkt produkt;
```

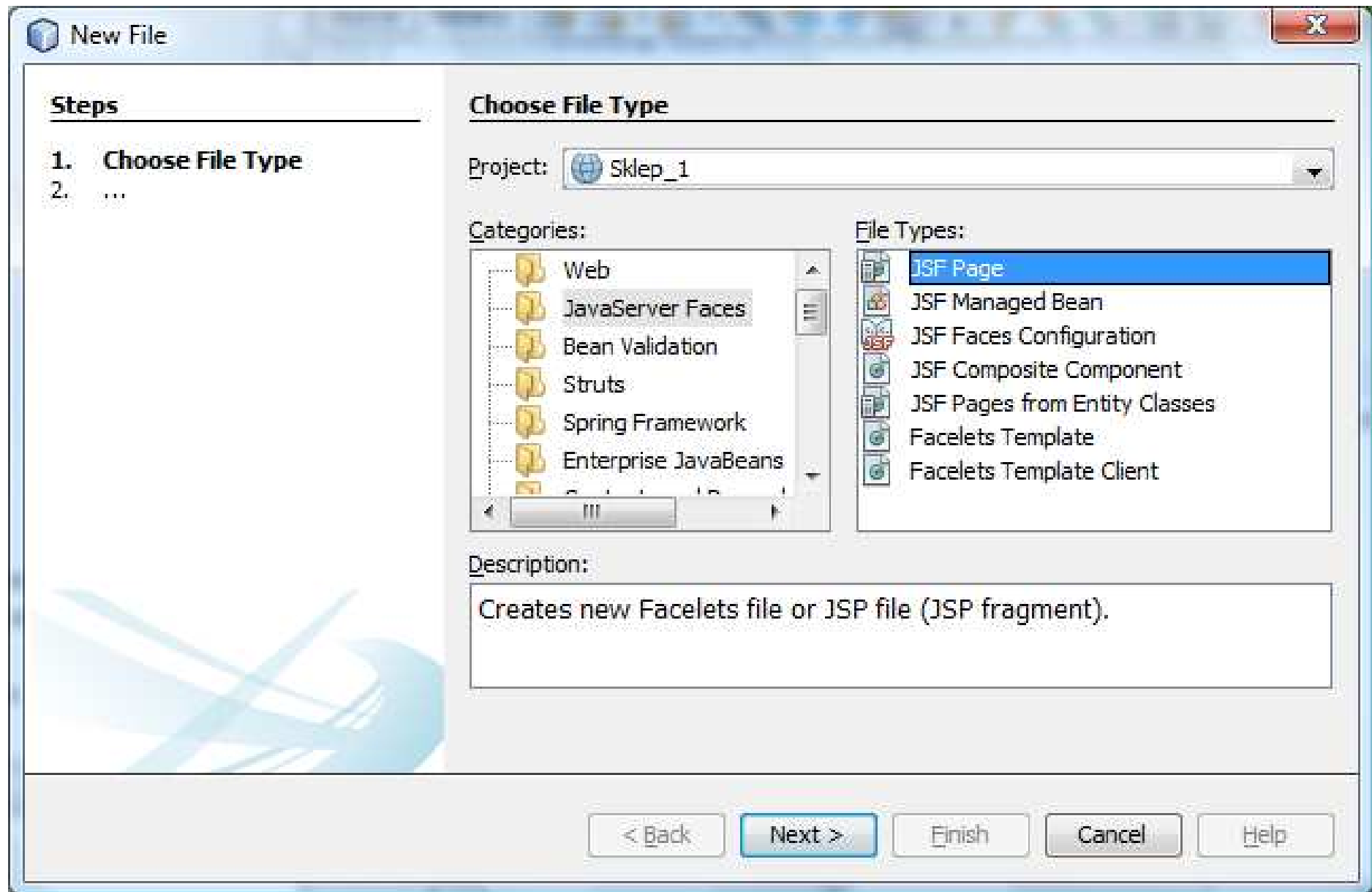
```
    public Produkt getProdukt() {  
        return produkt;  
    }  
}
```

```
    public void setProdukt(Produkt produkt) {  
        this.produkt = produkt;  
    }  
}
```

```
public void utworz_produkt(String dane[]) {  
    produkt = new Produkt();  
    produkt.setNazwa(dane[0]);  
    produkt.setCena(Float.parseFloat(dane[1]));  
    produkt.setPromocja(Integer.parseInt(dane[2]));  
}
```

