

# Budowa prostej aplikacji wielowarstwowej

Laboratorium 1

Technologie internetowe

<https://docs.oracle.com/javaee/7/javaxserver-faces-2-2/vdldocs-facelets/toc.htm>

Zofia Kruczkiewicz

Wykaz pytań dotyczących materiału wykorzystanego w lab1, które należy opracować (wykłady:1, 2, 3, 4)

1. Wyjaśnij rolę klasy Produkt1
2. Przedstaw rolę klasy Fasada\_warstwy biznesowej
3. Przedstaw rolę klasy typu Managed\_produkt
4. Wyjaśnij rolę atrybutów znacznika:

**<h:inputText**

```
id="nazwa"
title="Podaj nazwe:"
value="#{managed_produkt.nazwa}"
required="true"
requiredMessage="Bład: Podaj nazwe." >
```

**</h:inputText>**

Jaka metoda z klasy Managed\_produkt jest wywołana przy obsłudze atrybutu:

```
value="#{managed_produkt.nazwa}"?
```

5. Jaka metoda z klasy Managed\_produkt jest wywołana przy obsłudze atrybutu **value** w znaczniku

```
<h:outputText id="nazwa" value="#{managed_produkt.nazwa}"/>
```

6. W jaki sposób wyznaczona jest wartość ceny brutto produktu wyświetlana na stronie rezultat;

```
<h:outputText id="brutto" value="#{managed_produkt.cena_brutto}" />
```

7. W jakim sposób wyświetlana jest cena\_brutto?

8. Opisz atrybuty znacznika: <h:commandLink action ="#{managed+produkt.dodaj\_produkt}" value="OK">

9. Opisz atrybuty znacznika: <h:commandButton id="powrot" value="Powrot" action ="/faces/index">

10. Opisz atrybuty znacznika <h:link outcome="/warstwa\_intermetowa\_jsf/dodaj\_produkt1" value="Dodaj\_produkt" >

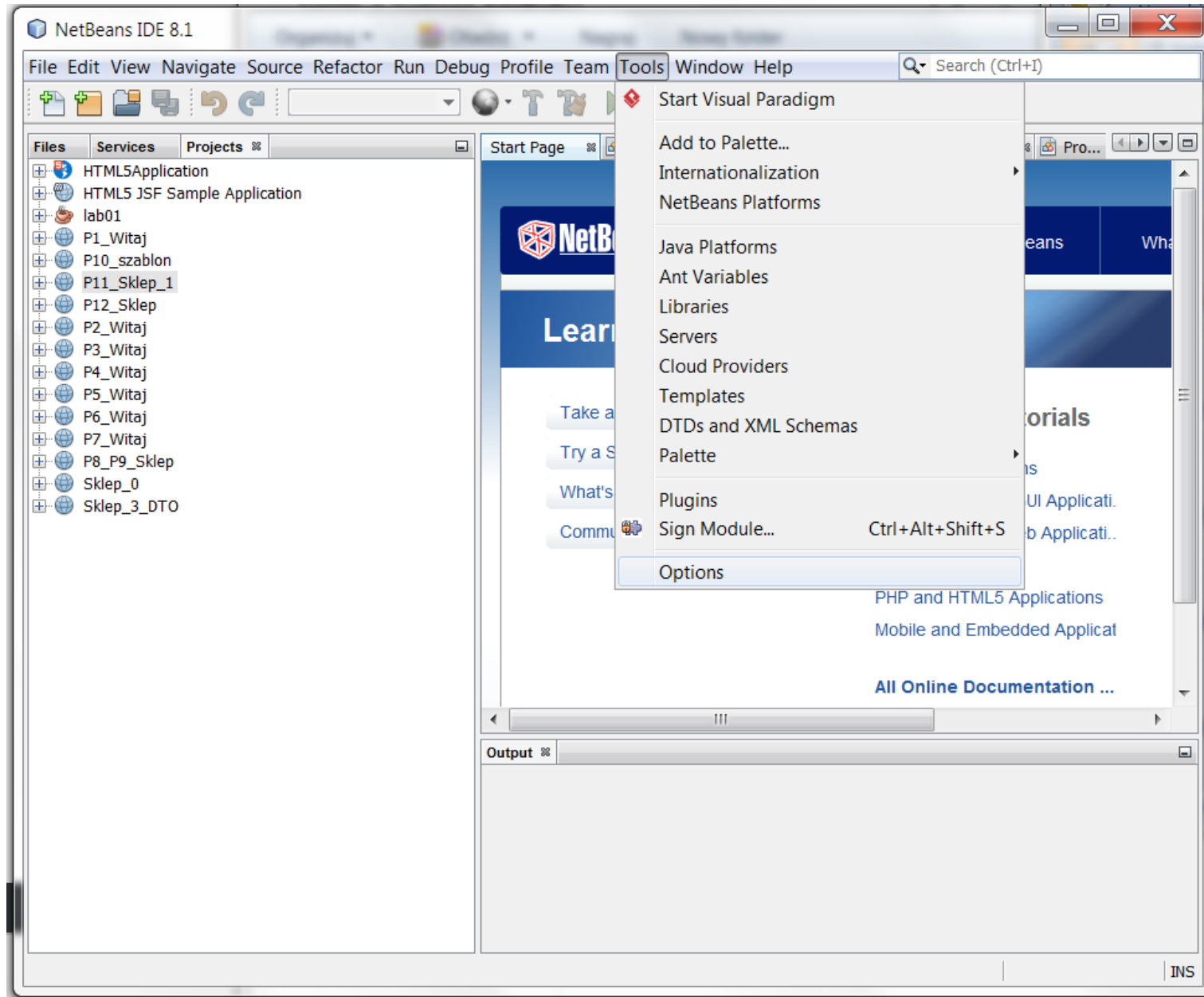
11. Co oznaczają elementy znacznika:

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://xmlns.jcp.org/jsf/html">
```

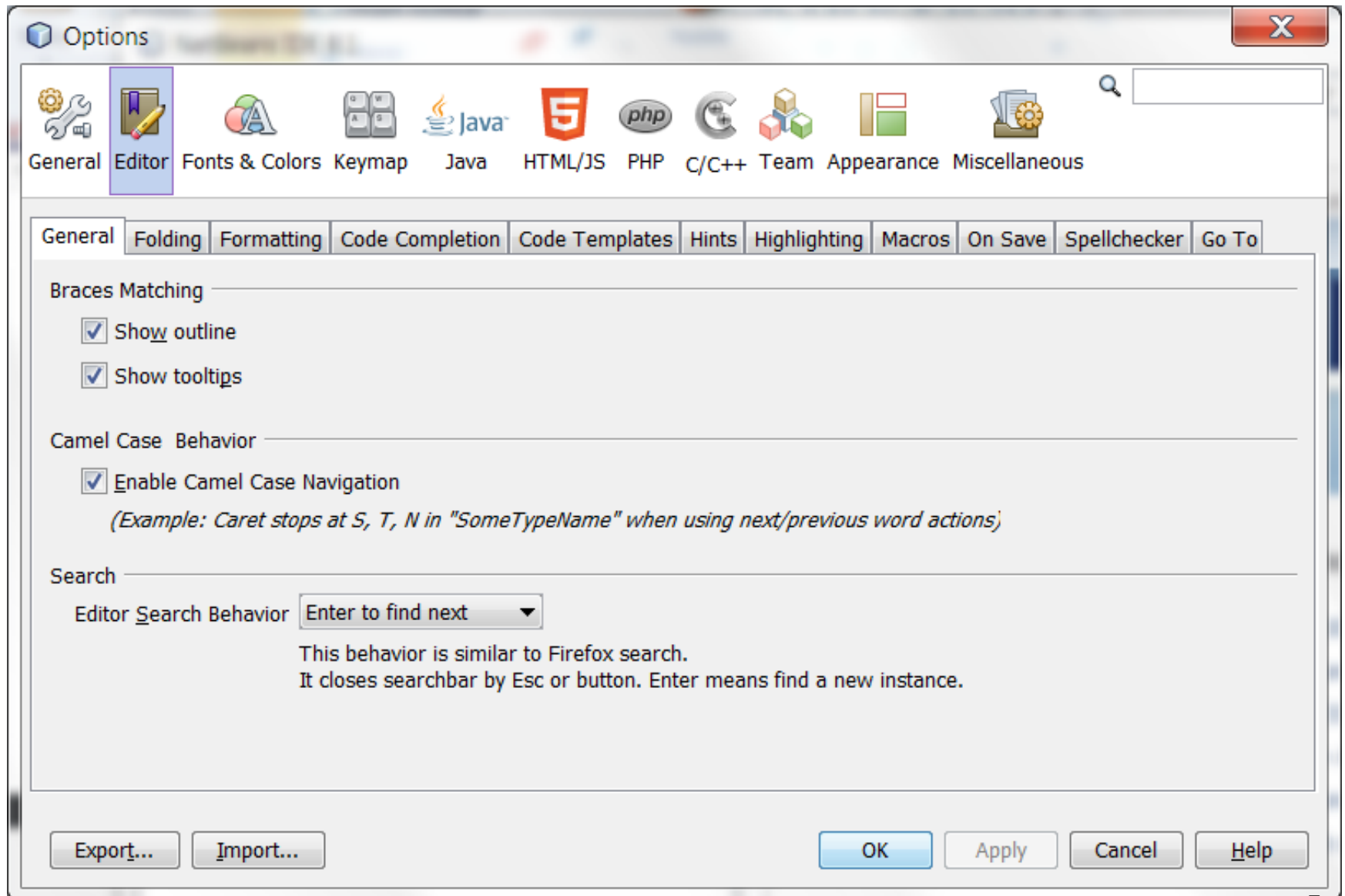
# Przykład 1

## Wykonanie prostych formularzy

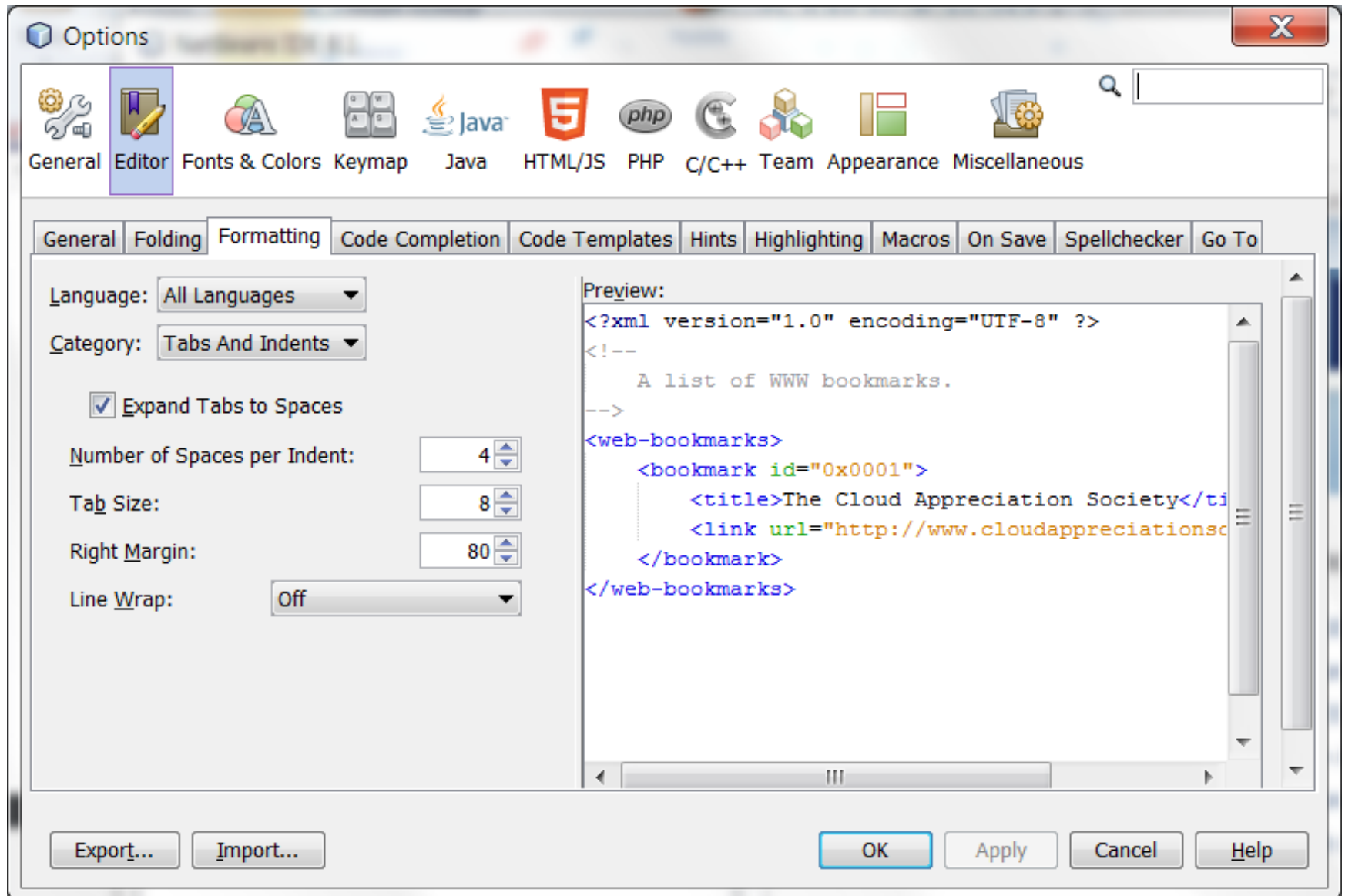
# 1. Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



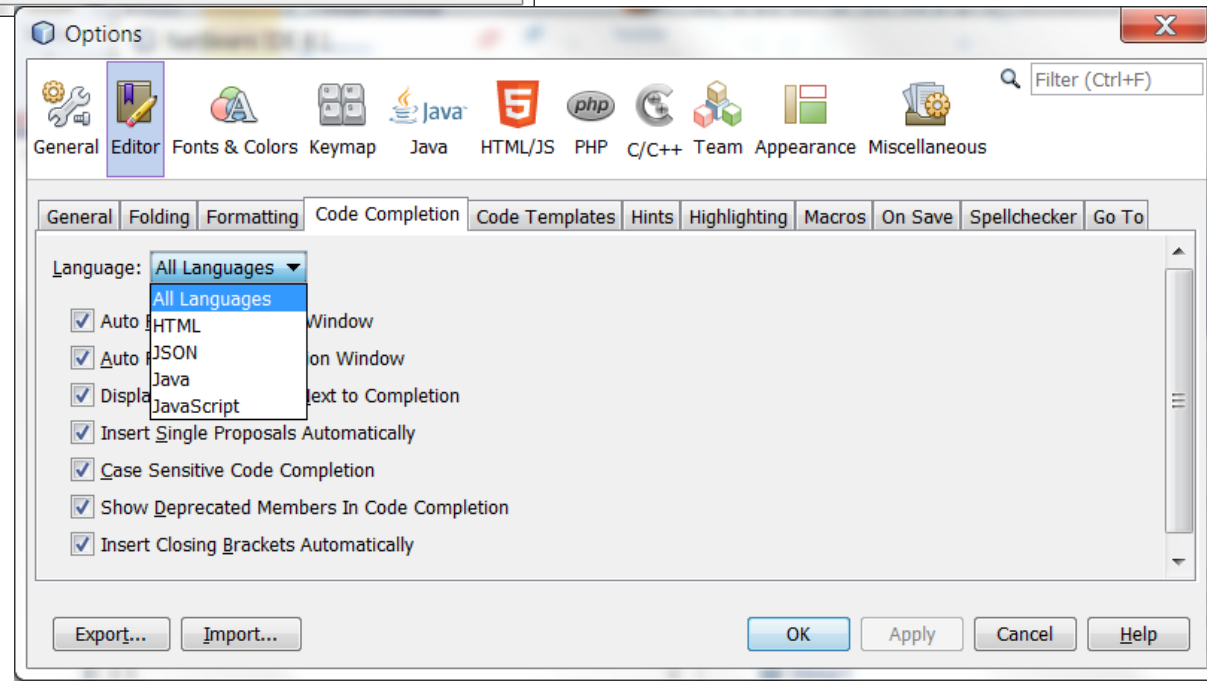
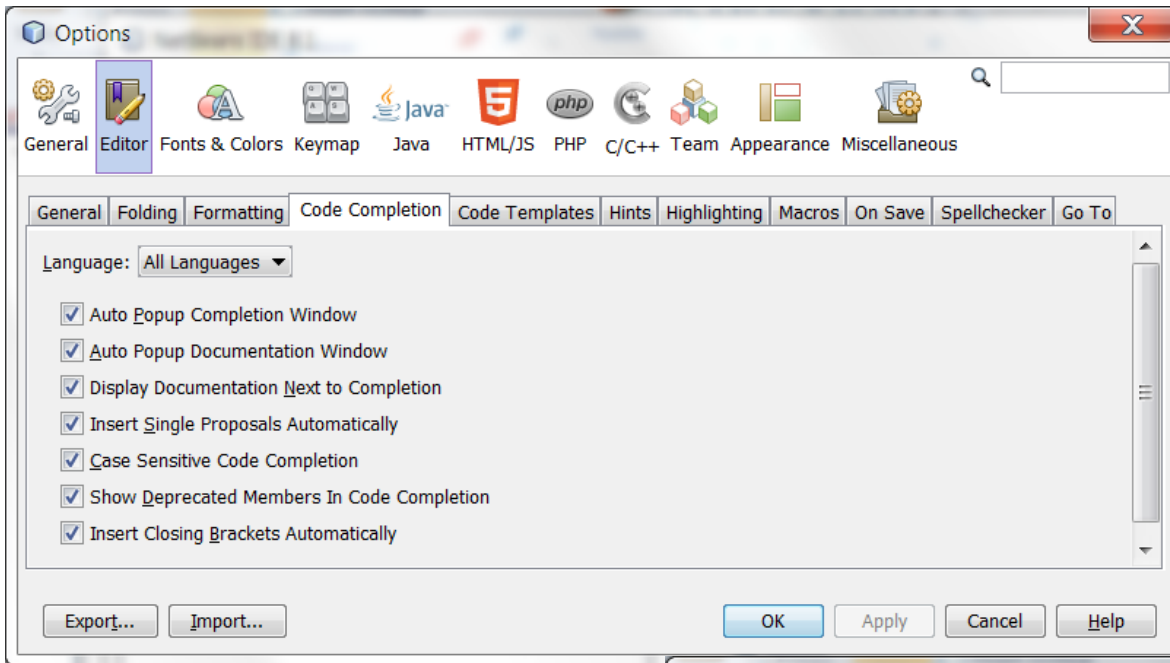
# Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



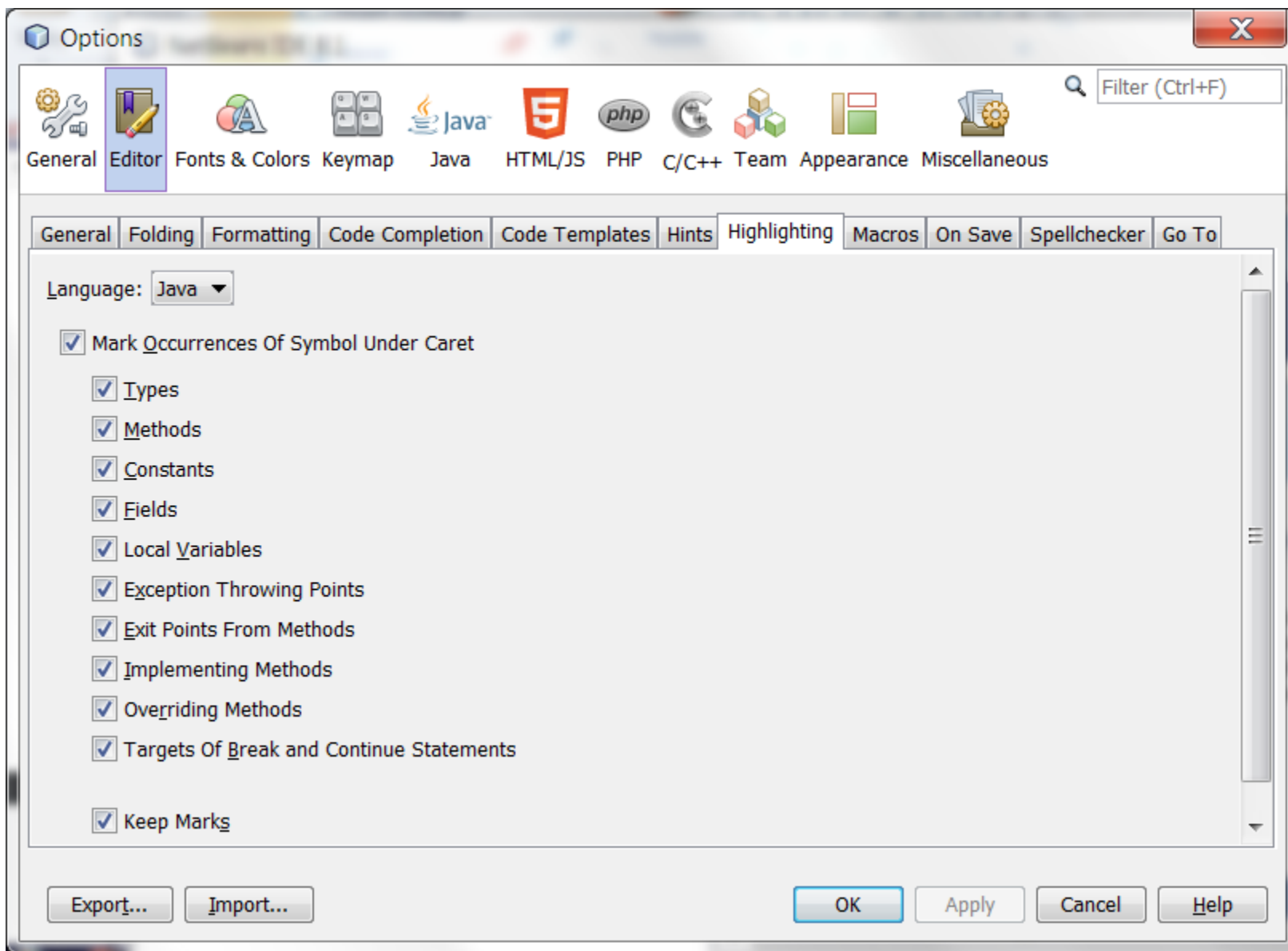
## Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



# Konfigurowanie edytora programu za pomocą Tools/Options/Editor

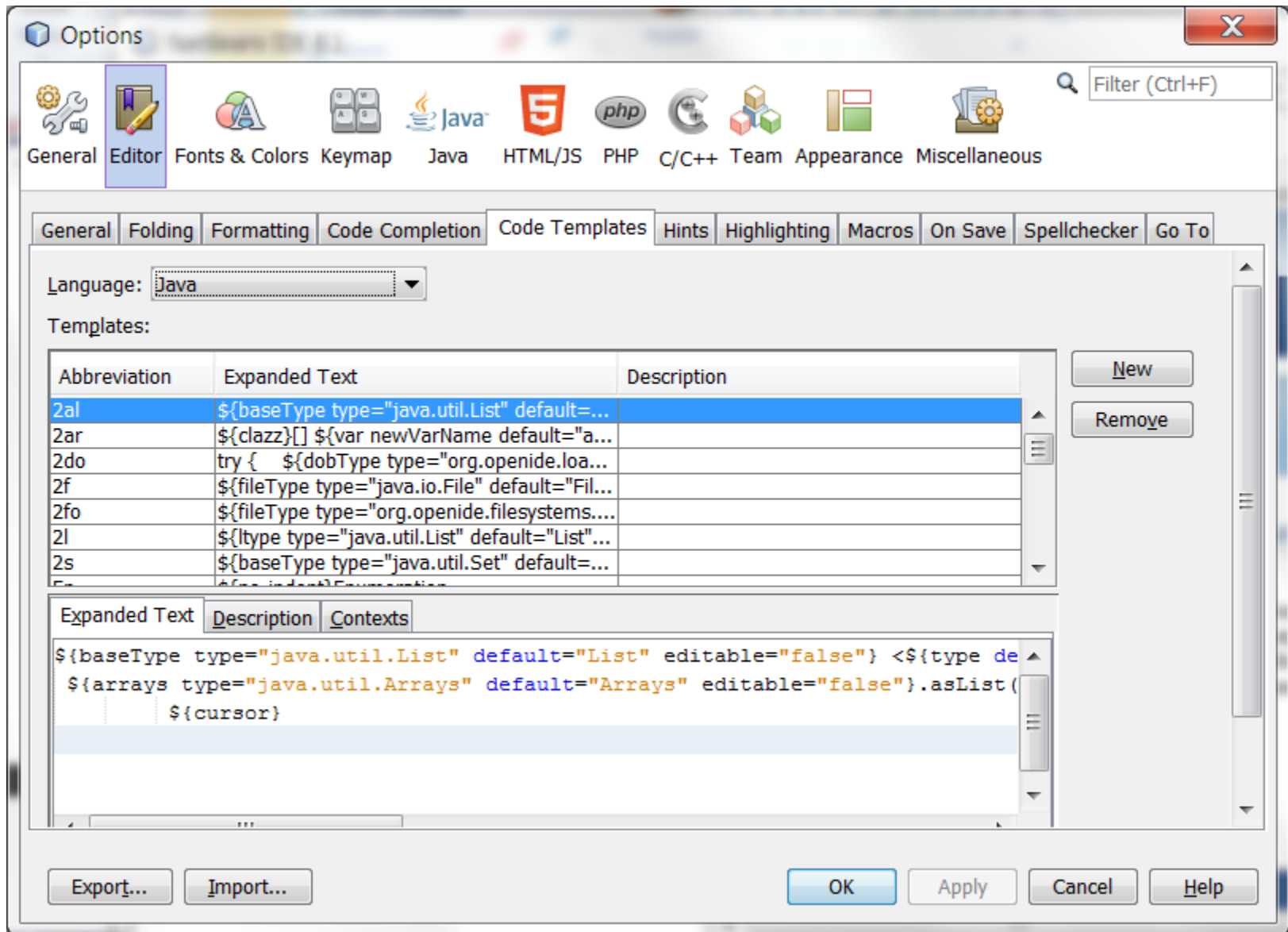


# Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**

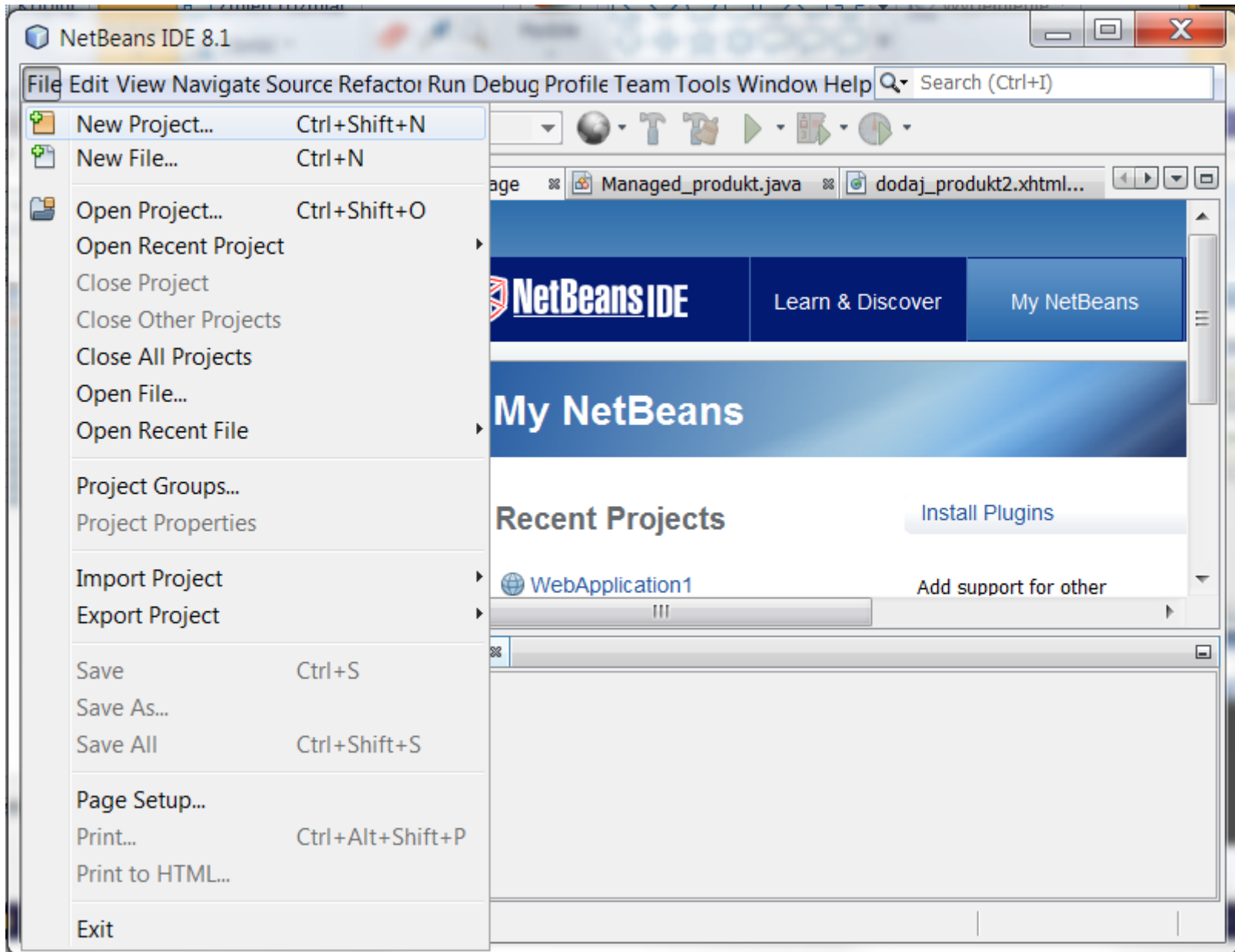




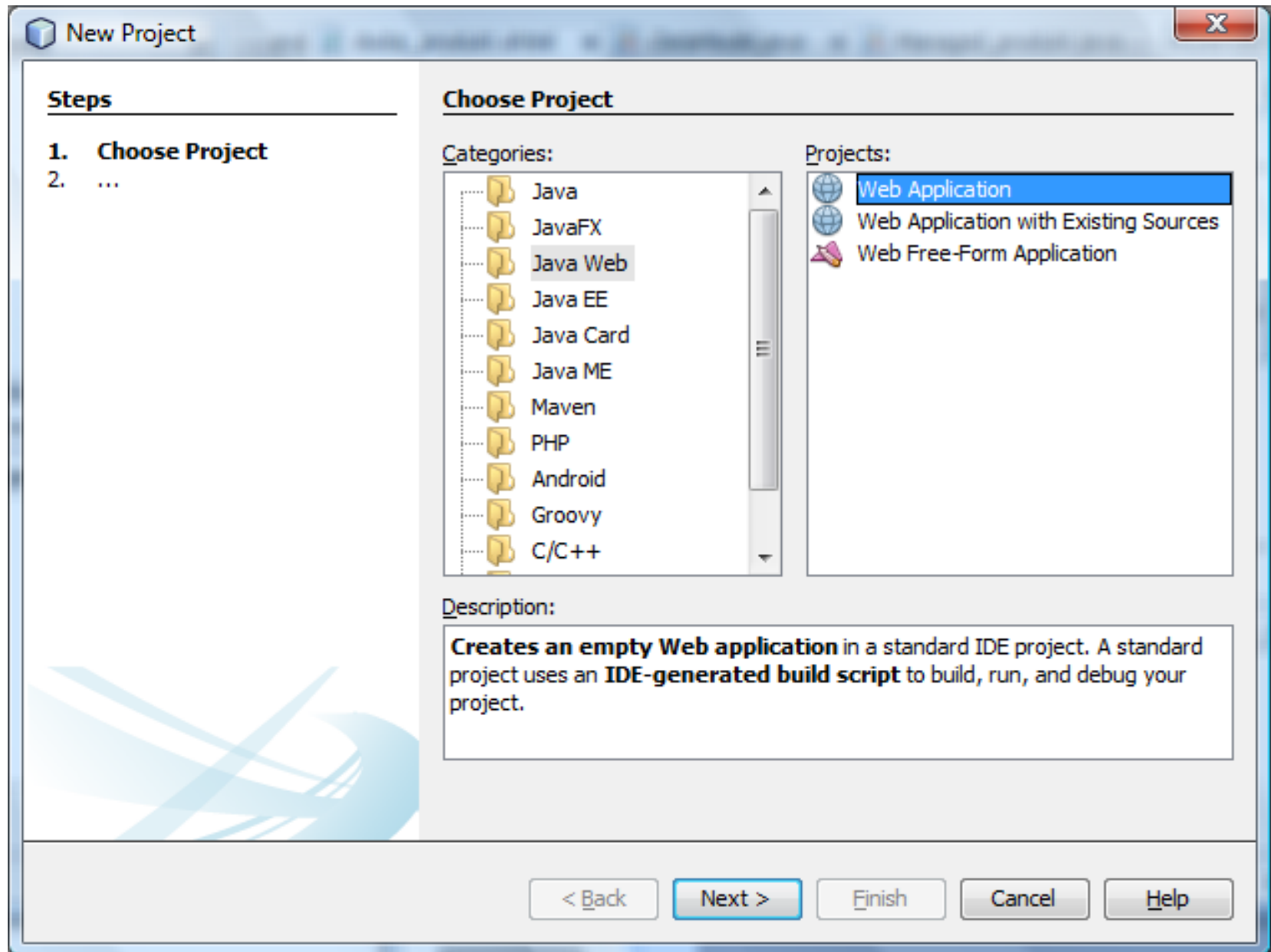
# Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



## 2. 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

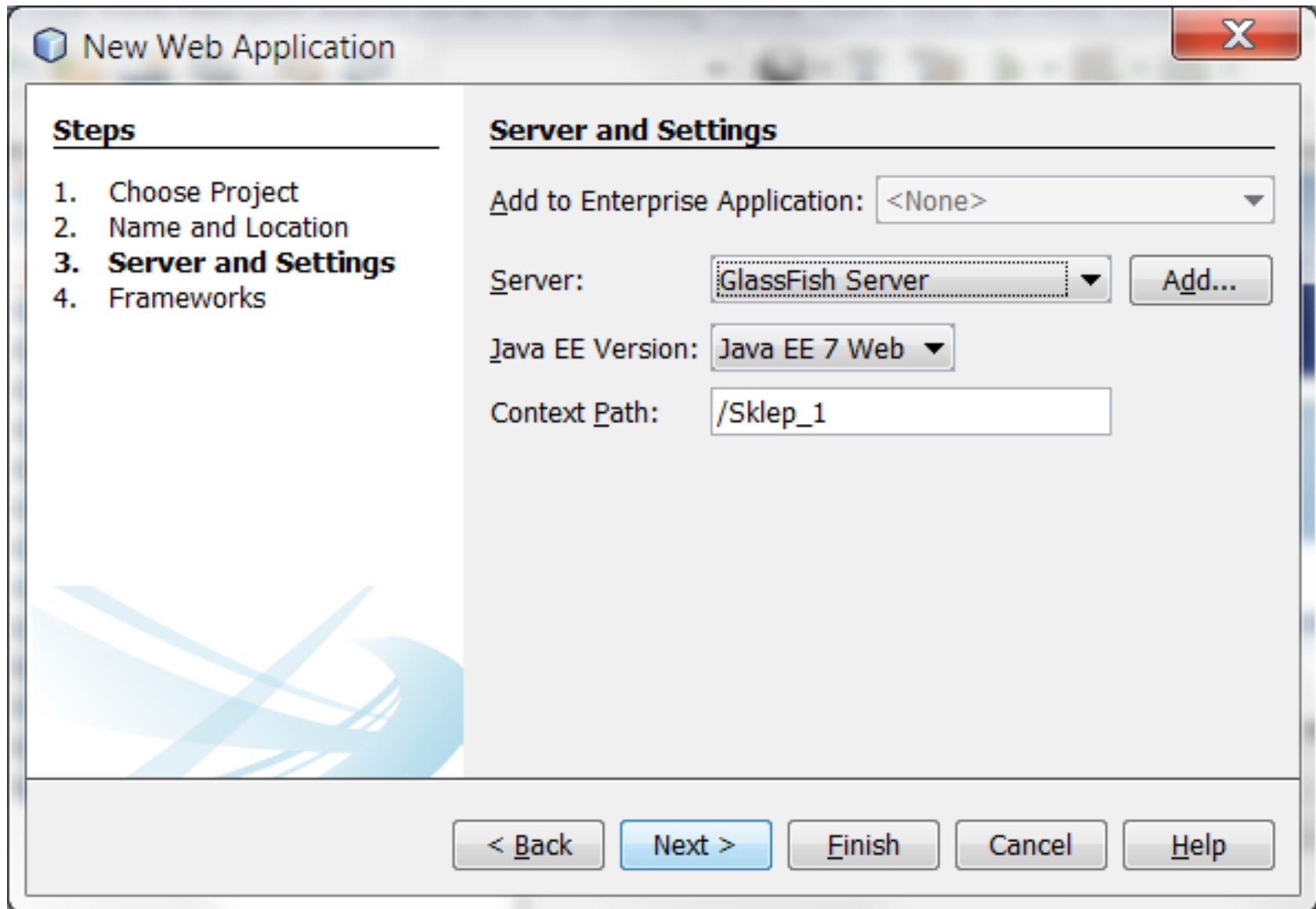
Use Dedicated Folder for Storing Libraries

Libraries Folder:

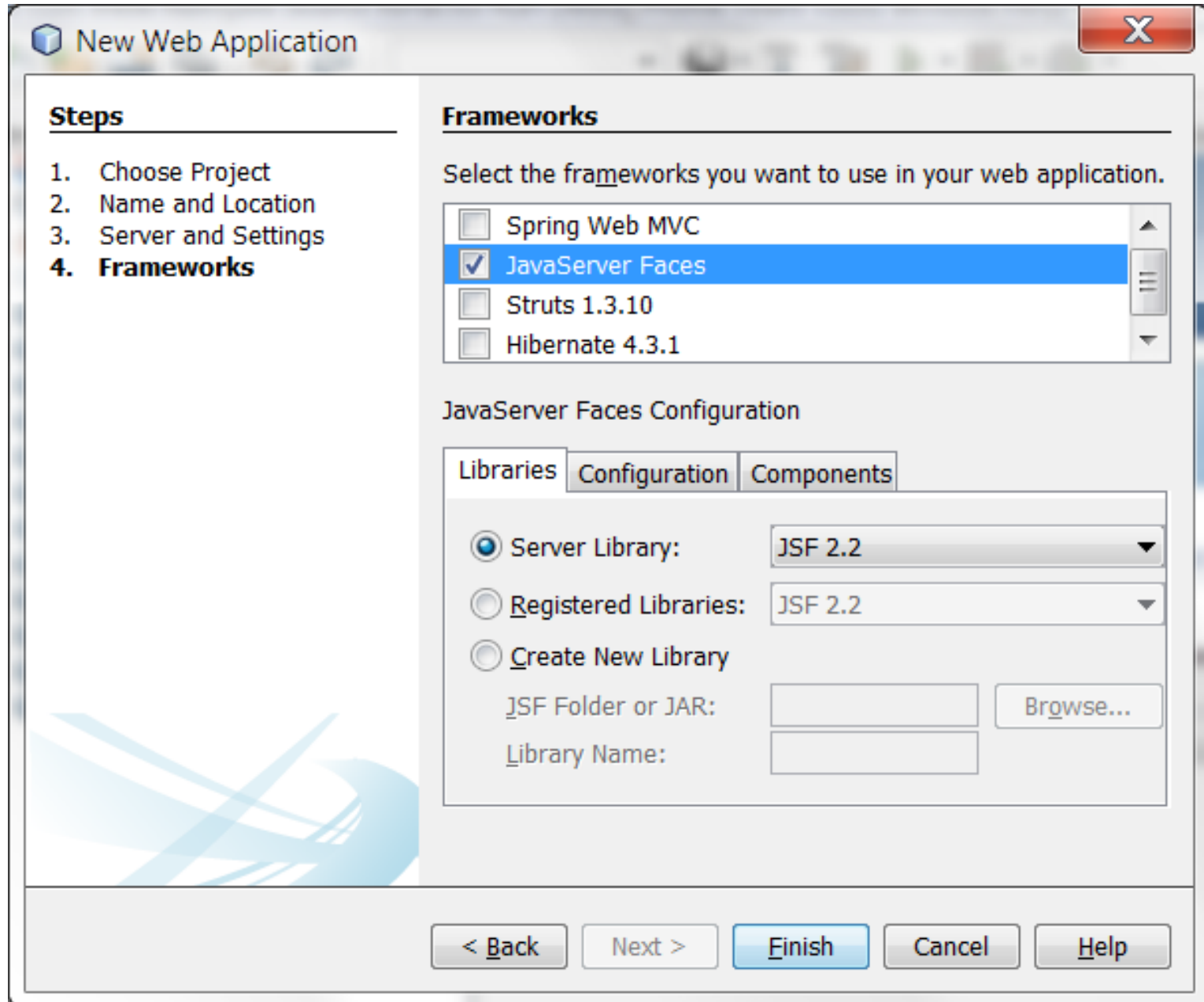
Different users and projects can share the same compilation libraries (see Help)

< Back    **Next >**    Finish    Cancel    Help

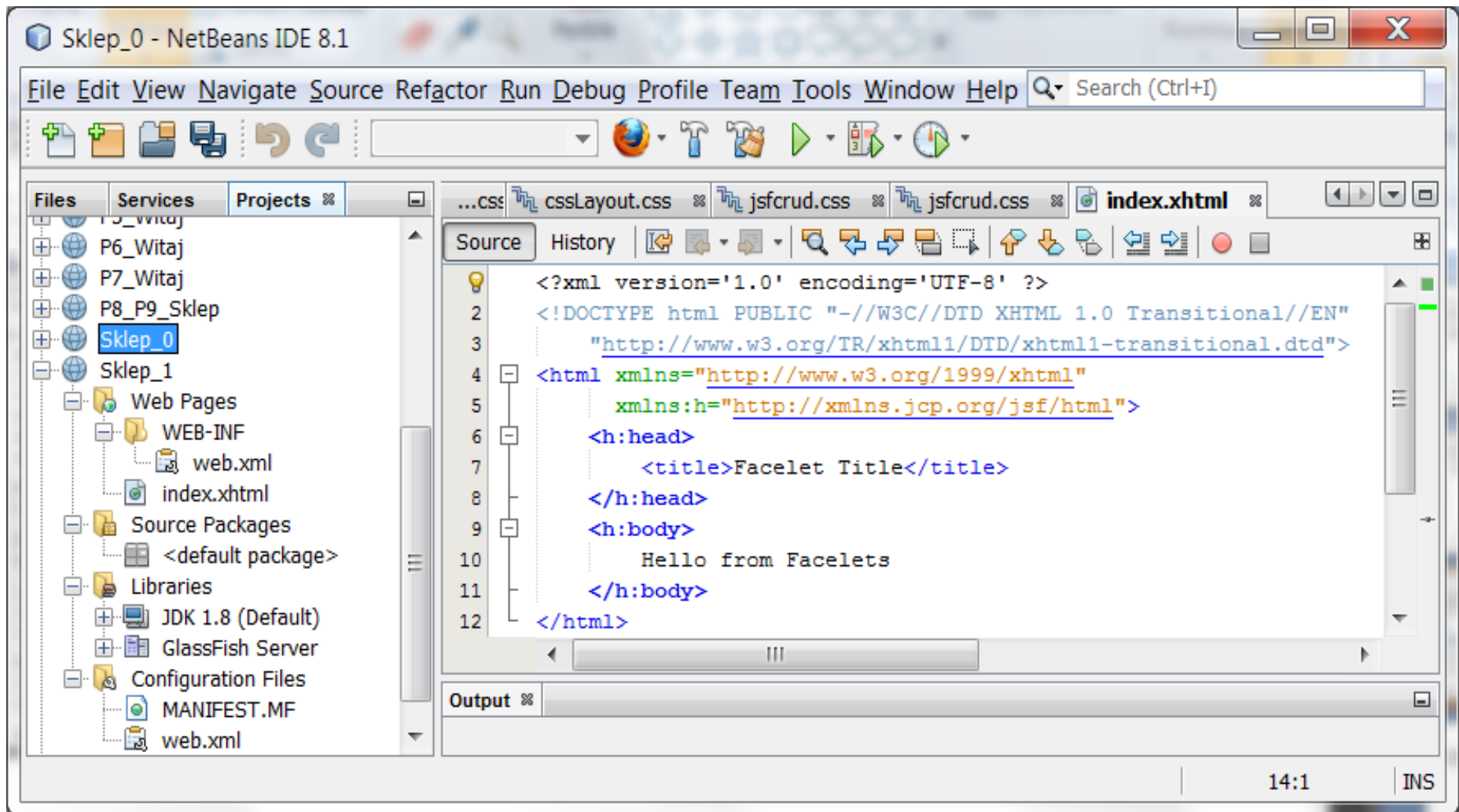
Zakładanie projektu typu Web Application: **Server: GlassFish 4.1, Java EE  
Version: Java EE 7 Web i Next**

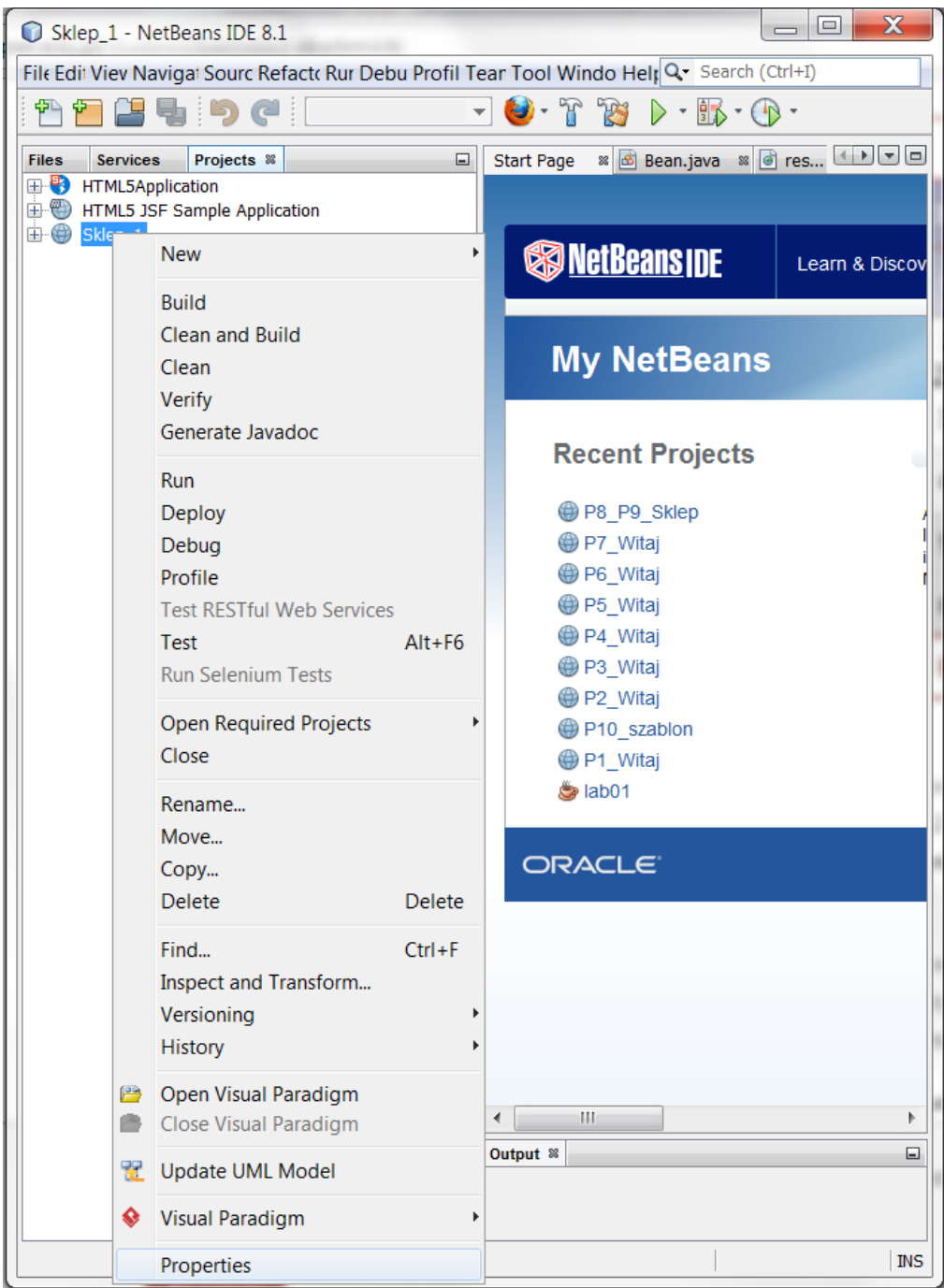


# Zakładanie projektu typu Web Application- zaznaczyć checkbox **JavaServer Faces i Finish**



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

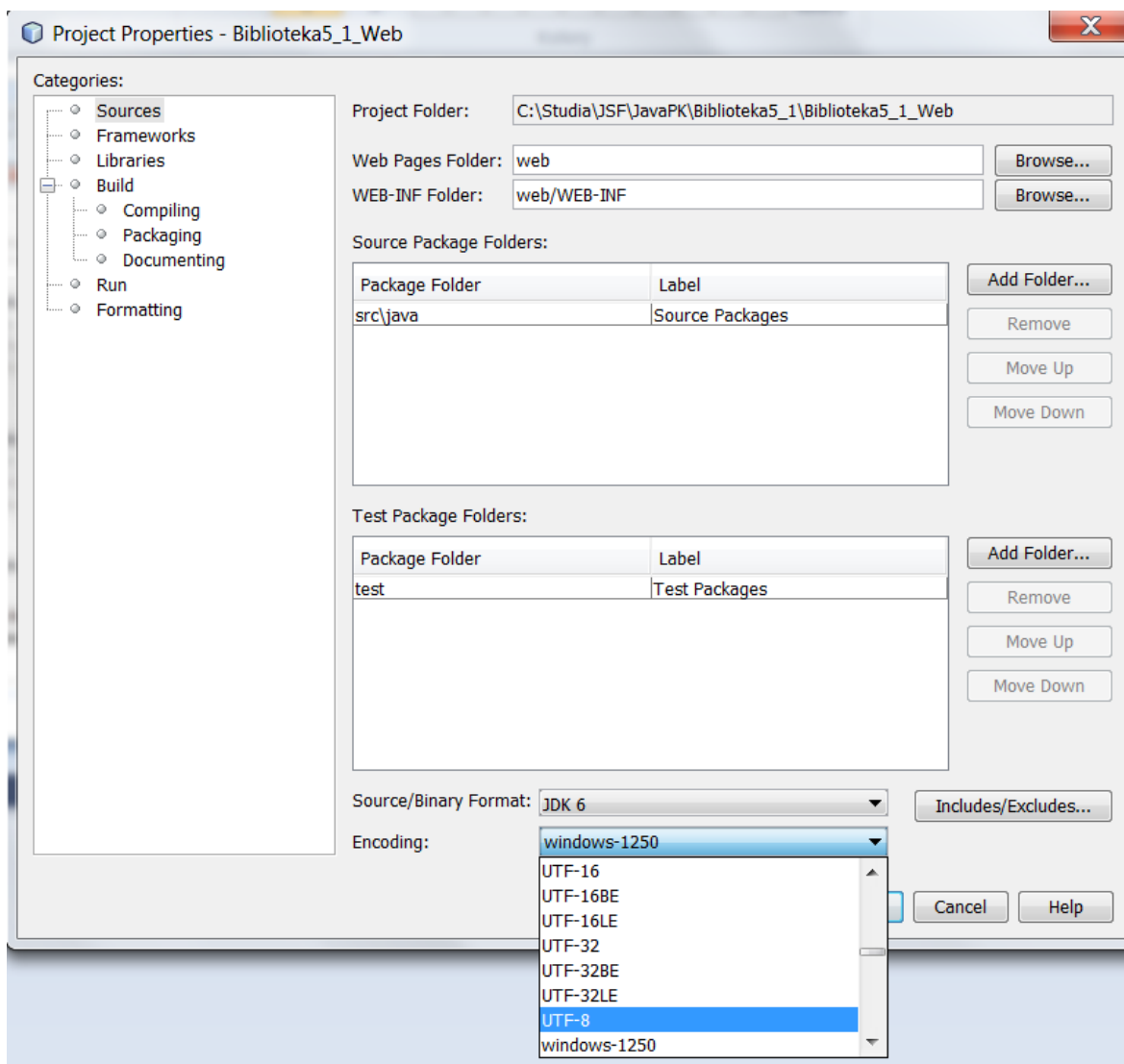




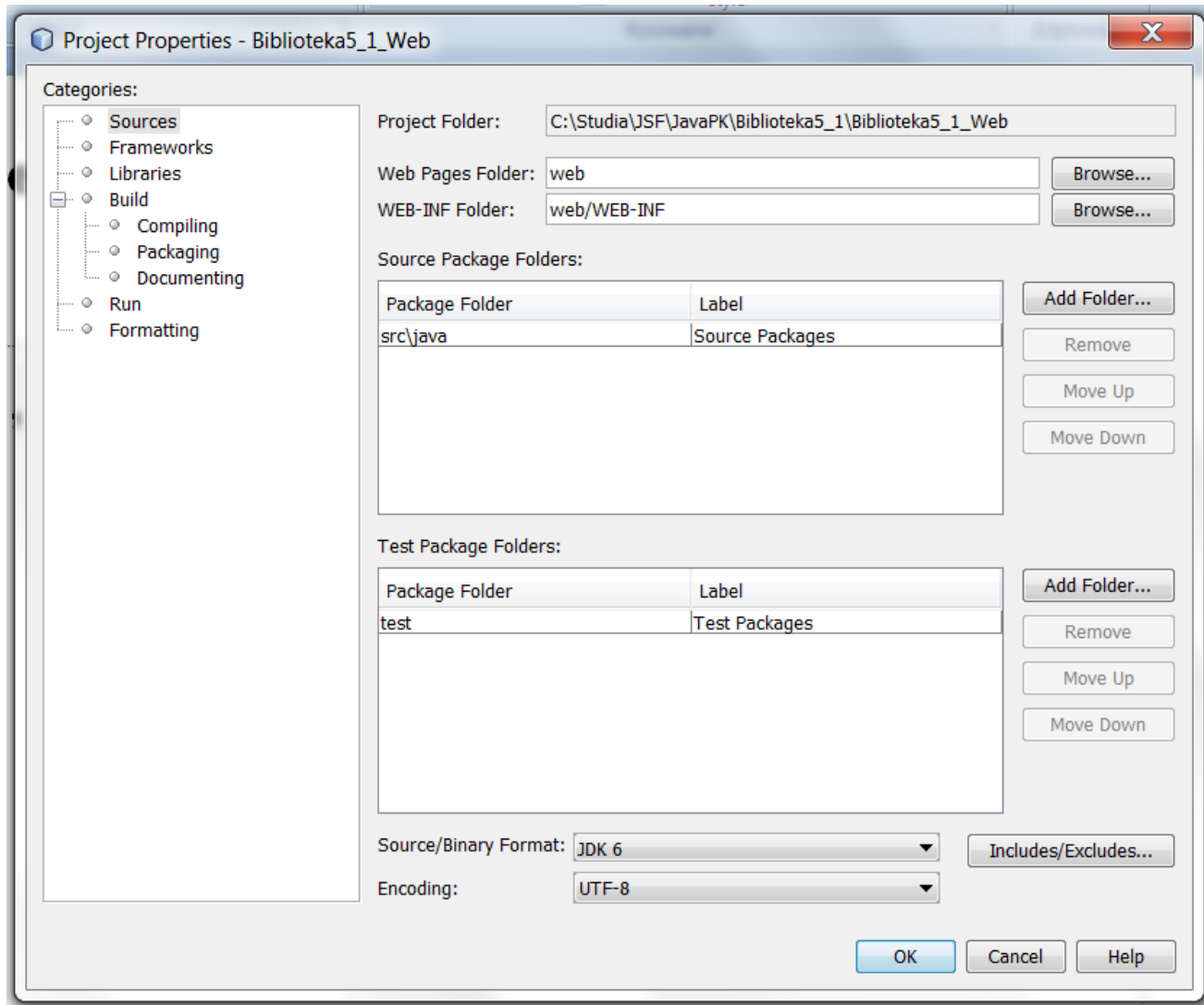
3. Ustawienie kodowania typu **UTF-8** w projektach **Java SE** oraz **Java EE** – w zakładce **Projects** należy prawym klawiszem kliknąć na nazwę projektu i wybrać z listy pozycję **Properties**



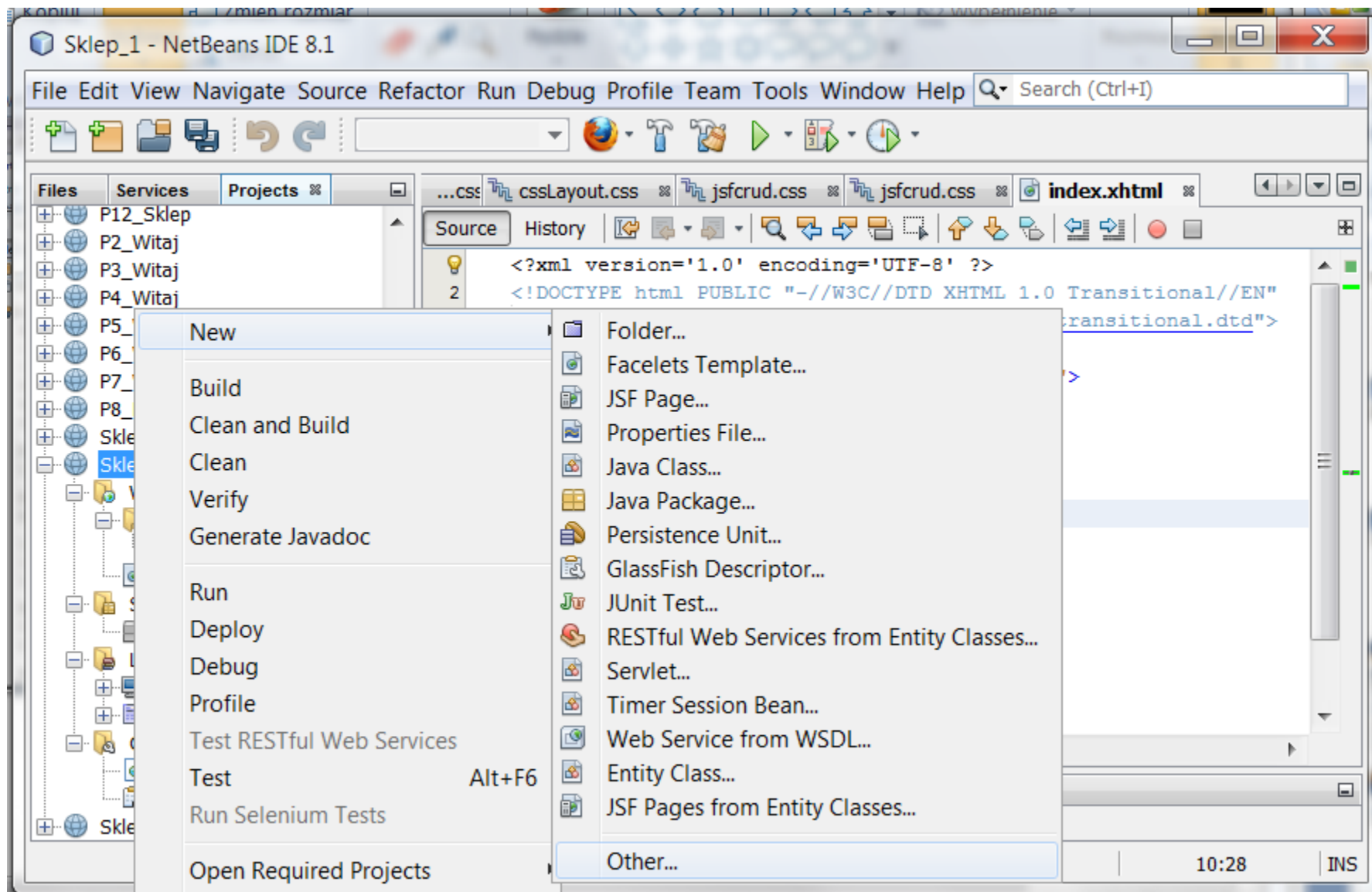
Ustawienie kodowania typu **UTF-8** w projektach **Java SE** oraz **Java EE** – w formularzu typu **Project Properties** należy rozwinąć listę **Encoding** i wybrać pozycję **UTF-8**



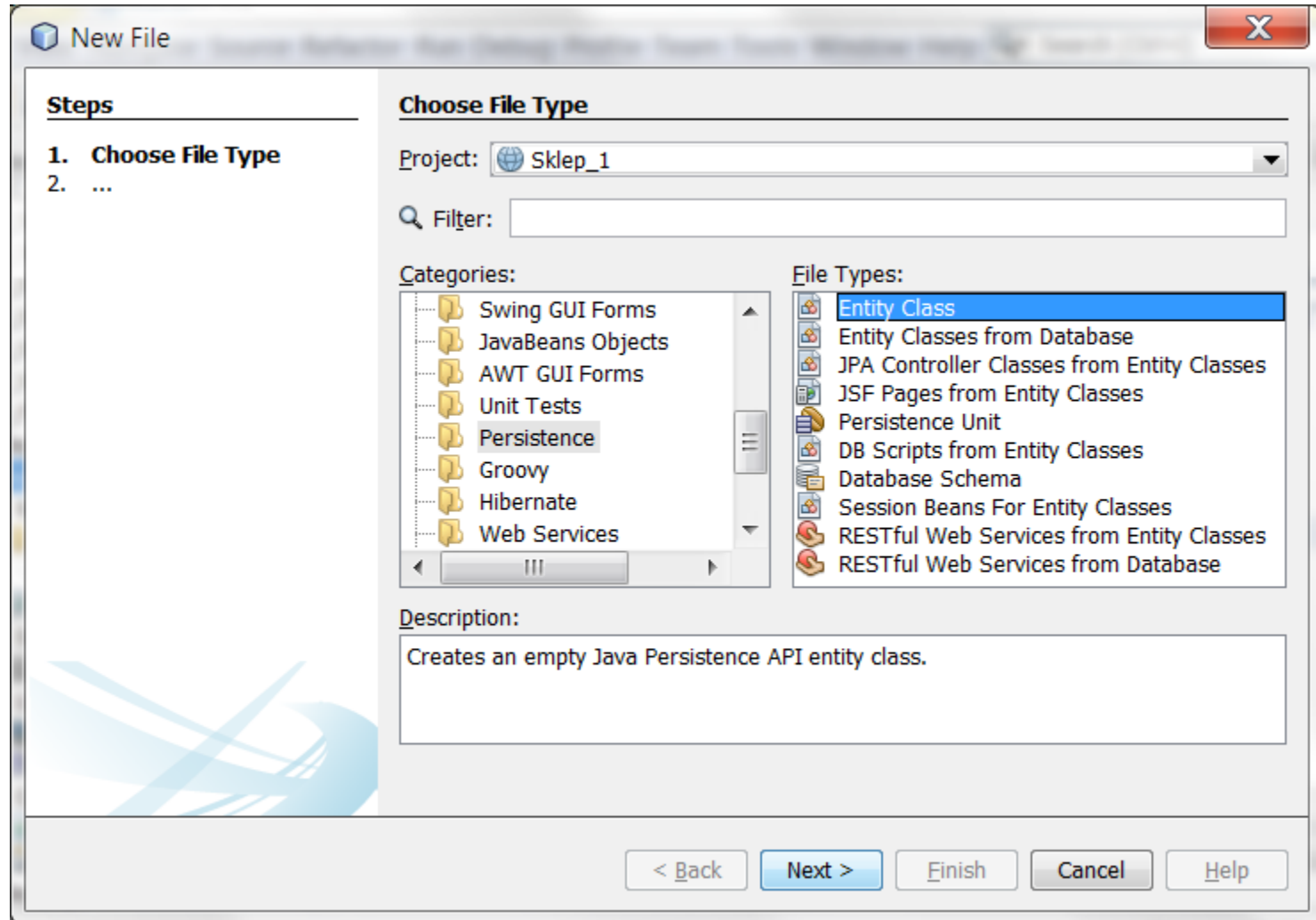
# Ustawienie kodowania typu **UTF-8** w projektach **Java SE** oraz **Java EE** – należy zatwierdzić ustawienia klawiszem **OK**



4. Dodanie klasy **typu Entity** przechowującej dane produktu i wyznaczającej cenę brutto (po wyborze projektu w oknie **Projects** należy kolejno kliknąć na **New/Other**)



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



Dodanie klasy typu **Entity** przechowującej dane produktu i wyznaczającej cenę brutto- **Class Name: Produkt1, Package: warstwa\_biznesowa.entity**; 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:


Project:

Location:

Package:

Created File:

Primary Key Type:  ...

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

Create Persistence Unit

< Back   Next >   **Finish**   Cancel   Help

# Wygenerowania klasa o nazwie Produkt1 typu Entity, reprezentująca obiektowy model danych aplikacji EE

The screenshot displays the NetBeans IDE 8.1 interface. On the left, the 'Files' pane shows a project structure for 'Sklep\_1', including 'Web Pages', 'Source Packages', and 'Libraries'. The 'Source Packages' folder is expanded to show the 'warstwa\_biznesowa.entity' package, which contains the 'Produkt1.java' file. The main editor window shows the source code of 'Produkt1.java' with the following content:

```
1 package warstwa_biznesowa.entity;
2
3 import java.io.Serializable;
4 import javax.persistence.Entity;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.GenerationType;
7 import javax.persistence.Id;
8
9 @Entity
10 public class Produkt1 implements Serializable {
11     private static final long serialVersionUID = 1L;
12     @Id
13     @GeneratedValue(strategy = GenerationType.AUTO)
14     private Long id;
15     public Long getId() { return id; }
16     public void setId(Long id) { this.id = id; }
17     @Override
18     public int hashCode() {
19         int hash = 0;
20         hash += (id != null ? id.hashCode() : 0);
21         return hash;
22     }
23     @Override
24     public boolean equals(Object object) {
25         if (!(object instanceof Produkt1)) {
26             return false;
27         }
28         Produkt1 other = (Produkt1) object;
29         if ((this.id == null && other.id != null) ||
30             (this.id != null && !this.id.equals(other.id))) {
31             return false;
32         }
33         return true;
34     }
35     @Override
36     public String toString() {
37         return "warstwa_biznesowa.entity.Produkt1[ id=" + id + " ]";
38     }
39 }
```

The 'Output' pane at the bottom is empty. The status bar at the bottom right shows '3:1' and 'INS'.

## Wygenerowany kod klasy **Produkt1**