```
public String[] dane_produktu() {  
    String nazwa = "brak produktu";  
    String cena = "brak produktu";  
    String promocja ="brak produktu";  
    String cena_brutto="brak produktu";  
    if (produkt != null) {  
        nazwa = produkt.getNazwa();  
        cena=""+produkt.getCena();  
        promocja=""+produkt.getPromocja();  
        cena_brutto=""+produkt.cena_brutto();  
    }  
    String dane[] = {nazwa, cena, promocja, cena_brutto};  
    return dane;  
}
```

Dodanie strony typu JavaServer Faces do wstawiania danych produktu:
New/Other/JavaServer Faces/JSF Page



Dodanie strony typu JavaServer Faces do wstawiania danych produktu: **File Name:** dodaj_produk, **Folder:** jsf i **Finish**

New JSF Page

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

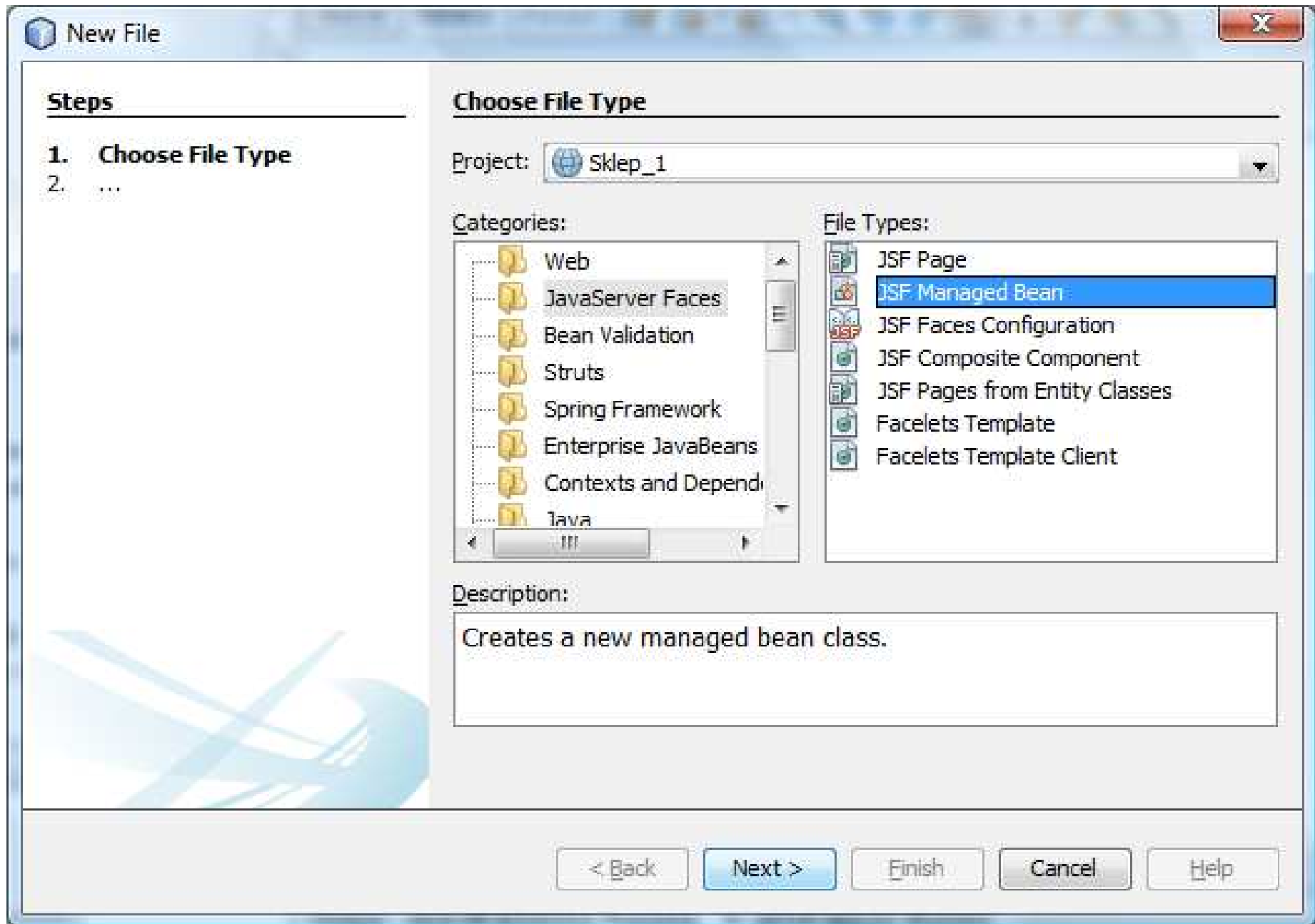
Options:

Facelets

JSP File (Standard Syntax) Create as a JSP Segment

Description:

Dodanie klasy typu Managed Bean: **New/Other/JavaServer Faces/JSF Managed Bean** i Next



Dodanie klasy typu Managed Bean: **Name: Managed_produk**t, **Package: jsf**,
Scope: request i **Finish**

Steps

1. Choose File Type
- 2. Name and Location**

Name and Location

Class Name: Managed_produk

Project: Sklep_1

Location: Source Packages

Package: jsf

Created File: E:\JSF\JavaPK\Sklep_1\src\java\jsf\Managed_produk.java

Add data to configuration file

Configuration File:

Name: managed_produk

Scope: request

Bean Description:

< Back Next > **Finish** Cancel Help

Wygenerowany kod klasy Managed Bean

The screenshot displays the NetBeans IDE 7.2 interface. The main editor window shows the source code for the `Managed_produkty` class. The code is as follows:

```
4  /*
5  package jsf;
6
7  import javax.faces.bean.ManagedBean;
8  import javax.faces.bean.RequestScoped;
9
10 /**
11  *
12  * @author kruczkiewicz
13  */
14 @ManagedBean
15 @RequestScoped
16 public class Managed_produkty {
17
18     /**
19     * Creates a new instance of Managed_produkty
20     */
21     public Managed_produkty() {
22     }
23 }
24
```

The left sidebar shows the project structure for `Sklep_1`, with `Managed_produkty.java` selected under the `jsf` source package. The bottom status bar indicates the IDE is running on a local host, with 2 open files, 1 line selected, and the cursor at the end of the line (INS).

```

package jsf;

import Warstwa_biznesowa.Fasada_warstwy_biznesowej;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
@ManagedBean
@RequestScoped
public class Managed_produk {

    @EJB
    private Fasada_warstwy_biznesowej fasada;
    private String nazwa;
    private String cena;
    private String promocja;
    private String cena_brutto;

    public Managed_produk() {
    }
    public Fasada_warstwy_biznesowej getFasada() {
        return fasada;
    }
    public void setFasada(Fasada_warstwy_biznesowej fasada) {
        this.fasada = fasada;
    }
}

```

Dodany kod do klasy Managed_produk: atrybut **fasada** typu EJB Fasada_warstwy_biznesowej (z adnotacją EJB, ponieważ jest to obiekt typu EJB) oraz atrybuty: nazwa, cena, promocja oraz cena_brutto bindowane z komponentami stron JSF. Metody dostępu do atrybutów dodano za pomocą pozycji **Insert Code** (slajdy 20-21) – następny slajd

```
public String getNazwa() {
    return nazwa;
}
public void setNazwa(String nazwa) {
    this.nazwa = nazwa;
}
public String getCena() {
    return cena;
}
public void setCena(String cena) {
    this.cena = cena;
}
public String getPromocja() {
    return promocja;
}
public void setPromocja(String promocja) {
    this.promocja = promocja;
}
public String getCena_brutto() {
    return cena_brutto;
}
public void setCena_brutto(String cena_brutto) {
    this.cena_brutto = cena_brutto;
}
```

Dodane metod do klasy Managed_produkci obsługujacych dodawanie produktu (**dodaj_produkci**) po pobraniu danych z formularza za pomoca atrybutow: nazwa, cena, promocja i wywolaniu metody **utworz_produkci** ziarna EJB z obiektu fasada klasy typu Fasada_warstwy_biznesowej oraz wyswietlanie danych za pomoca metody **dane_produkci** pobranych z warstwy biznesowej od obiektu typu EJB fasada za pomoca metody **dane_produkci**

```
public String dodaj_produkci() {  
    String[] dane = {nazwa, cena, promocja};  
    fasada.utworz_produkci(dane);  
    dane_produkci();  
    return "rezultat";  
}
```

```
public void dane_produkci()  
{ String[] dane=fasada.dane_produkci();  
    nazwa=dane[0];  
    cena=dane[1];  
    promocja=dane[2];  
    cena_brutto=dane[3];  
}  
}
```

Uzupełniona treść strony **dodaj_produk**t

```
<?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:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    <h:form>
```

```

<h:panelGrid columns="2">
    <h:outputLabel value="Podaj nazwe produktu" for="nazwa" />
    <h:inputText
        id="nazwa"
        title="Podaj nazwe:"
        value="#{managed_produkt.nazwa}"
        required="true"
        requiredMessage="Blad: Podaj nazwe." >
    </h:inputText>
    <h:outputLabel value="Podaj cene netto produktu" for="cena" />
    <h:inputText
        id="cena"
        title="Podaj cene:"
        value="#{managed_produkt.cena}"
        required="true"
        requiredMessage="Blad: Podaj cene." >
    </h:inputText>
    <h:outputLabel value="Podaj promocje produktu" for="promocja" />
    <h:inputText
        id="promocja"
        title="Podaj promocje:"
        value="#{managed_produkt.promocja}"
        required="true"
        requiredMessage="Blad: Podaj promocje." >
    </h:inputText>
</h:panelGrid>

```

Siatka **panelGrid** umożliwia wprowadzanie danych produktu do obiektu typu **Managed_produkt** w dwóch kolumnach za pomocą komponentów **outputLabel** oraz **inputText**.

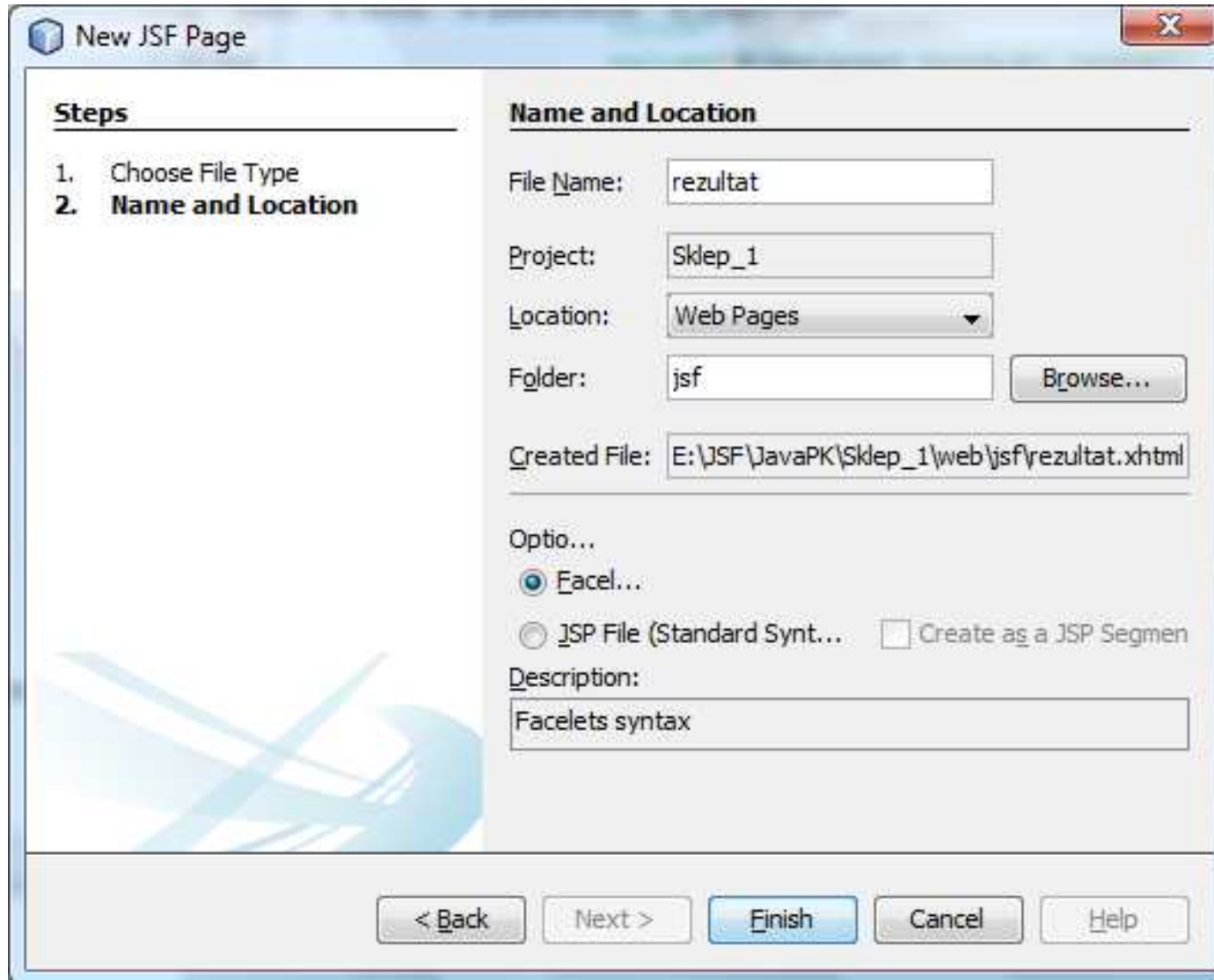
Atrybuty **required** i **requiredMessage** obsługują błąd wynikający z braku wprowadzenia danych do komponentów typu **inputText**