```
package warstwa_biznesowa.entity;

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

@Entity
public class Produkt1 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 Produkt1)) {  
        return false;  
    }  
    Produkt1 other = (Produkt1) object;  
    if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {  
        return false;  
    }  
    return true;  
}
```

```
@Override
```

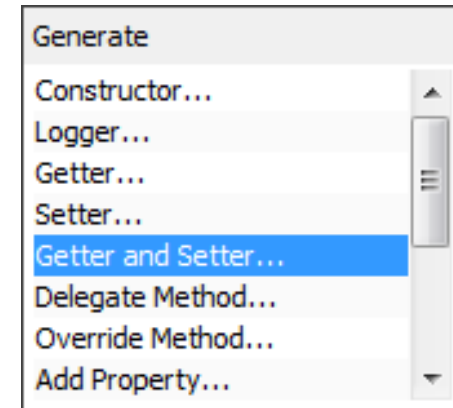
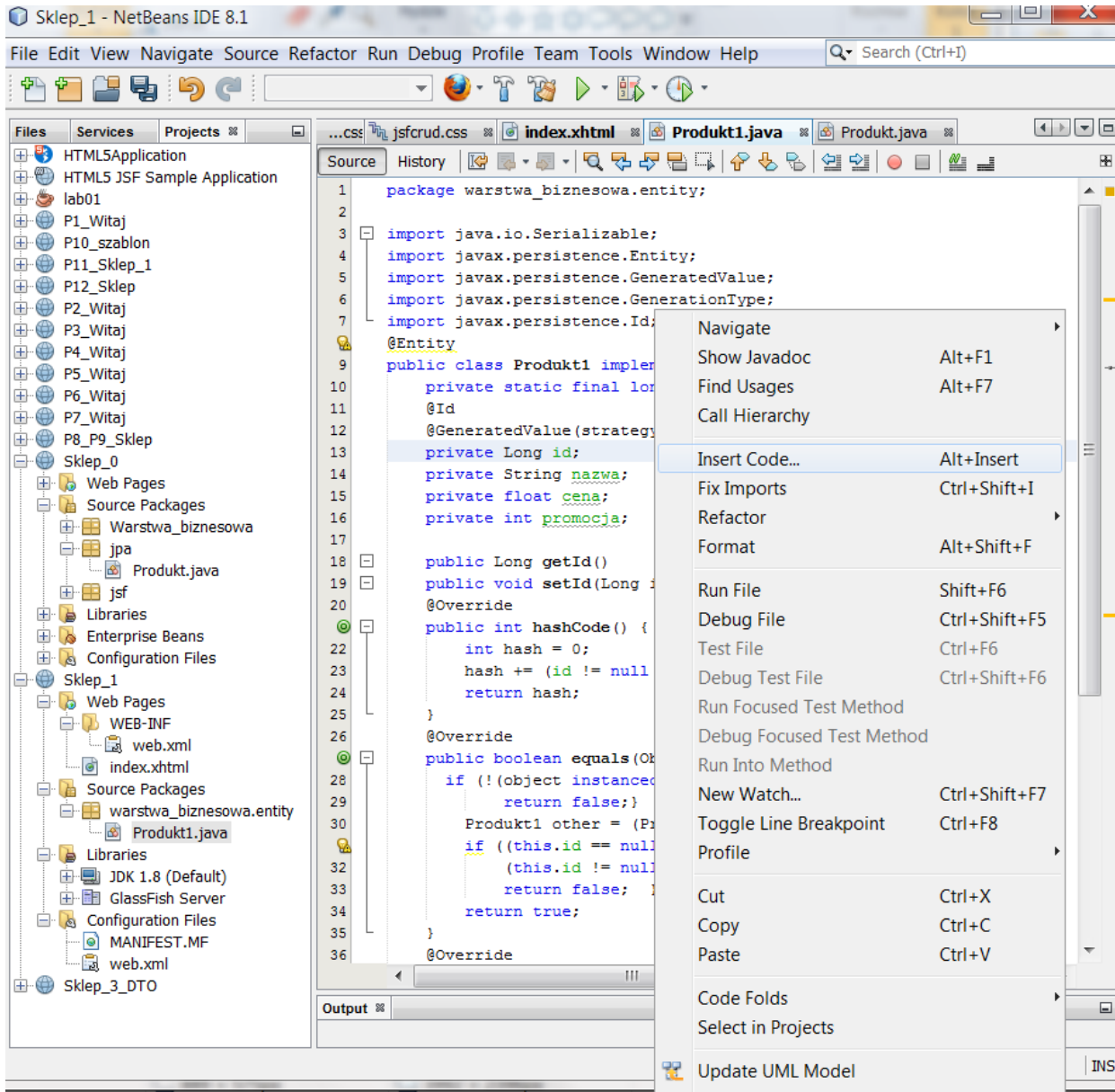
```
public String toString() {  
    return "warstwa_biznesowa.entity.Produkt1[ id=" + id + " ]";  
}
```

```
}
```

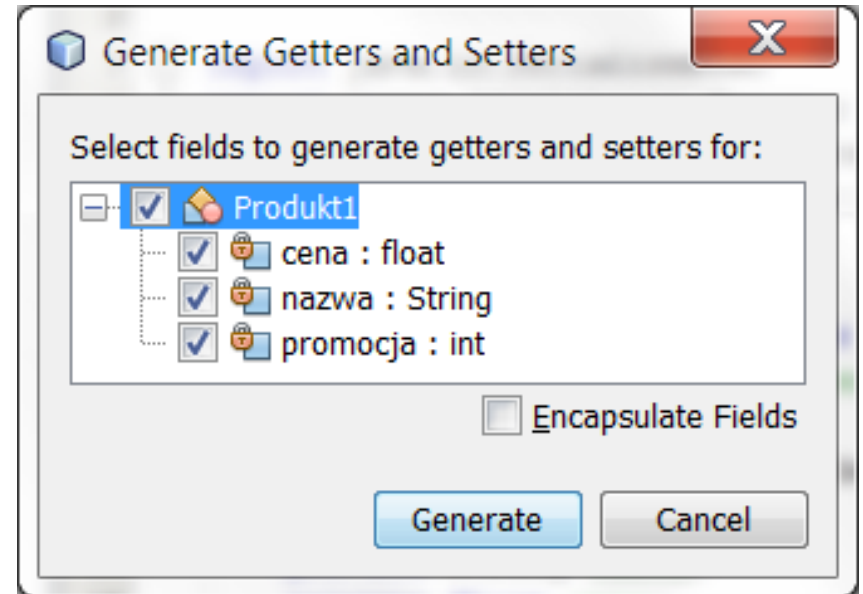
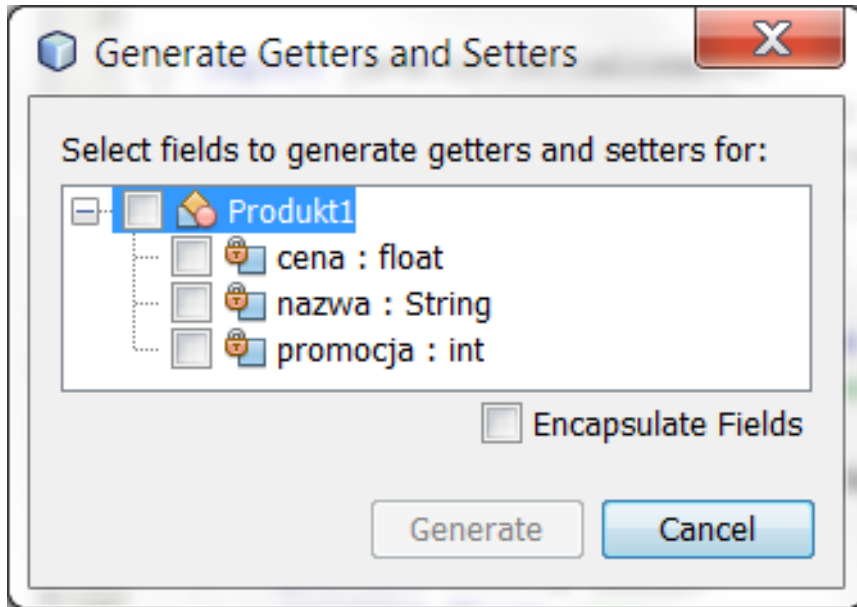


```
private String nazwa;  
private float cena;  
private int promocja;
```

Uzupełnienie kodu klasy Produkt 1- należy dodać atrybuty podane z lewej strony i po kliknięciu prawym klawiszem myszy na okno edytora w obrębie ciała klasy wybrać pozycję **Insert Code/ Getter and Setter...**



## Zaznaczyć atrybuty do wygenerowania metod dostępu do atrybutów



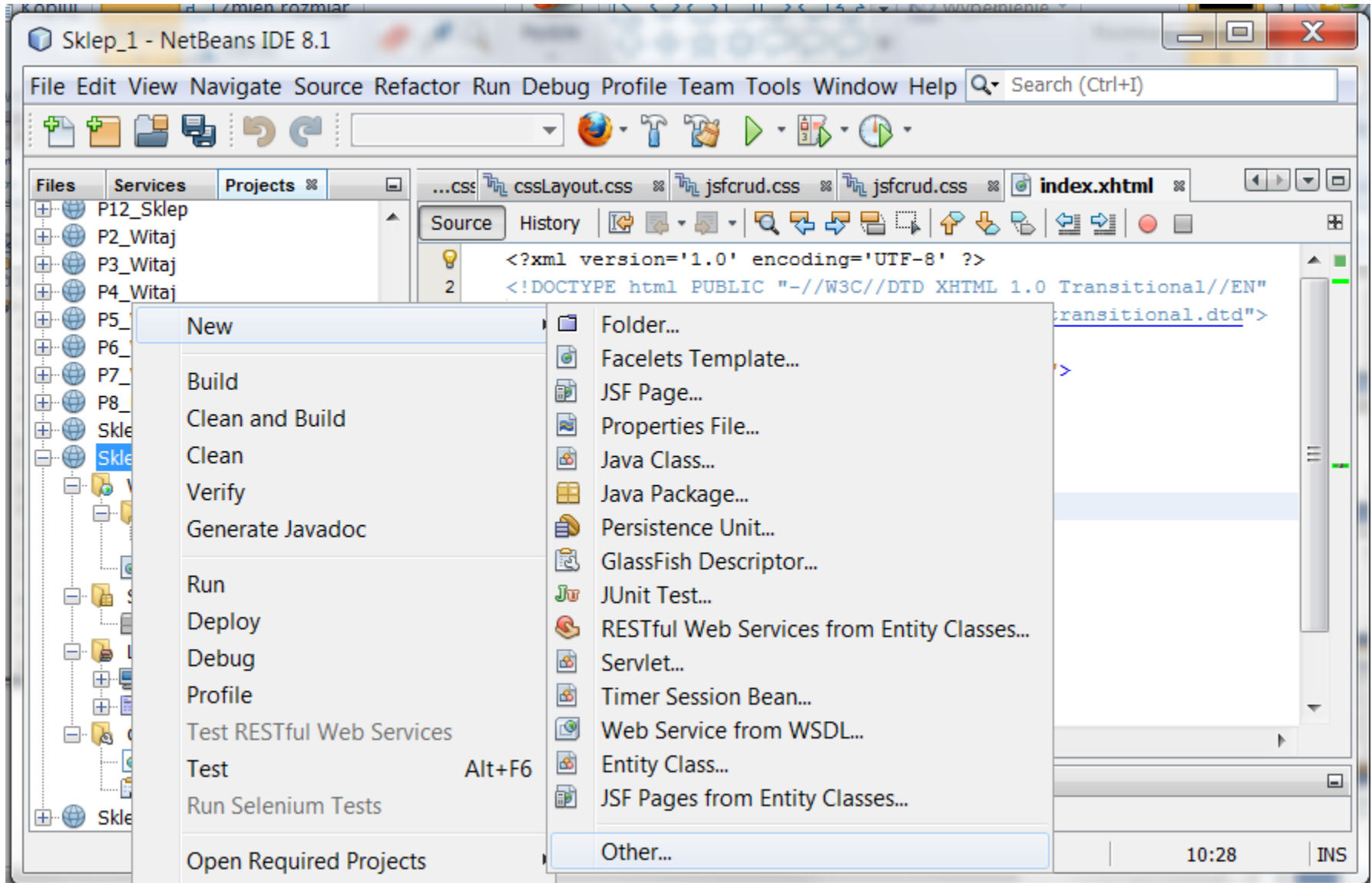
Oprócz wygenerowanych metod należy dodać metodę **cena\_brutto()** wyznaczającą cenę brutto na podstawie ceny netto i promocji

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

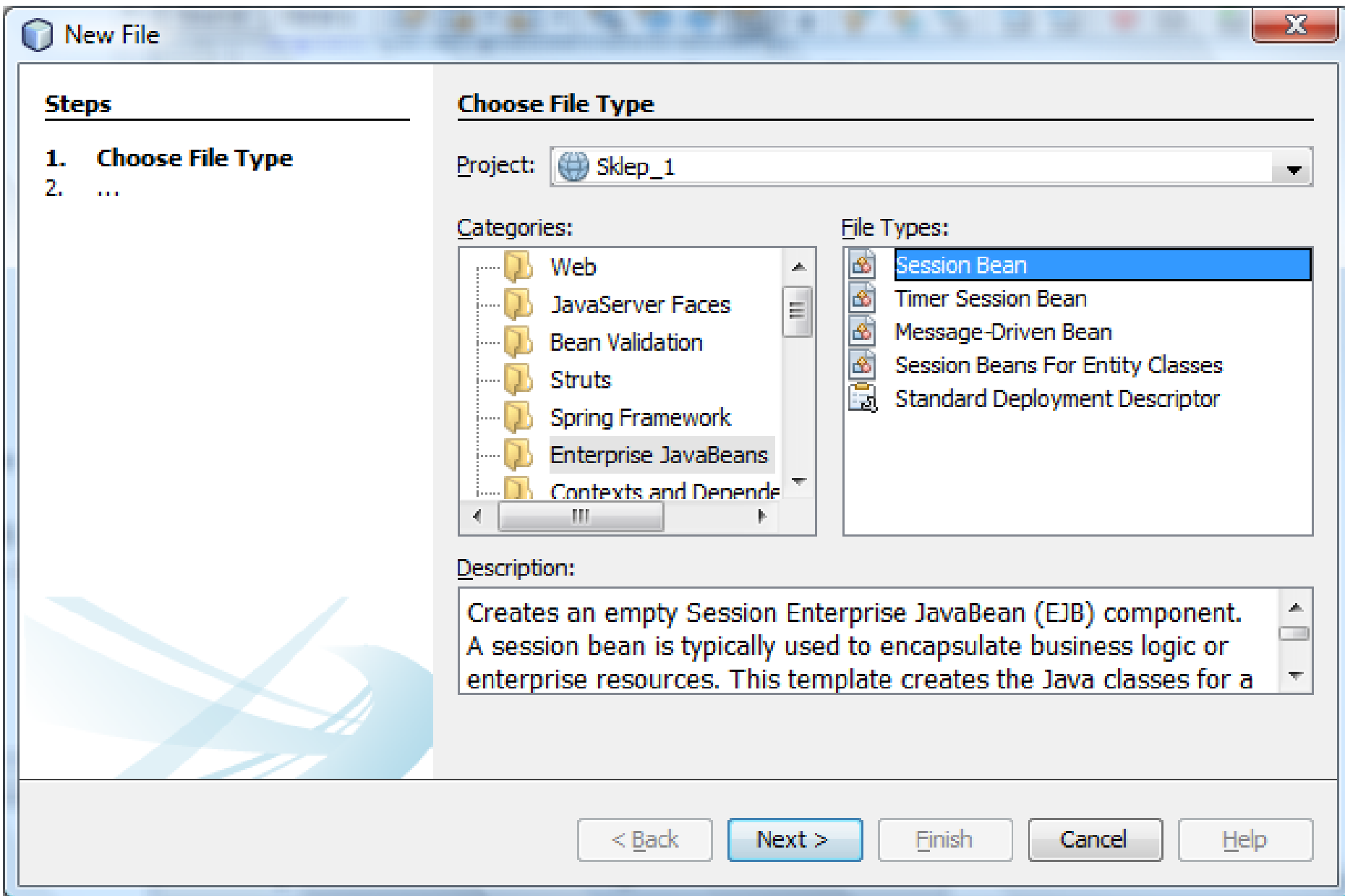
public String getNazwa()           { return nazwa;           }
public void setNazwa(String nazwa) { this.nazwa = nazwa;   }
public float getCena()             { return cena;         }
public void setCena(float cena)    { this.cena = cena;    }
public int getPromocja()           { return promocja;      }
public void setPromocja(int promocja) { this.promocja = promocja; }

public float cena_brutto ()
{
    float cena_brutto= cena*(1-(float)promocja/100);
    return cena_brutto;
}
```

## 5. Należy dodać ziarno EJB do przetwarzania obiektu typu Entity (Produkt1) – **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**

**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

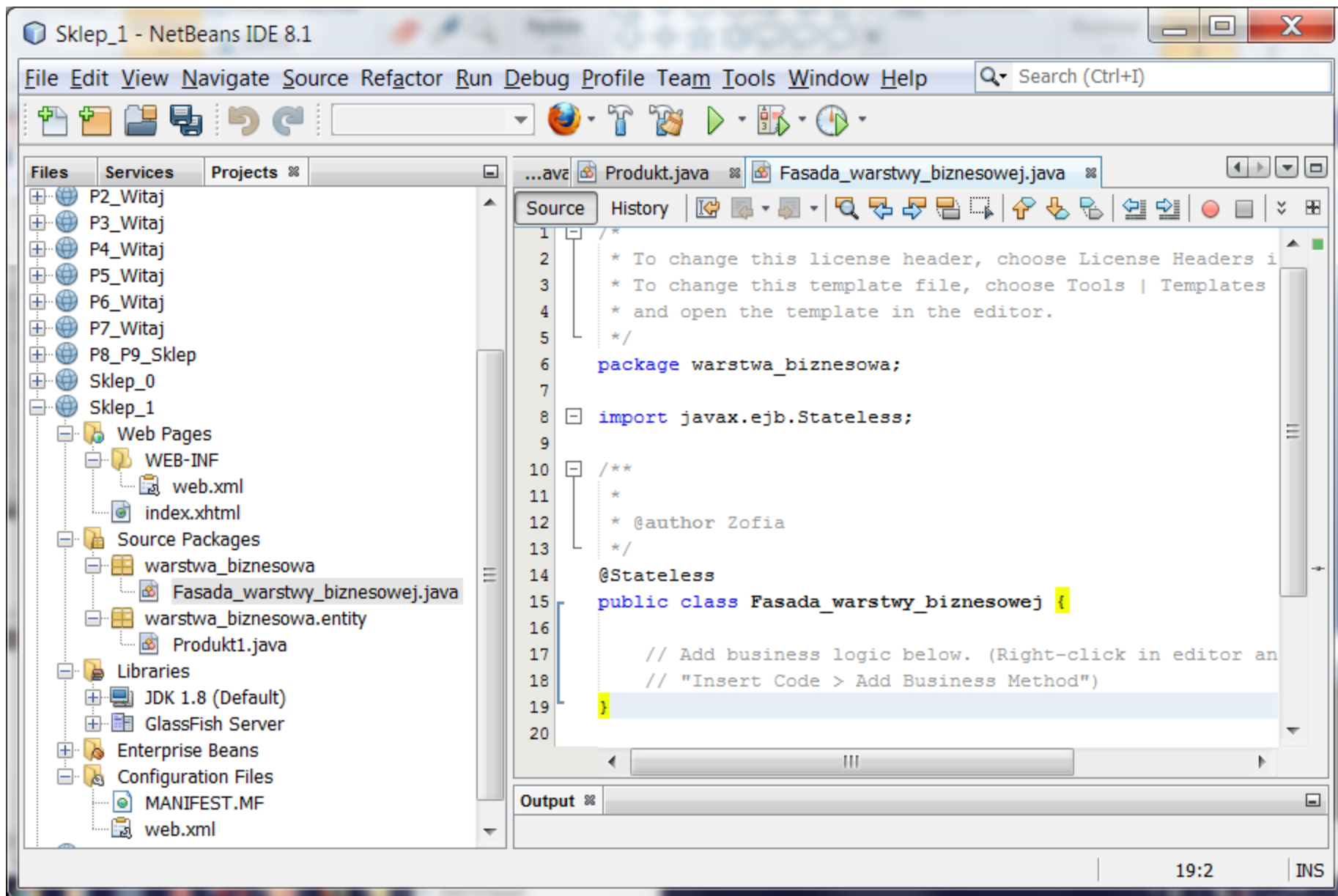
Singleton

Create Interface:

Local

< Back   Next >   **Finish**   Cancel   Help

# Kod wygenerowany ziarna EJB do przetwarzania obiektu typu Entity



The screenshot shows the NetBeans IDE 8.1 interface. The left sidebar displays a project tree for 'Sklep\_1', with 'Fasada\_warstwy\_biznesowej.java' selected under the 'warstwa\_biznesowa' package. The main editor window shows the source code of this file, which is a stateless EJB facade. The code includes a package declaration, an import for javax.ejb.Stateless, and a class declaration for Fasada\_warstwy\_biznesowej.

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

The status bar at the bottom right indicates the cursor is at line 19, column 2, in the INS (Insert) mode.

Uzupełniono kod klasy **Fasada\_warstwy\_biznesowej**: dodano atrybut produkt typu **Produkt1**, metody typu **set** i **get** (za pomocą opcji **Insert Code** - (slajdy 23-25)) 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 warstwa_biznesowa.entity.Produkt1;
```

```
@Stateless
```

```
public class Fasada_warstwy_biznesowej {
```

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

```
    private Produkt1 produkt;
```

```
    public Produkt1 getProdukt() {  
        return produkt;  
    }
```

```
    public void setProdukt(Produkt1 produkt) {  
        this.produkt = produkt;  
    }
```



```
//wykonanie obiektu typu Produkt1
```

```
public void utworz_produkt(String dane[]) {  
    produkt = new Produkt1();  
    produkt.setNazwa(dane[0]);  
    produkt.setCena(Float.parseFloat(dane[1]));  
    produkt.setPromocja(Integer.parseInt(dane[2]));  
}
```

```
//wykonanie modelu obiektu typu Produkt1
```

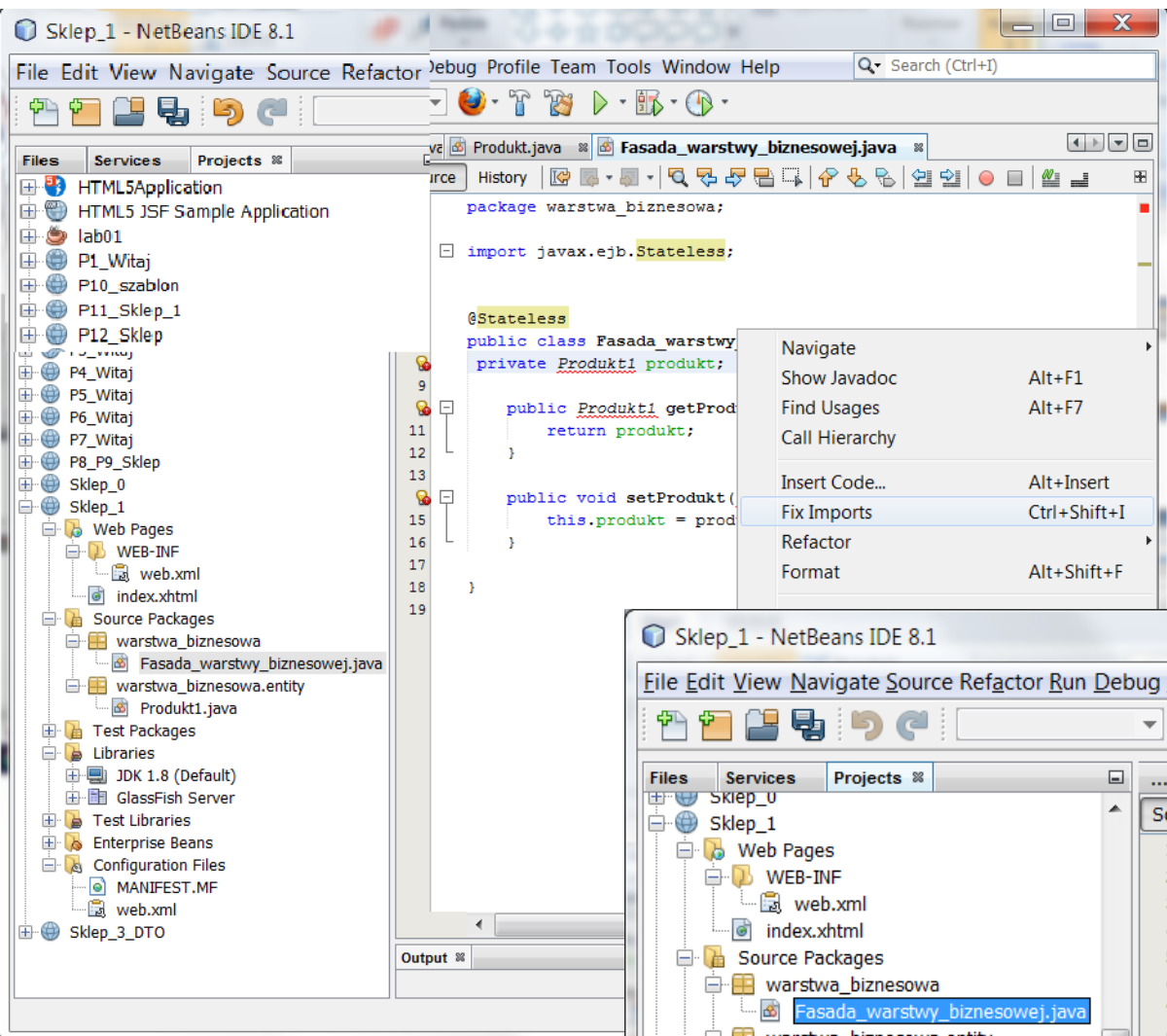
```
public String[] dane_produktu() {  
    String nazwa = "brak produktu";  
    String cena = "brak produktu";  
    String promocja = "brak produktu";  
    String cena_brutto = "brak produktu";  
    if (produkt != null) {
```

```
//tworzenie tablicy łańcuchów (czyli obiektów typu String) z atrybutów obiektu typu Produkt1
```

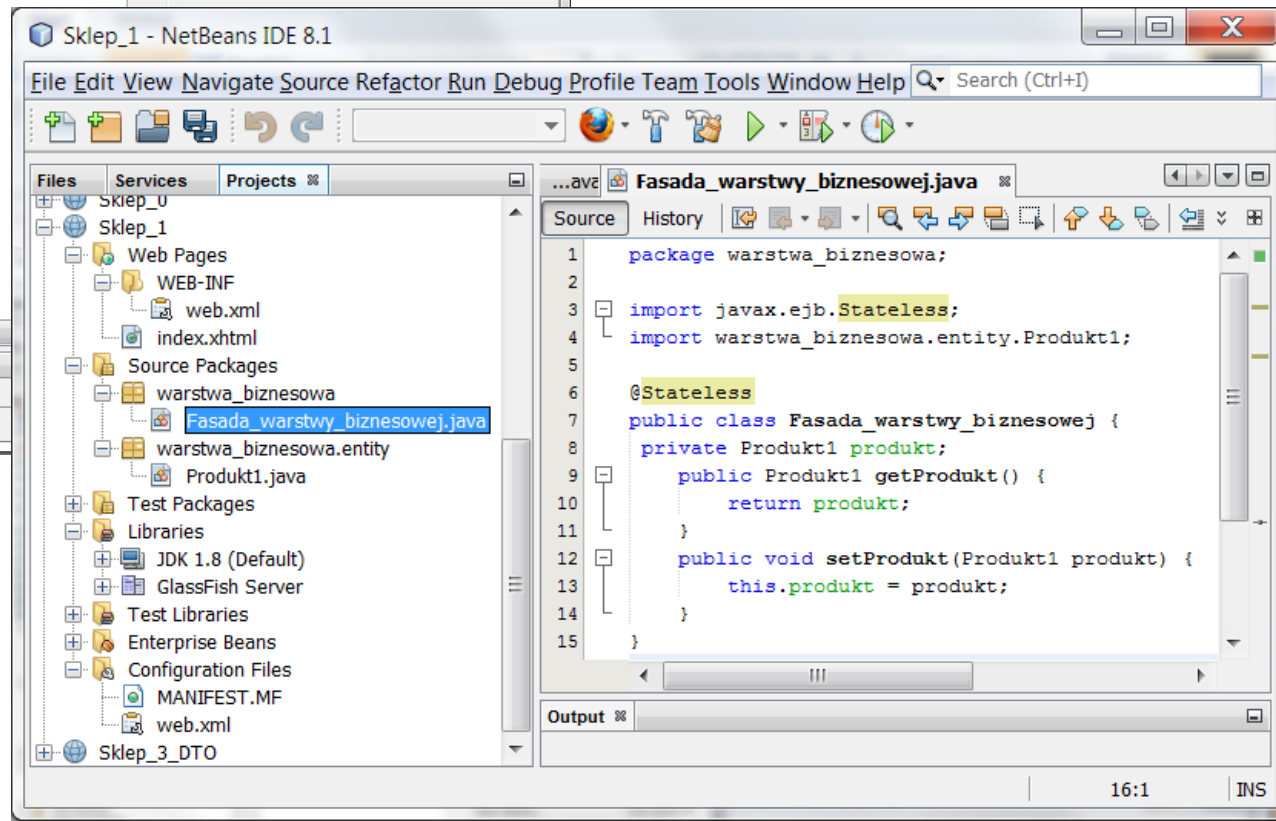
```
        nazwa = produkt.getNazwa();  
        cena = "" + produkt.getCena();  
        promocja = "" + produkt.getPromocja();  
        cena_brutto = "" + produkt.cena_brutto();
```