```
<h:commandLink action="#{managed_produkt.dodaj_produkt}"
value="OK" />
</h:form>
</h:body>
</html>
```

Znacznik **<h:commandLink** pozwala powrócić do strony, której nazwę zwraca bezparametrowa metoda `dodaj_produkt` z obiektu klasy `Managed_produkt` (wartość atrybutu **action**) – jest to strona **rezultat.xhtml**:

```
public String dodaj_produkt() {
    String[] dane = {nazwa, cena, promocja};
    fasada.utworz_produkt(dane);
    dane_produktu();
    return "rezultat";
}
```

Należy dodać stronę rezultat w folderze jsf – **New/Other/JavaServer Faces/JSF Page**, File Name- rezultat, Folder: jsf



Do strony rezultat.xhtml dodano kod JSF do prezentacji danych produktu oraz ceny brutto, pobieranych z atrybutów obiektu managed_produkkt typu Managed_produkkt

The screenshot shows the NetBeans IDE interface with the following components:

- Projects:** A tree view on the left showing the project structure, including files like `web.xml`, `jsf`, `dodaj_produkkt.xhtml`, `rezultat.xhtml`, and `Managed_produkkt.java`.
- Source Editor:** The main window displays the XML code for `rezultat.xhtml`. The code is as follows:

```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/T
3 <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://java.sun.com/jsf/html">
5   <h:head>
6       <title>Facelet Title</title>
7   </h:head>
8   <h:body>
9       <h:form>
10          <h:panelGrid columns="2">
11              <h:outputLabel value="Nazwa produktu" for="nazwa" />
12              <h:outputText id="nazwa" value="#{managed_produkkt.nazwa}" />
13              <h:outputLabel value="Cena produktu" for="cena" />
14              <h:outputText id="cena" value="#{managed_produkkt.cena}" />
15              <h:outputLabel value="Promocja produktu" for="promocja" />
16              <h:outputText id="promocja" value="#{managed_produkkt.promocja}" />
17              <h:outputLabel value="Cena brutto produktu" for="brutto" />
18              <h:outputText id="brutto" value="#{managed_produkkt.cena_brutto}" />
19          </h:panelGrid>
20          <h:commandButton id="powrot" value="Powrot" action="/faces/index" />
21        </h:form>
22      </h:body>
23 </html>
```
- Navigator:** A panel at the bottom left showing the component tree for `rezultat.xhtml`, including `Namespaces`, `Rules`, `HTML`, and `title`.
- Output:** A panel at the bottom right showing logs from `DRDA_SecurityInstalled.I` and `Server sieciowy Apache Derby - 10.4.2.1 - (706043) uruchomiony i gotowy do zaakceptowania połączeń n`.