```
    }  
    String dane[] = {nazwa, cena, promocja, cena_brutto};  
    return dane;
```

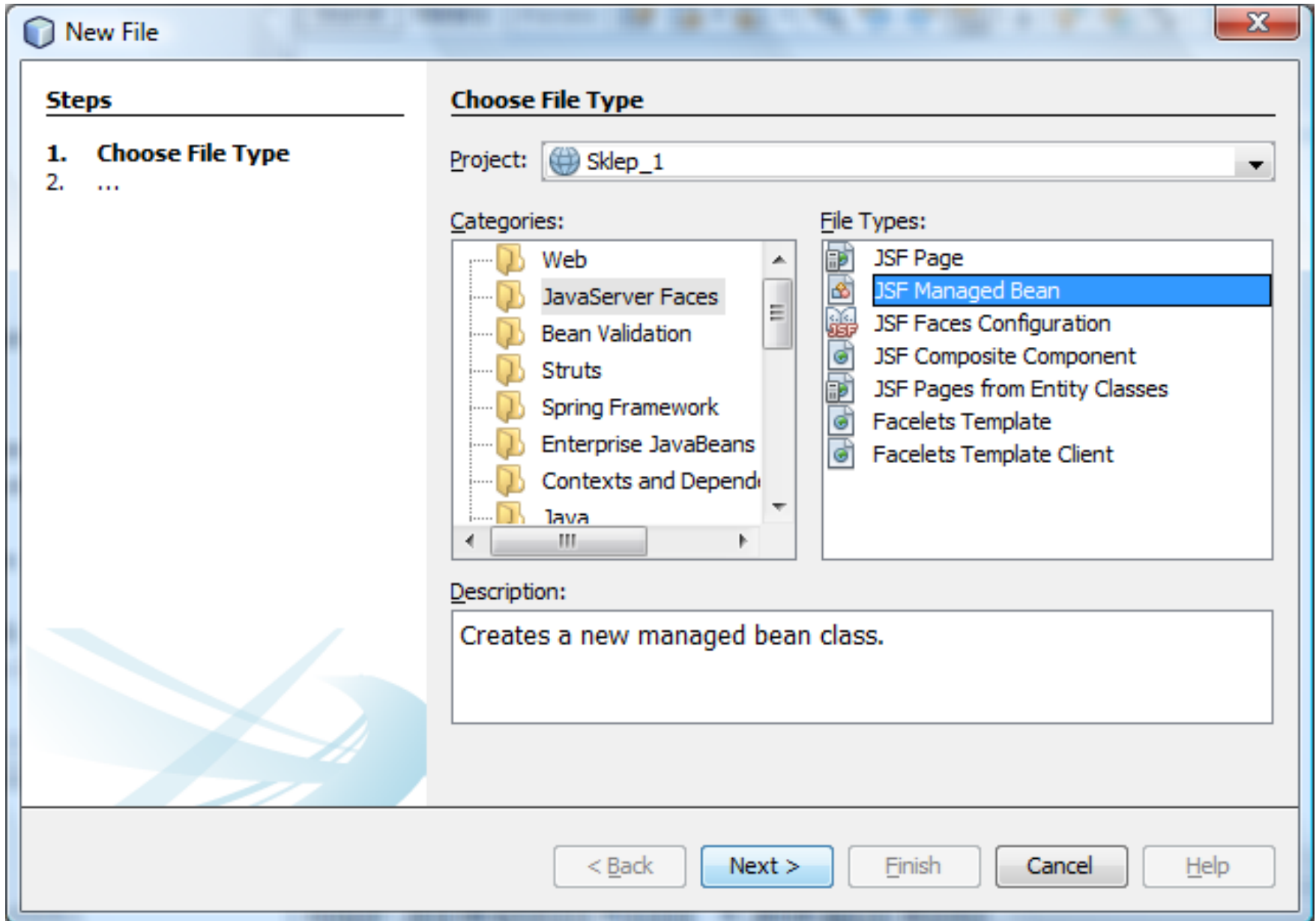
```
    }  
}
```



Sposób dodania  
brakujących importów  
klasy Produkt1: kliknąć  
na powierzchnię edytora  
(na ciele klasy) prawym  
klawiszem myszy i  
wybrać pozycję **Fix  
Imports**



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



# Dodanie klasy typu Managed Bean: **Name: Managed\_produk**t, **Package: warstwa\_internetowa**, **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: warstwa\_internetowa

Created File: C:\Studia\Szkola\TINT\Sklep\_1\src\java\warstwa\_internetowa\Managed\_produk.java

Add data to configuration file

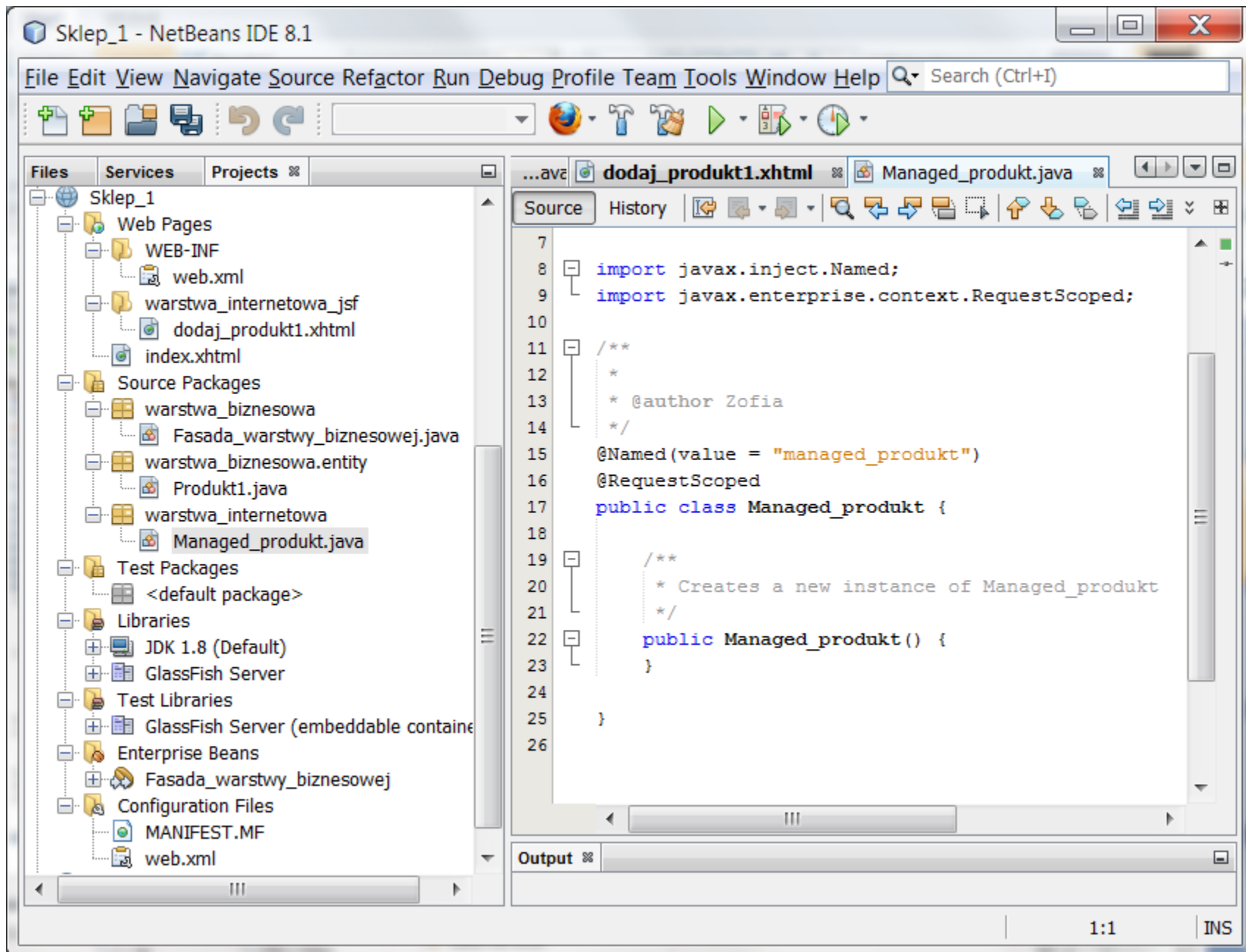
Configuration File:

Name: managed\_produk

Scope: request

< Back   Next >   **Finish**   Cancel   Help

## Wygenerowany kod klasy typu **ManagedBean**, czyli **Managed\_produk**



The screenshot displays the NetBeans IDE 8.1 interface. The left-hand pane shows a project tree for 'Sklep\_1' with various folders and files, including 'warstwa\_internetowa' and 'Managed\_produk.java'. The main editor window shows the source code of the 'Managed\_produk.java' file. The code includes imports for 'javax.inject.Named' and 'javax.enterprise.context.RequestScoped', followed by a class definition for 'Managed\_produk' with a constructor.

```
7
8 import javax.inject.Named;
9 import javax.enterprise.context.RequestScoped;
10
11 /**
12  *
13  * @author Zofia
14  */
15 @Named(value = "managed_produk")
16 @RequestScoped
17 public class Managed_produk {
18
19     /**
20      * Creates a new instance of Managed_produk
21      */
22     public Managed_produk() {
23     }
24
25 }
26
```

The bottom right corner of the IDE shows the 'Output' window and the status bar with '1:1' and 'INS'.

7. Przebieg dodawania ziarna typu **EJB** do obiektu typu **ManagedBean**, czyli ziarna **Fasada\_warstwy\_biznesowej** do komponentu **Managed\_produk** – należy kliknąć na powierzchnię edytora w obrębie ciała klasy (w miejscu, w którym będzie wstawiany kod) i wybrać kolejno pozycje **Insert Code.../ Call Enterprise Bean...**- na końcu należy wybrać właściwy projekt (bieżący) i wybrać właściwy komponent EJB, czyli ziarno **Fasada\_warstwy\_biznesowej**.

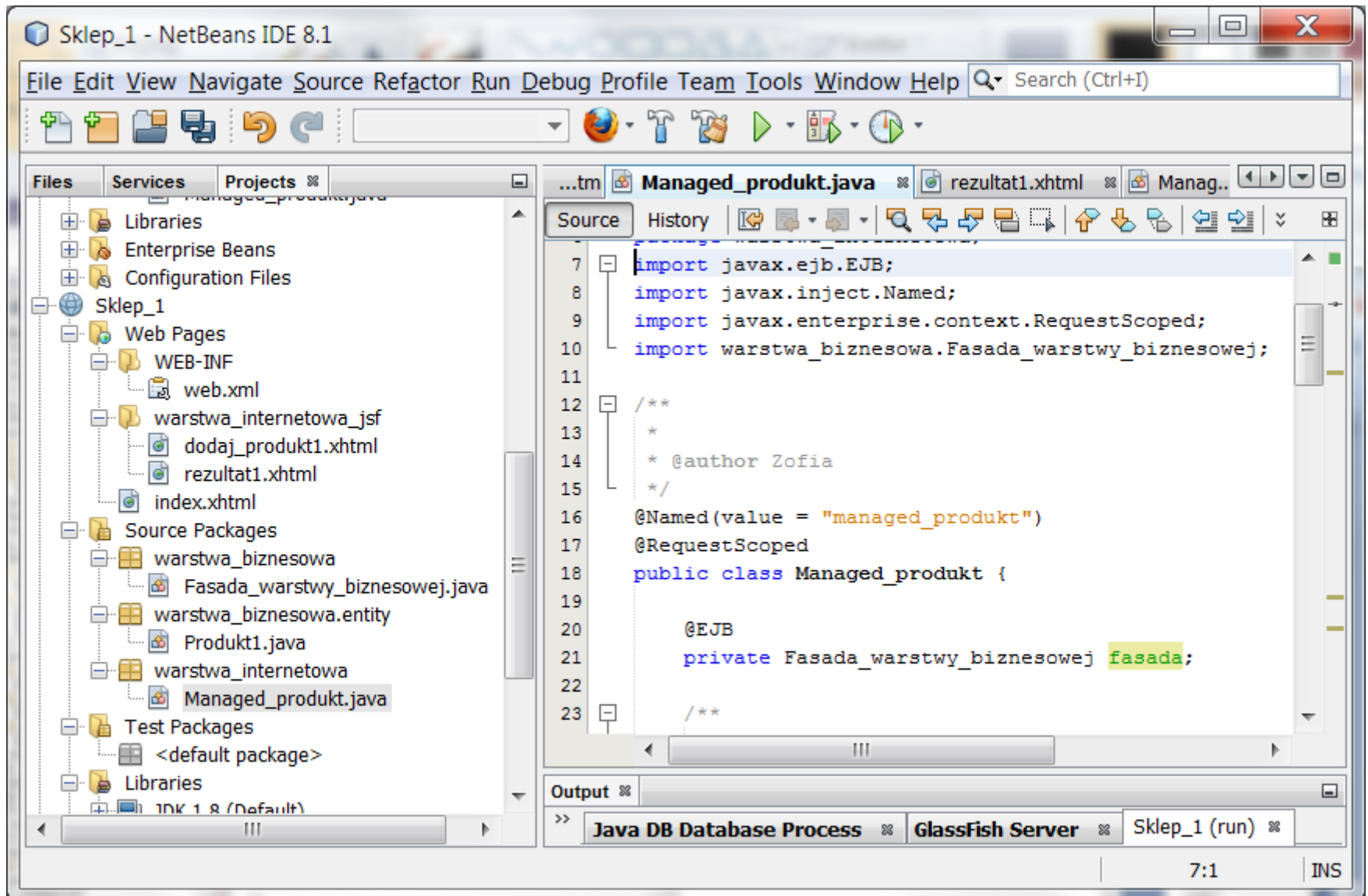
The screenshot illustrates the steps to add an EJB to a ManagedBean in an IDE. The main editor shows the source code of `managed_produk.java` with the following content:

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package warstwa_internetowa;
7
8
9  import javax.inject.Named;
10 import javax.enterprise.context.RequestScoped;
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

A context menu is open over the `import javax.enterprise.context.RequestScoped;` line, with the **Call Enterprise Bean...** option selected. A secondary menu is open over this option, listing various actions, with **Call Enterprise Bean...** highlighted.

To the right, a dialog box titled **Call Enterprise Bean** is open, showing a list of available beans. **Fasada\_warstwy\_biznesowej** is selected in the list. Below the list, the **Reference Name** is set to `Fasada_warstwy_biznesow` and **Referenced Interface** is set to **No in...**. The dialog has **OK**, **Cancel**, and **Help** buttons.

Wynik dodania ziarna typu **EJB** do obiektu typu **ManagedBean**, czyli ziarna **Fasada\_warstwy\_biznesowej** do komponentu **Managed\_produk**t.





```

package warstwa_internetowa;

import javax.ejb.EJB;
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import warstwa_biznesowa.Fasada_warstwy_biznesowej;

@Named(value = "managed_produkt")
@RequestScoped
public class Managed_produkt {

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

    public Managed_produkt() {
    }

    public Fasada_warstwy_biznesowej getFasada() {
        return fasada;
    }

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

```

Dodany kod do klasy **Managed\_produkt** (slajd 36):

**@EJB**

**private Fasada\_warstwy\_biznesowej fasada;**

Należy dodać następujące atrybuty:

**private String nazwa;**

**private String cena;**

**private String promocja;**

**private String cena\_brutto;**

bindowane z komponentami stron JSF.

Metody dostępu do tych atrybutów dodano za pomocą pozycji **Insert Code** (slajdy 23-25) – wynik na następnym slajdzie



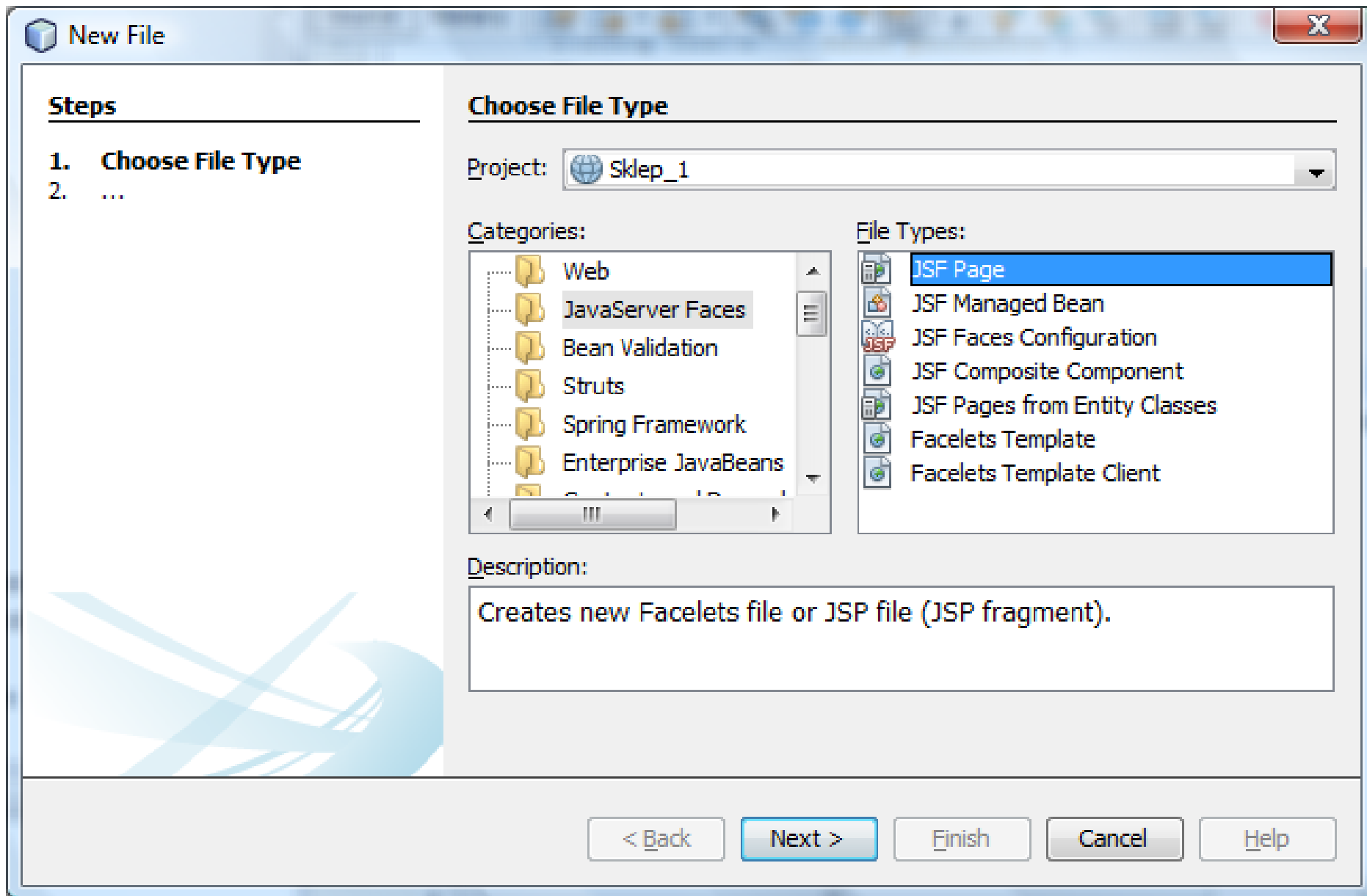
```
public String getNazwa() {
    return nazwa;
}
public void setNazwa(String nazwa) {
    this.nazwa = nazwa;
}
public String getCena() {
    return cena;
}
public void setCena(String cena) {
    this.cena = cena;
}
public String getPromocja() {
    return promocja;
}
public void setPromocja(String 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\_produkct obsługujących dodawanie produktu (**dodaj\_produkct**) po pobraniu danych z formularza za pomocą atrybutów: nazwa, cena, promocja . **Metoda musi zostać dokończona w dalszej części instrukcji!**

```
public String dodaj_produkct() {  
    String[] dane = {nazwa, cena, promocja};  
    return "rezultat1";  
}  
  
}
```

8. 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\_produk1**, **Folder:** **warstwa\_internetowa\_jsf** i **Finish**