```
<?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:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core">
<h:head>
<title>Facelet Title</title>
</h:head>
<h:body>
<h:form>
<h:panelGrid columns="2">
<h:outputLabel value="Nazwa produktu" for="nazwa" />
<h:outputText id="nazwa" value="#{managed_produkt.nazwa}"/>
<h:outputLabel value="Cena produktu" for="cena" />
<h:outputText id="cena" value="#{managed_produkt.cena}"/>
<h:outputLabel value="Promocja produktu" for="promocja" />
<h:outputText id="promocja" value="#{managed_produkt.promocja}"/>
<h:outputLabel value="Cena brutto produktu" for="brutto" />
<h:outputText id="brutto" value="#{managed_produkt.cena_brutto}" />
</h:panelGrid>
<h:commandButton id="powrot" value="Powrot" action="/faces/index"/>
</h:form>

</h:body>
</html>
```

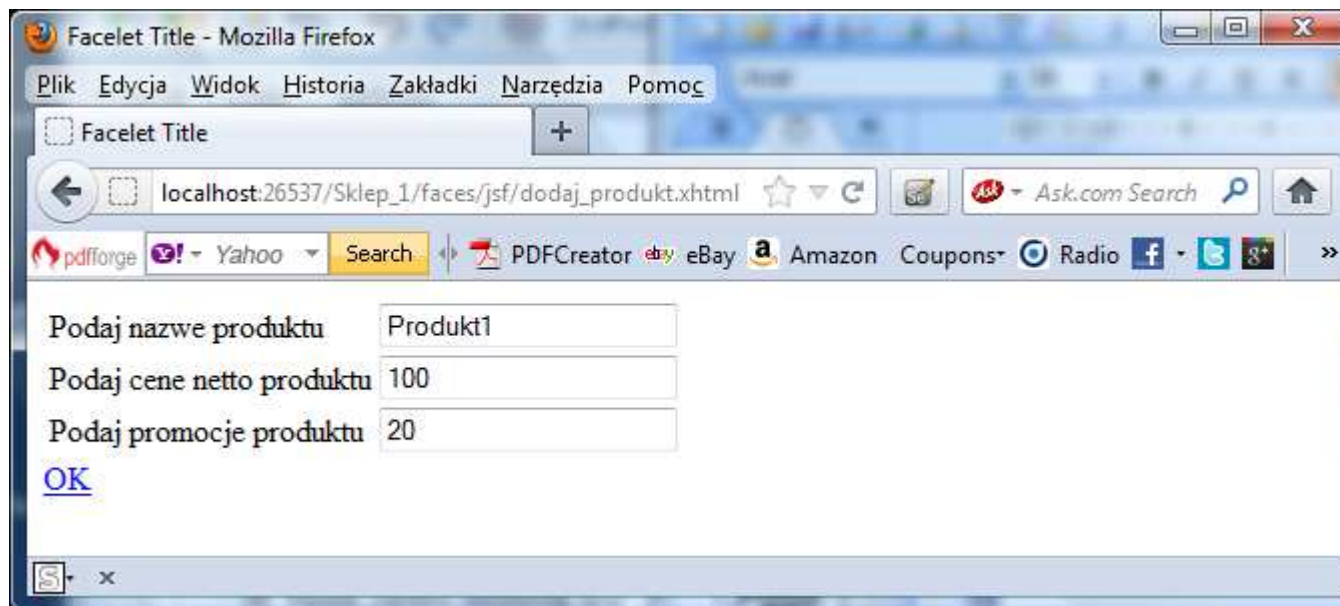
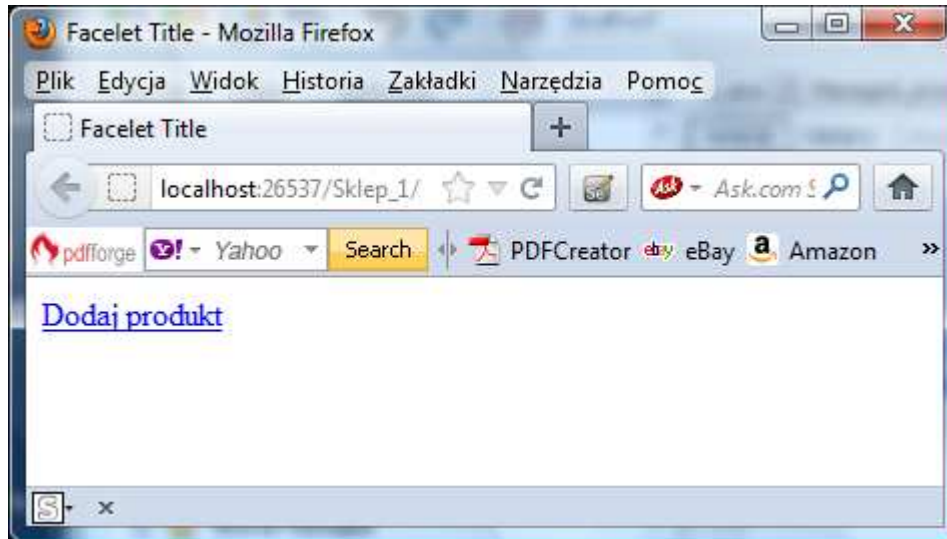
Siatka **panelGrid** umożliwiającą wyświetlanie danych produktu pobranych z obiektu typu `Managed_produkt` w dwóch kolumnach za pomocą komponentów **outputLabel** i **outputText**

Znacznik **<h:commandButton** pozwala powrócić do strony głównej **index** (wartość atrybutu **action**)

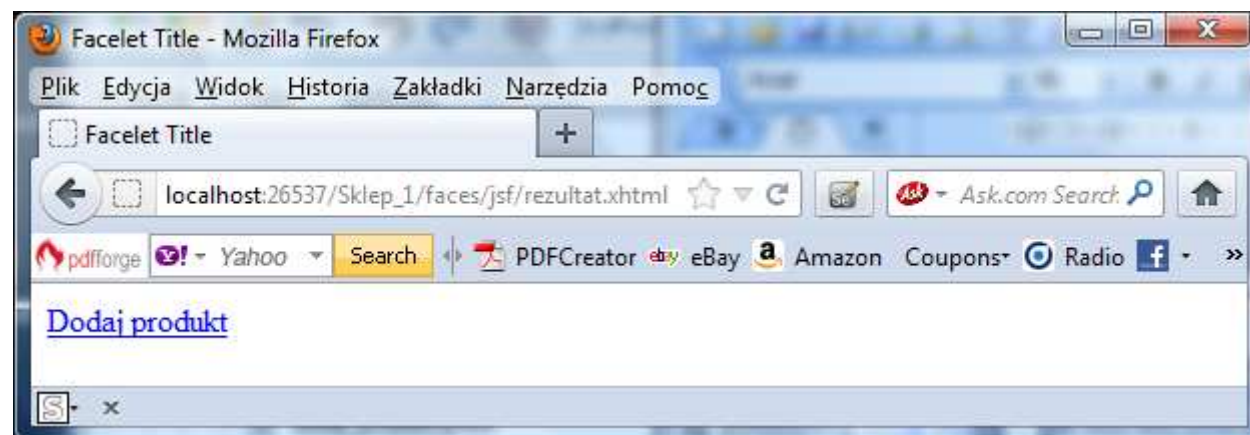
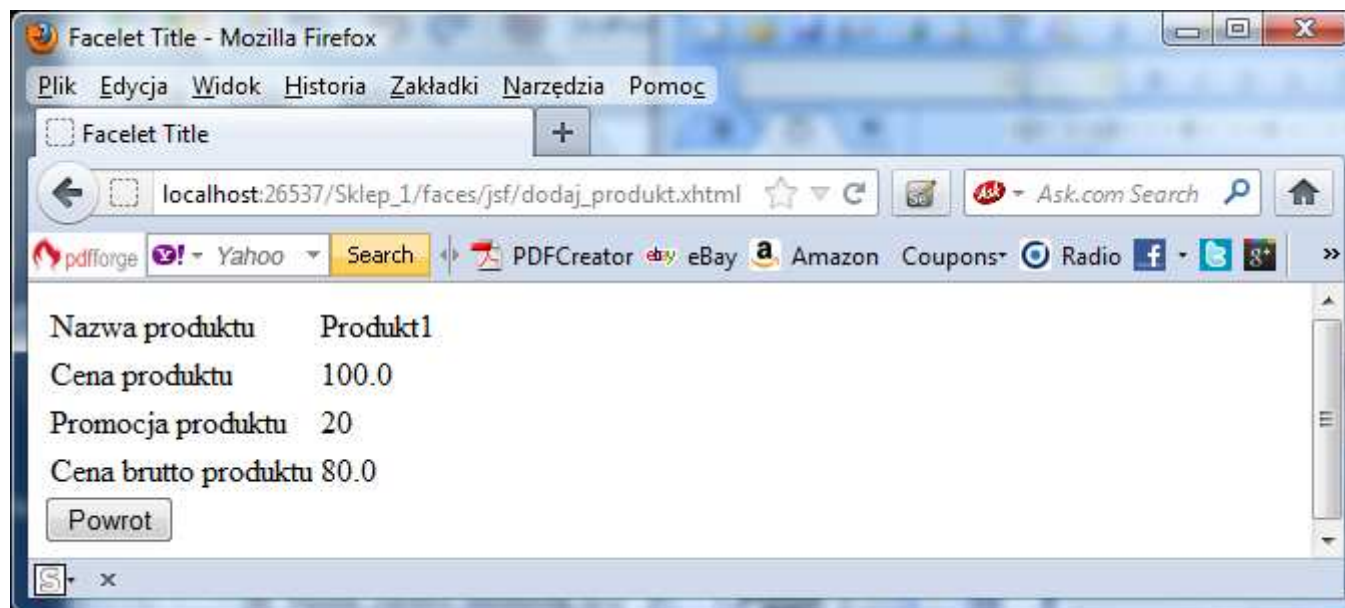
Uzupełniono kod klasy głównej **index.xhtml** – znacznik **link** pozwala wywołać stronę **dodaj_produkt.xhtml**

```
<?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:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    <h:link outcome="/jsf/dodaj_produkt" value="Dodaj produkt"/>
  </h:body>
</html>
```

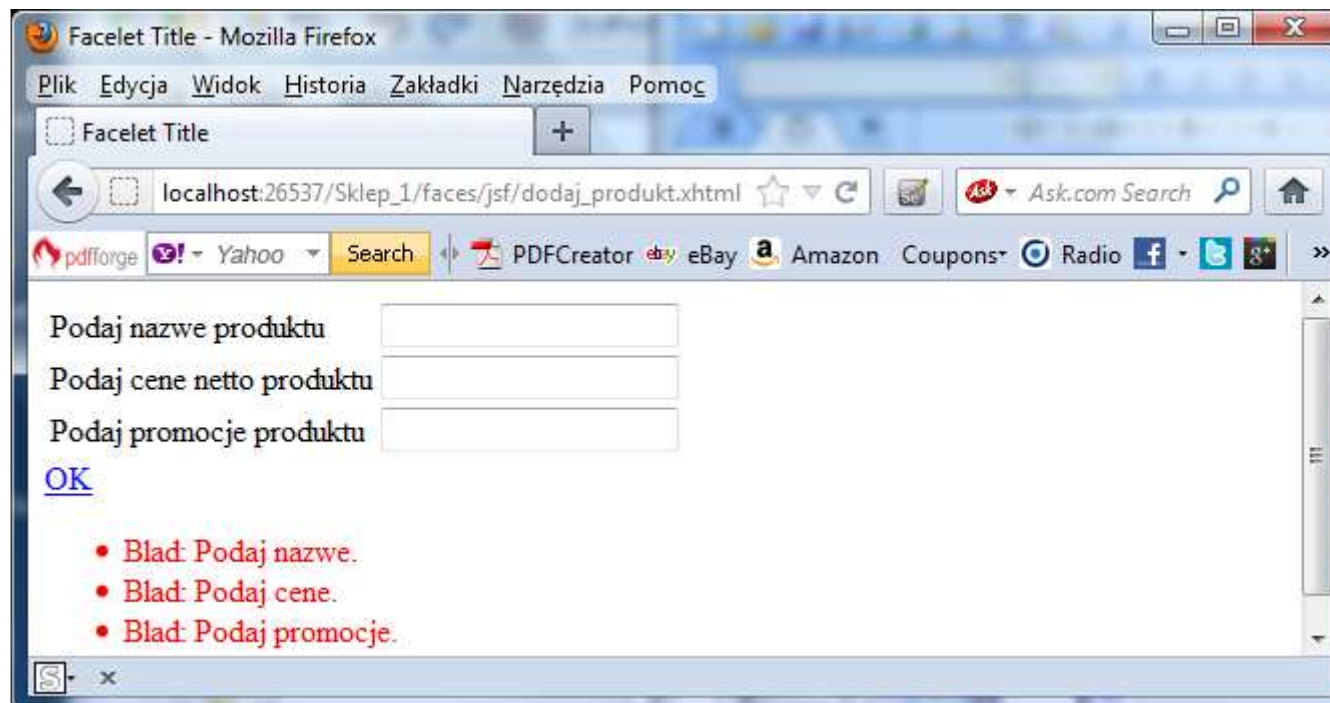
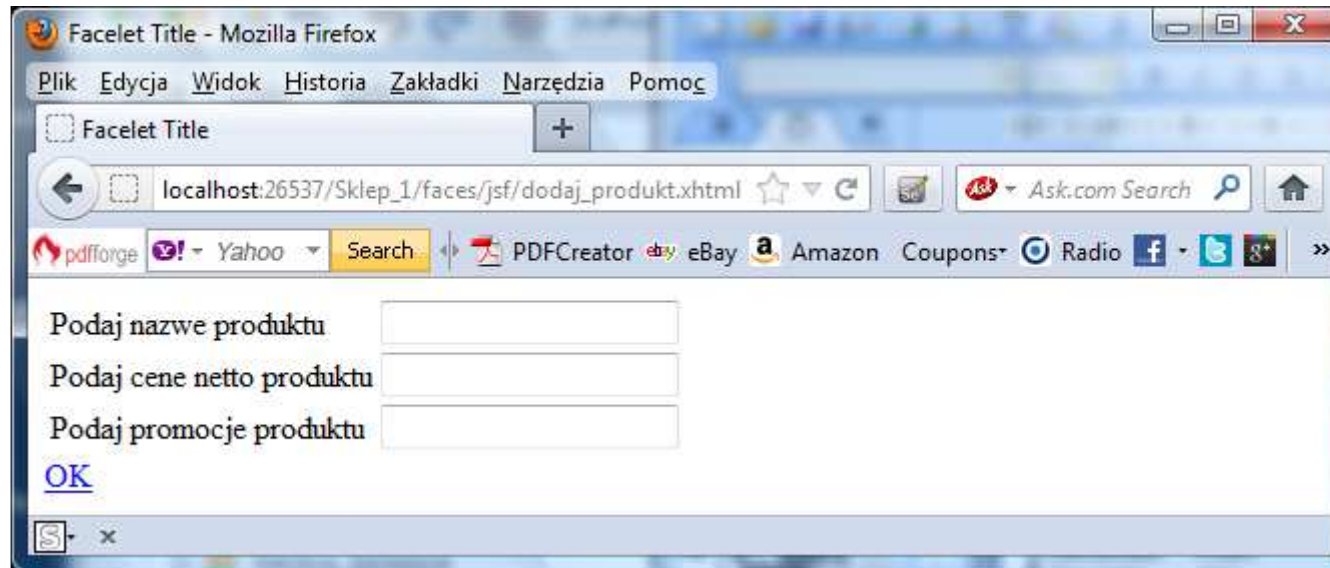
Stany aplikacji – uruchomienie strony głównej. Po wybraniu linku Dodaj produkt przejście do strony dodaj_produk



Po kliknięciu OK. na stronie **dodaj_produk** przejście do strony **rezultat** i wyświetlenie danych. Po kliknięciu na **Powrót** przejście do strony **głównej** index

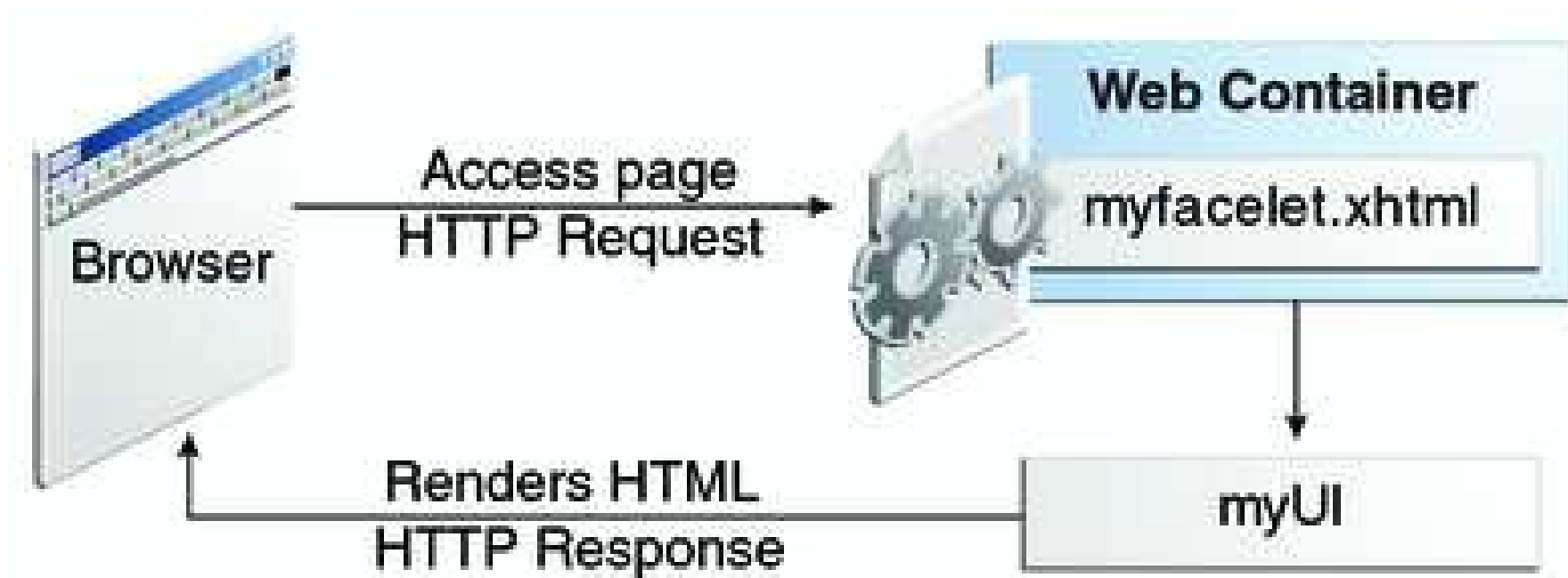


Działanie walidacji typu **required** kontrolującej brak danych



**Dodatek-Biblioteki znaczników obsługiwanych przez
Facelets wg
<http://docs.oracle.com/javaee/6/tutorial/doc/giqzr.html>**

Interakcja między warstwą klienta i warstwą internetową



Przebieg fazy: żądanie - odpowiedź

Strona www, myfacelet.xhtml, jest zbudowana ze **znaczników komponentów JavaServer Faces**. Znaczniki elementów są wykorzystywane do dołączenia do widoku elementów (reprezentowanych przez myUI na rysunku), które są po stronie serwera, w warstwie internetowej.

Oprócz elementów, strona może odwołać się do obiektów, takich jak:

- **słuchaczy zdarzeń, walidatorów oraz konwerterów**, które są reprezentowane w postaci komponentów