**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:

< Back   Next >   **Finish**   Cancel   Help

## Uzupełniona treść strony **dodaj\_produkt1**

```
<?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://xmlns.jcp.org/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**.

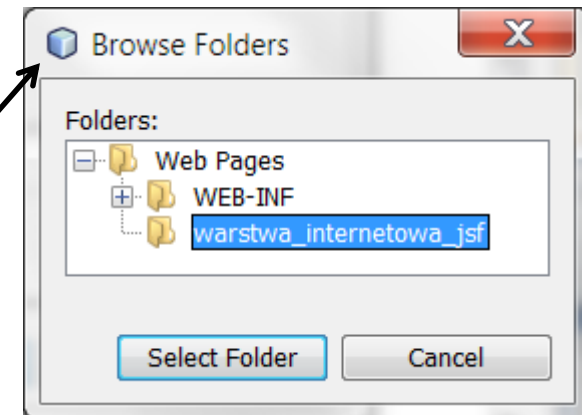
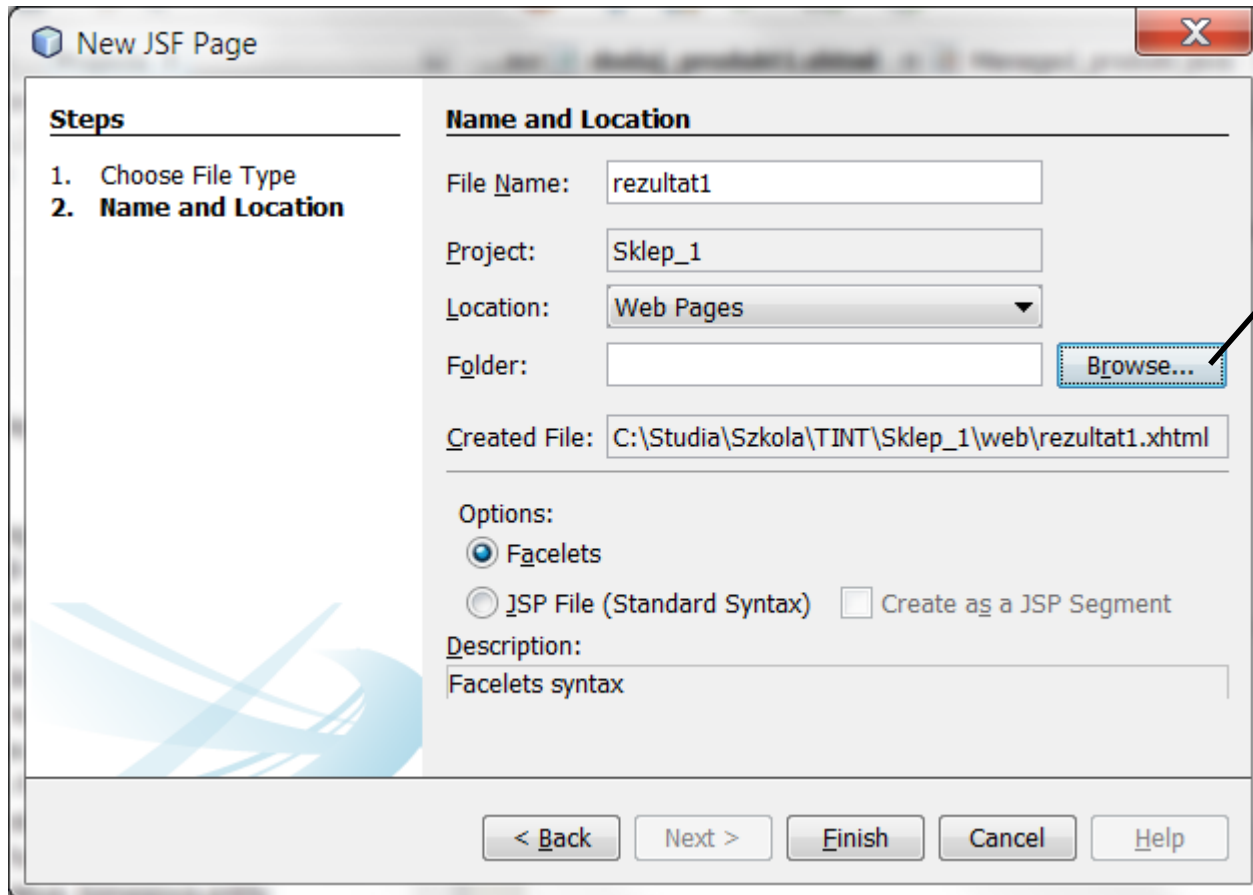
Atrybuty **required** i **requiredMessage** obsługują błąd wynikający z braku wprowadzenia danych do komponentów typu **inputText**

```
<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**:

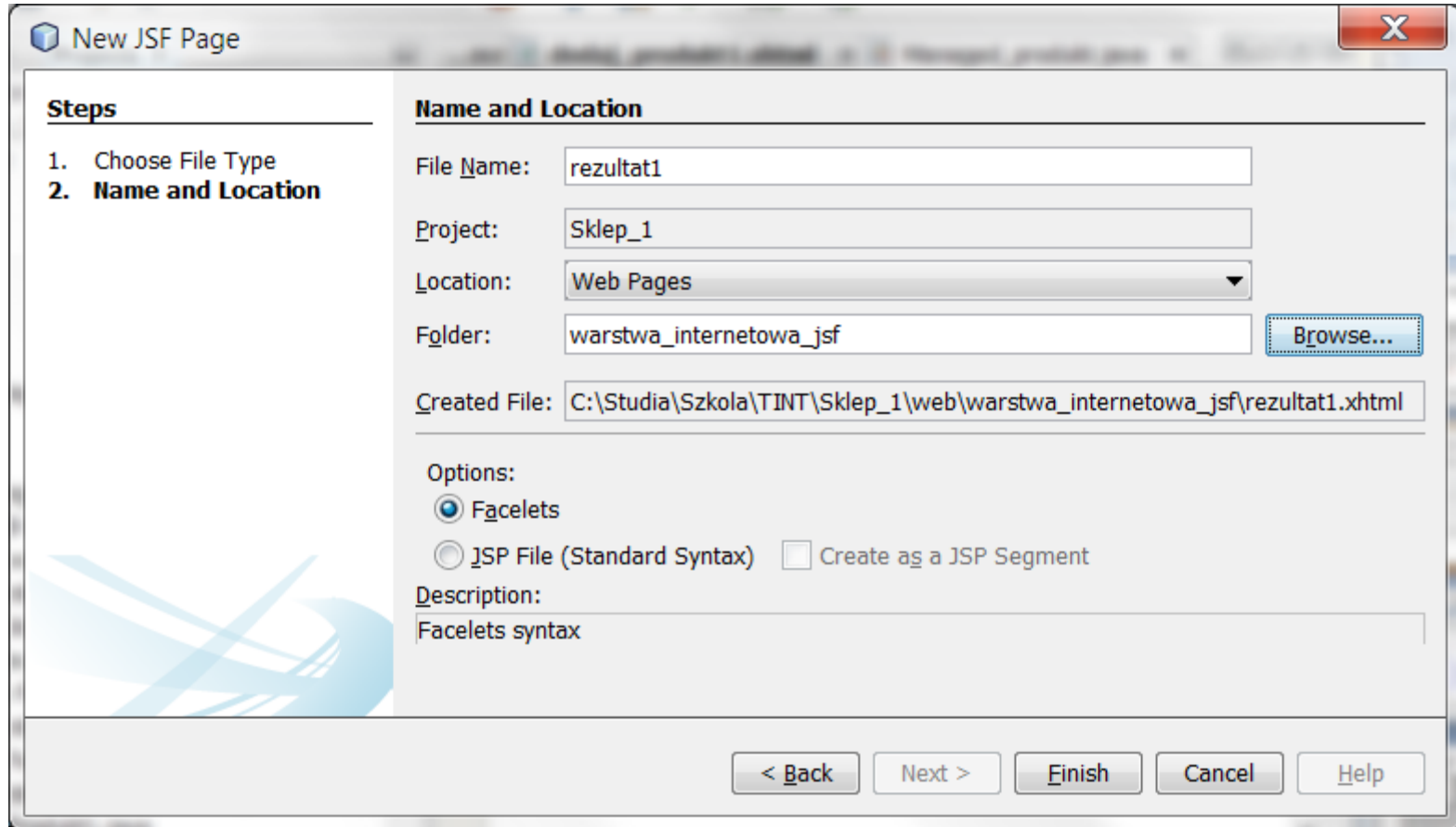
```
public String dodaj_produkt() {
    String[] dane = {nazwa, cena, promocja};
    return "rezultat1";
}
```

9. Należy dodać stronę **rezultat1** w folderze **warstwa\_internetowa\_jsf** – **New/Other/JavaServer Faces/JSF Page**, File Name- rezultat1, Folder: warstwa\_internetowa\_jsf

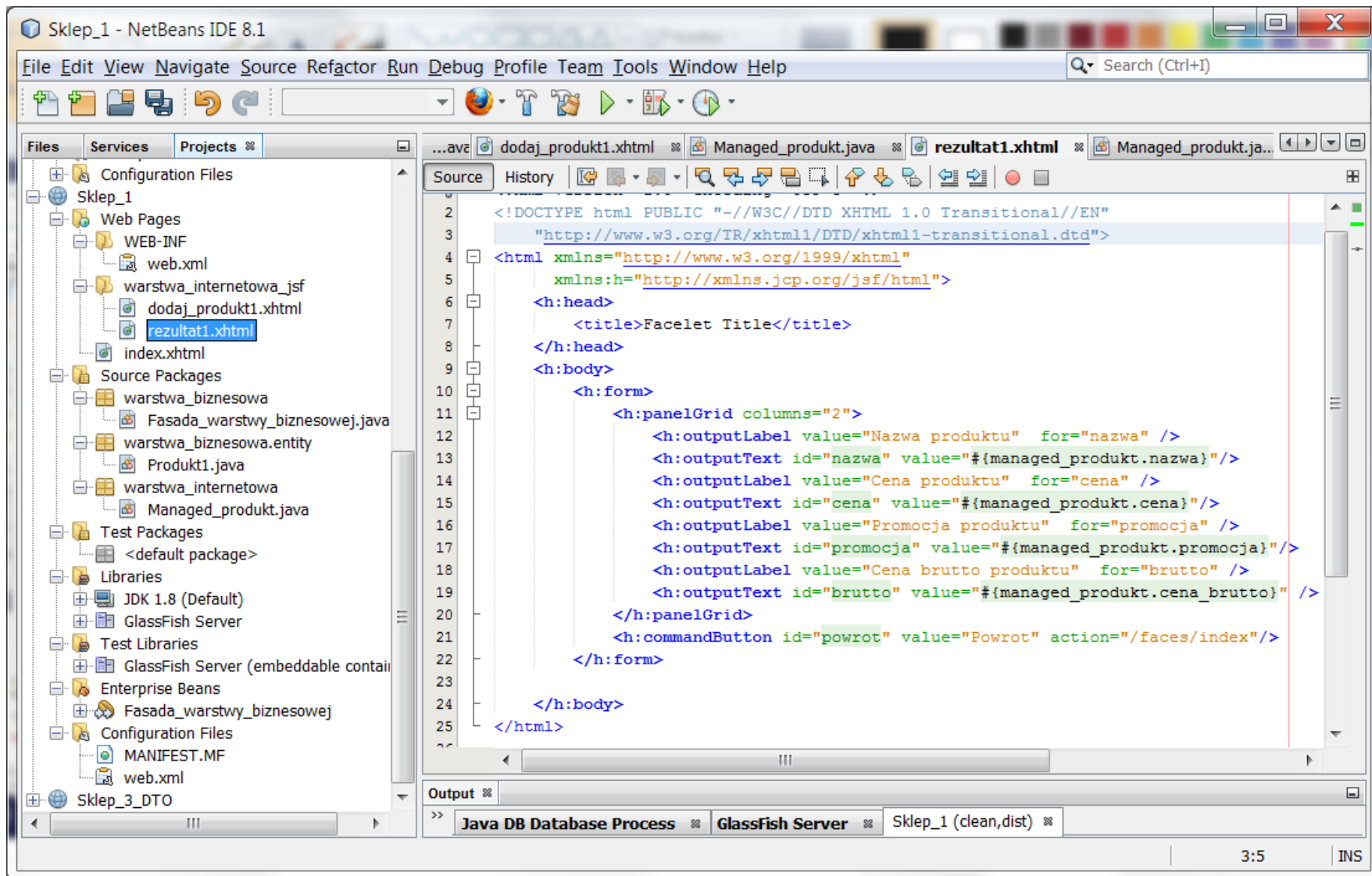




Należy dodać stronę **rezultat1** w folderze warstwa\_internetowa\_jsf – **New/Other/JavaServer Faces/JSF Page, File Name- rezultat1, Folder: warstwa\_internetowa\_jsf**



Do strony **rezultat1.xhtml** dodano kod JSF do prezentacji danych produktu oraz ceny brutto, pobieranych z atrybutów obiektu **managed\_produk** typu **Managed\_produk**



The screenshot shows the NetBeans IDE 8.1 interface. The left sidebar displays the project structure for 'Sklep\_1', including 'Web Pages' and 'Source Packages'. The main editor window shows the source code of 'rezultat1.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"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml"
5 xmlns:h="http://xmlns.jcp.org/jsf/html">
6 <h:head>
7 <title>Facelet Title</title>
8 </h:head>
9 <h:body>
10 <h:form>
11 <h:panelGrid columns="2">
12 <h:outputLabel value="Nazwa produktu" for="nazwa" />
13 <h:outputText id="nazwa" value="#{managed_produk.nazwa}" />
14 <h:outputLabel value="Cena produktu" for="cena" />
15 <h:outputText id="cena" value="#{managed_produk.cena}" />
16 <h:outputLabel value="Promocja produktu" for="promocja" />
17 <h:outputText id="promocja" value="#{managed_produk.promocja}" />
18 <h:outputLabel value="Cena brutto produktu" for="brutto" />
19 <h:outputText id="brutto" value="#{managed_produk.cena_brutto}" />
20 </h:panelGrid>
21 <h:commandButton id="powrot" value="Powrot" action="/faces/index" />
22 </h:form>
23 </h:body>
24 </html>
```

The bottom of the IDE shows the 'Output' window with the following tabs: 'Java DB Database Process', 'GlassFish Server', and 'Sklep\_1 (clean,dist)'. The status bar at the bottom right indicates '3:5' and 'INS'.

```

<?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://xmlns.jcp.org/jsf/html"
xmlns:f="http://xmlns.jcp.org/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>

```

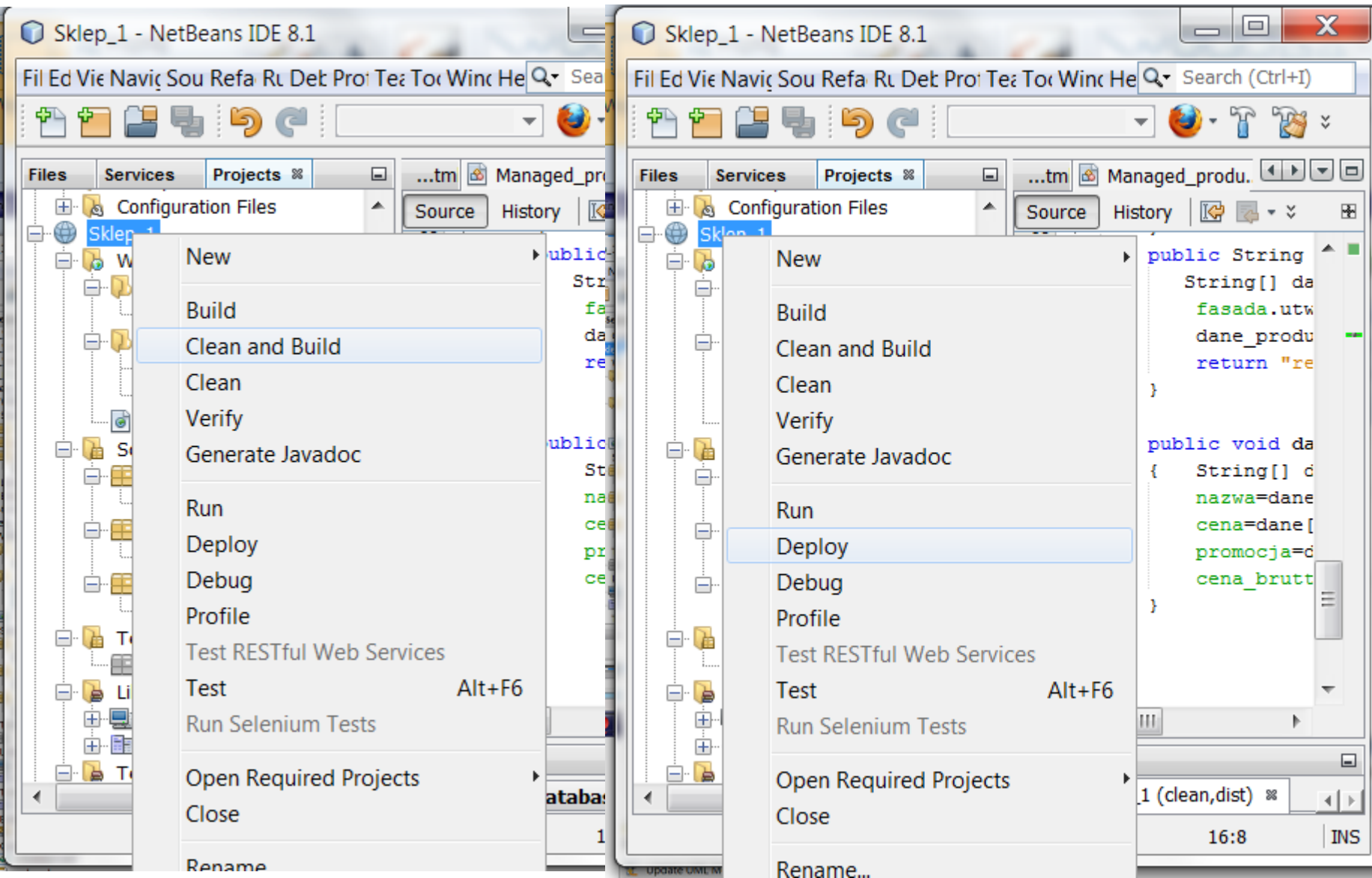
Siatka **panelGrid** umożliwiająca wyświetlanie danych produktu pobranych z obiektu typu `Managed_produkt` w dwóch kolumnach za pomocą komponentów **outputLabel** i **outputText**

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

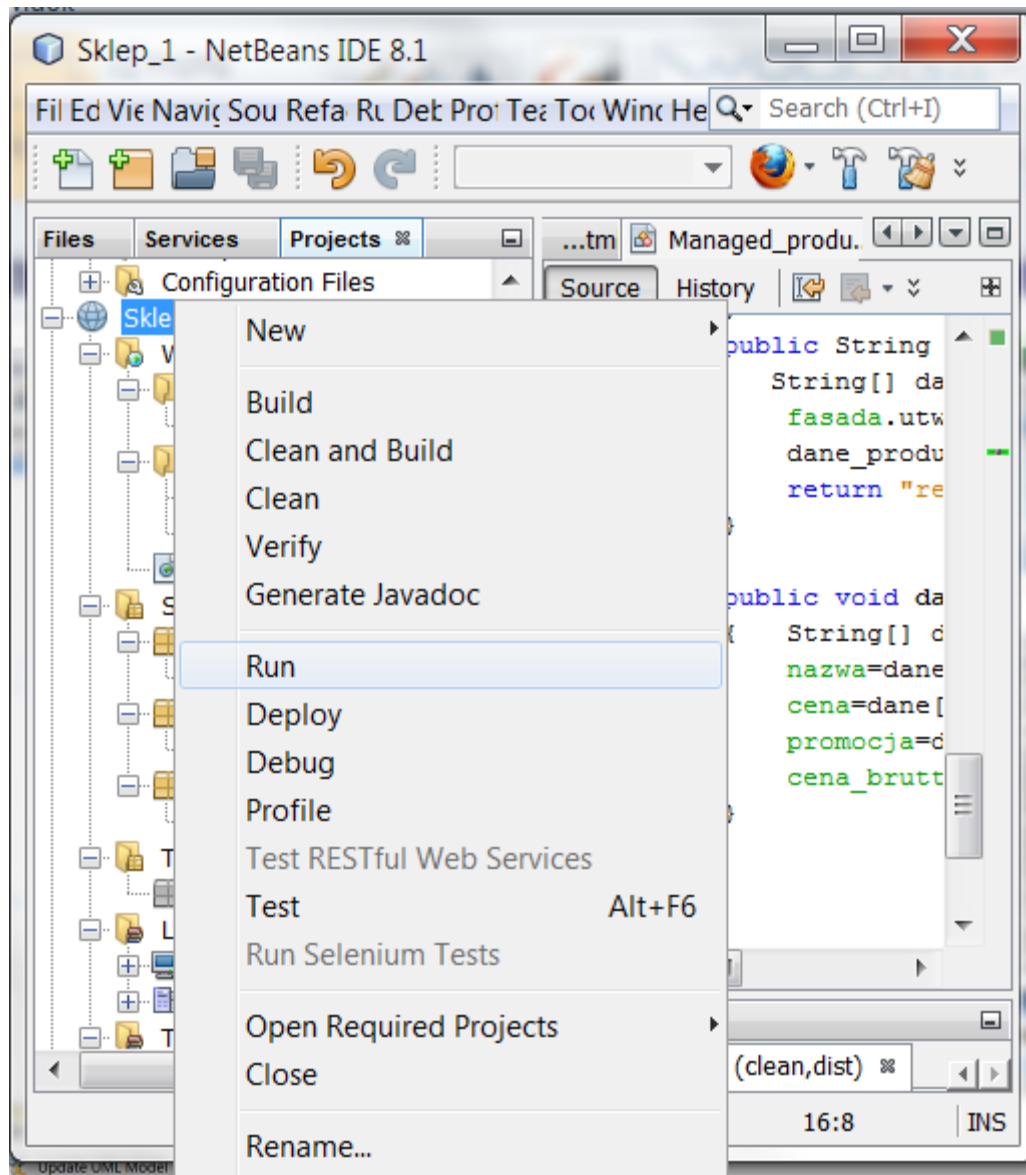
10. Uzupełniono kod klasy głównej **index.xhtml** – znacznik **link** pozwala wywołać stronę **dodaj\_produk.html**

```
<?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://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    <h:link outcome="/warstwa_internetowa_jsf/dodaj_produk1"
            value="Dodaj produkt"/>
  </h:body>
</html>
```

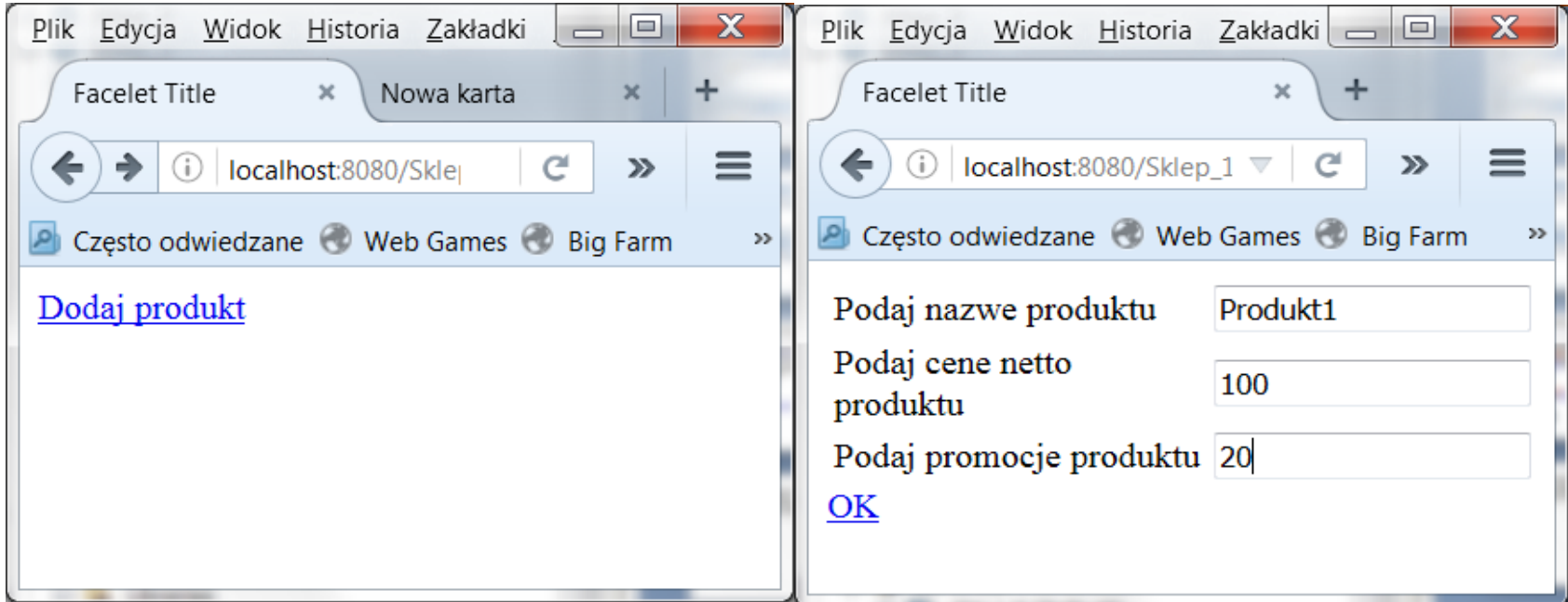
11. Po kliknięciu na nazwę projektu w zakładce **Projects** należy wybrać kolejno pozycje **Clean and Build**, a potem podobnie wybrać pozycję **Deploy**



Uruchomienie aplikacji – za pomocą pozycji **Run** w oknie domyślnej przeglądarki

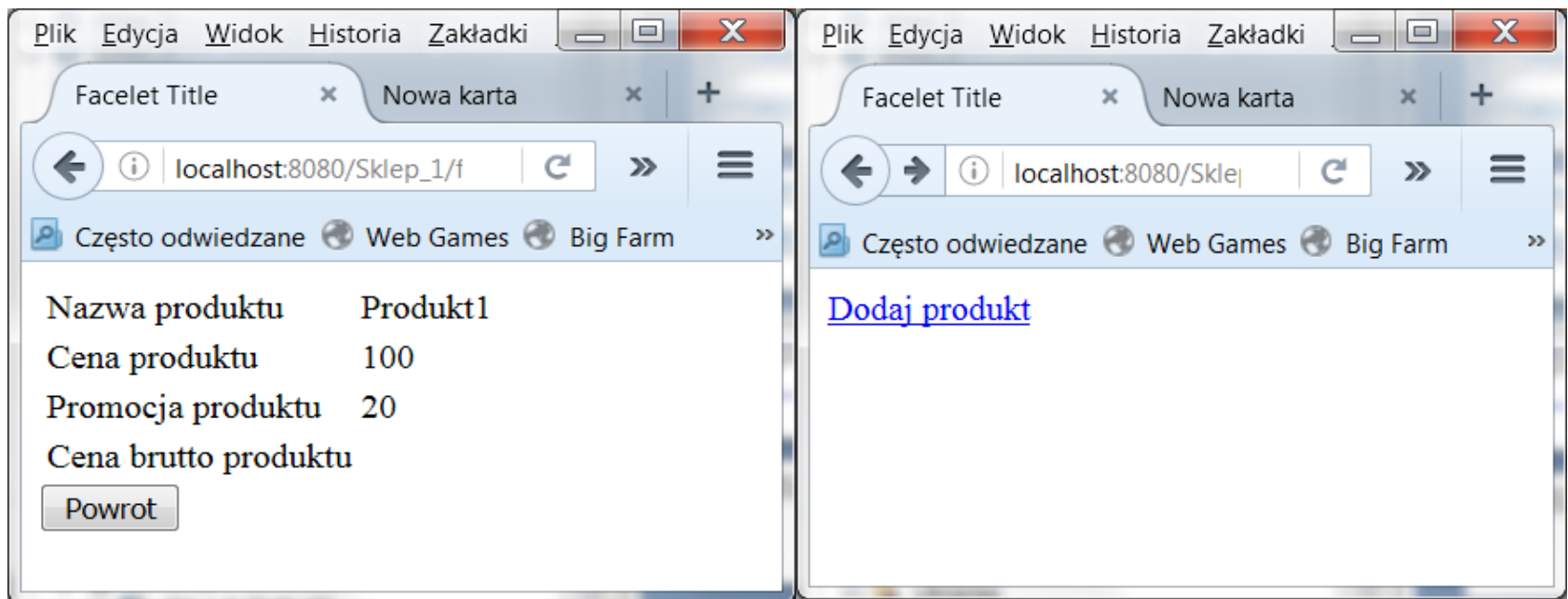


Stany aplikacji – uruchomienie strony głównej. Po wybraniu linku **Dodaj produkt** przejście do strony **dodaj\_produk1**



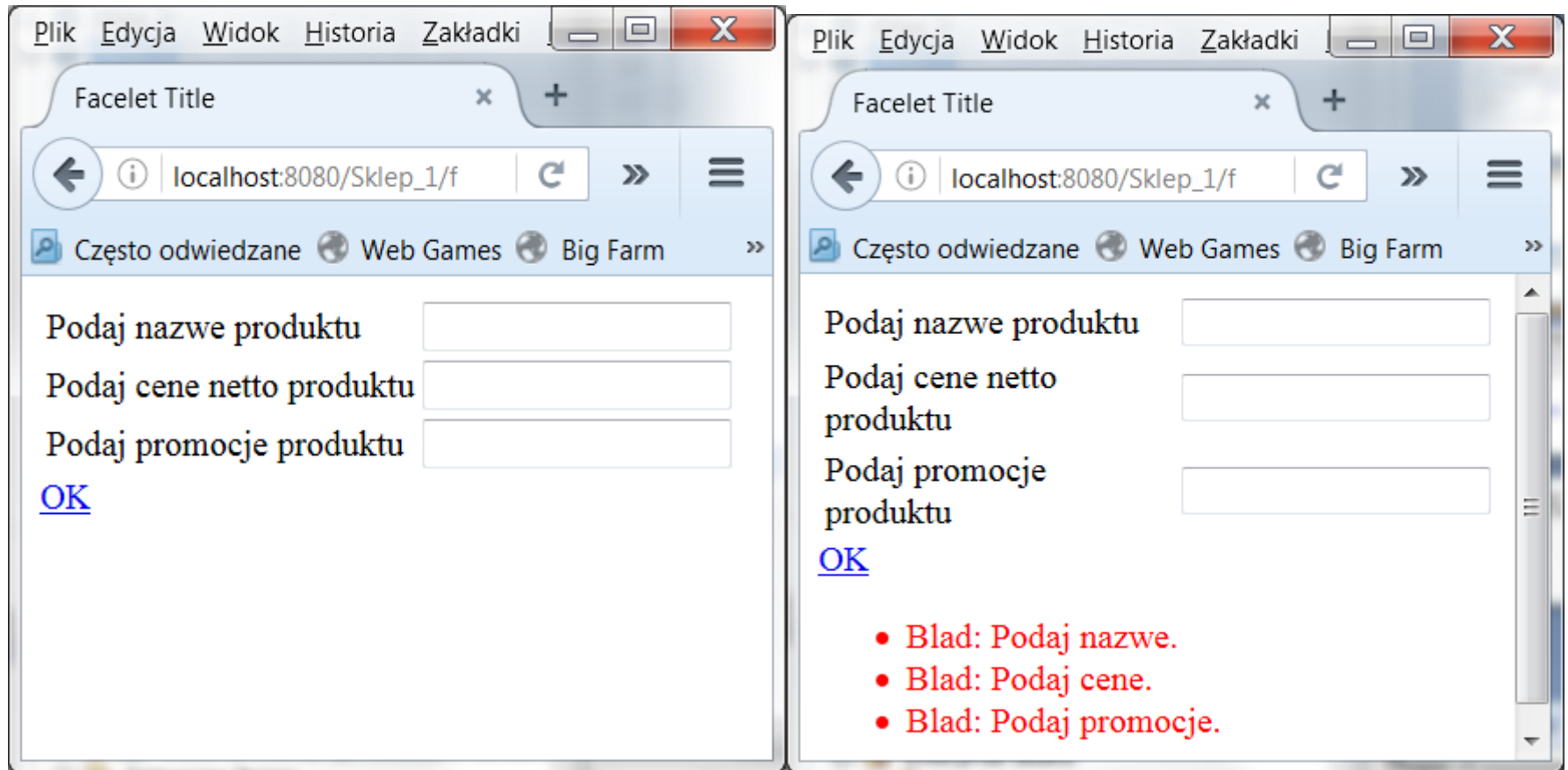
Po kliknięciu OK. na stronie **dodaj\_produk1** przejście do strony **rezultat1** i wyświetlenie danych. Po kliknięciu na **Powrót** przejście do strony **główniej index**.

Obecnie należy dodać możliwość wyznaczenia ceny brutto produktu. Sposób rozwiązania podano na str. 56. Na slajdzie 55 przedstawiono zachowanie aplikacji w przypadku braku wprowadzonych danych na stronie **dodaj\_produk1**.





## Działanie walidacji typu **required** kontrolującej brak danych

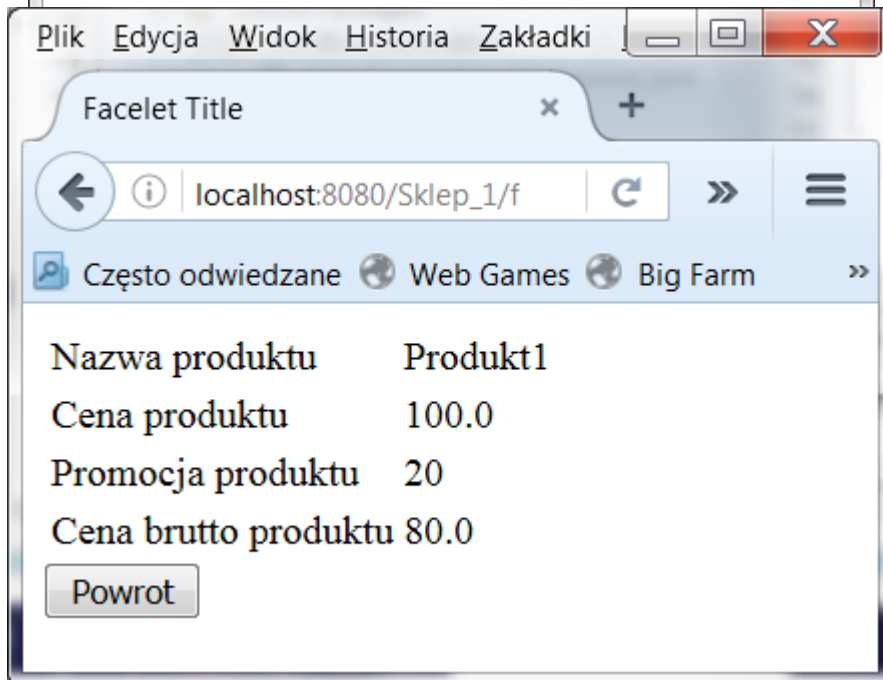
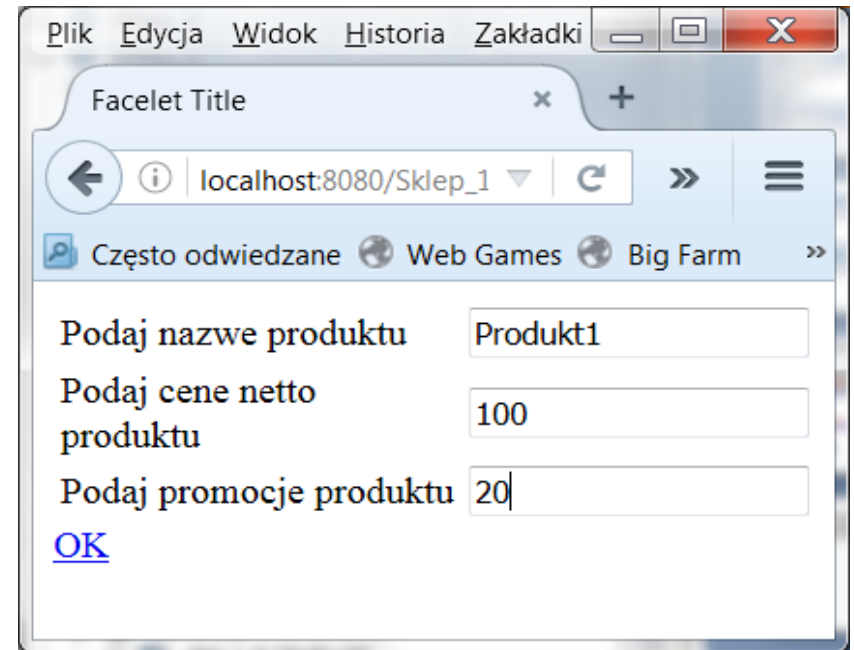
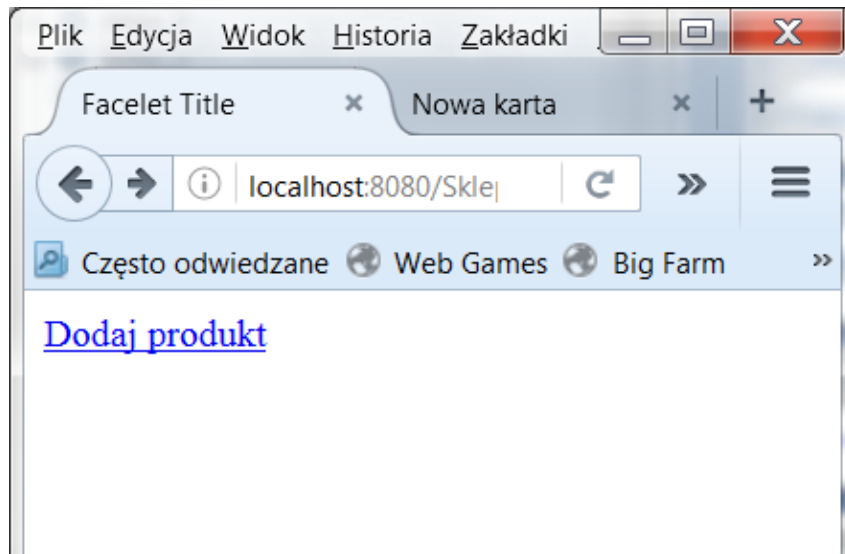


12. Dodane metod do klasy Managed\_produkci obsługujących dodawanie produktu (**dodaj\_produkci**) po pobraniu danych z formularza za pomocą atrybutów: nazwa, cena, promocja i wywołaniu metody **utworz\_produkci** ziarna EJB z obiektu fasada klasy typu Fasada\_warstwy\_biznesowej oraz wyświetlanie danych za pomocą metody **dane\_produkci** pobranych z warstwy biznesowej od obiektu typu EJB fasada za pomocą metody **dane\_produkci**

```
public String dodaj_produkci() {  
    String[] dane = {nazwa, cena, promocja};  
    fasada.utworz_produkci(dane);           // slajd 31  
    dane_produkci();  
    return "rezultat1";  
}
```

```
public void dane_produkci()  
{ String[] dane=fasada.dane_produkci();           // slajd 31  
    nazwa=dane[0];  
    cena=dane[1];  
    promocja=dane[2];  
    cena_brutto=dane[3];  
}  
}
```

## Wynik po uruchomieniu aplikacji



## 13. Zadanie do wykonania - dodatkowe

Zadanie należy zmodyfikować dodając w definicji klasy dodatkowy atrybut np. Producent lub Kategoria\_produkту.

Należy uzupełnić metody klasy Fasada\_warstwy\_biznesowej oraz Managed\_produkт oraz zmodyfikować pliki dodaj\_produkт1.xhtml i rezultat1.xhtml w celu wprowadzania nowej danej oraz wyświetlania jej danych.

Wykonanie tego zadania z p.13 – można otrzymać ocenę 5.5

## Zaliczenie lab1

1. Wykonanie p.1-12 instrukcji do lab1 jest niezbędne do zaliczenia lab1.
2. Ocena z laboratoriów będzie średnią arytmetyczną z lab. 2-6. Jeśli zostanie wykonany p.13 z lab1, wtedy ocena końcowa będzie wystawiona jako średnia arytmetyczna z lab1-6.
3. Ocena z lab. 6 będzie wystawiona na podstawie sprawdzianu z umiejętności wykonania dodatkowych operacji walidacji lub konwersji w programach wykonanych podczas lab4 na polu dodanym do obiektu typu Produkt1 i wartościach wprowadzanych do tego pola za pośrednictwem strony internetowej.

**Dodatek-Biblioteki znaczników obsługiwanych przez  
Facelets wg  
<https://docs.oracle.com/javaee/7/javaxserver-faces-2-2/vdldocs-facelets/toc.htm>**