- **komponenty JavaBeans**, pozyskujące i przetwarzające dane specyficznym dla komponentów

Skutkiem żądania z warstwy klienta (**Request**), widok jest renderowany jako odpowiedź (**Response**). Renderowanie jest procesem, w którym na podstawie zawartości strony kontener internetowy tworzy strony typu HTML lub XHTML, które mogą być odczytywane przez warstwę klienta zawierającą przeglądarkę.

Dodatek-Biblioteki znaczników obsługiwanych przez Facelets

Biblioteki znaczników	URI	Prefiks	Przykład	Zawartość
JavaServer Faces Facelets Tag Library	http://java.sun.com/jsf/facelets	ui:	ui:component ui:insert	Znaczniki szablonów
JavaServer Faces HTML Tag Library	http://java.sun.com/jsf/html	h:	h:head h:body h:outputText h:inputText	Znaczniki komponentów JavaServer Faces dla wszystkich obiektów komponentów typu UI
JavaServer Faces Core Tag Library	http://java.sun.com/jsf/core	f:	f:actionListener f:attribute	Znaczniki niestandardowych akcji JavaServer Faces, które są niezależne od narzędzia renderowania
JSTL Core Tag Library	http://java.sun.com/jsp/jstl/core	c:	f:actionListener f:attribute	JSTL 1.2 Core Tags
JSTL Functions Tag Library	http://java.sun.com/jsp/jstl/functions	fn:	fn:toUpperCase fn:toLowerCase	JSTL 1.2 Functions Tags