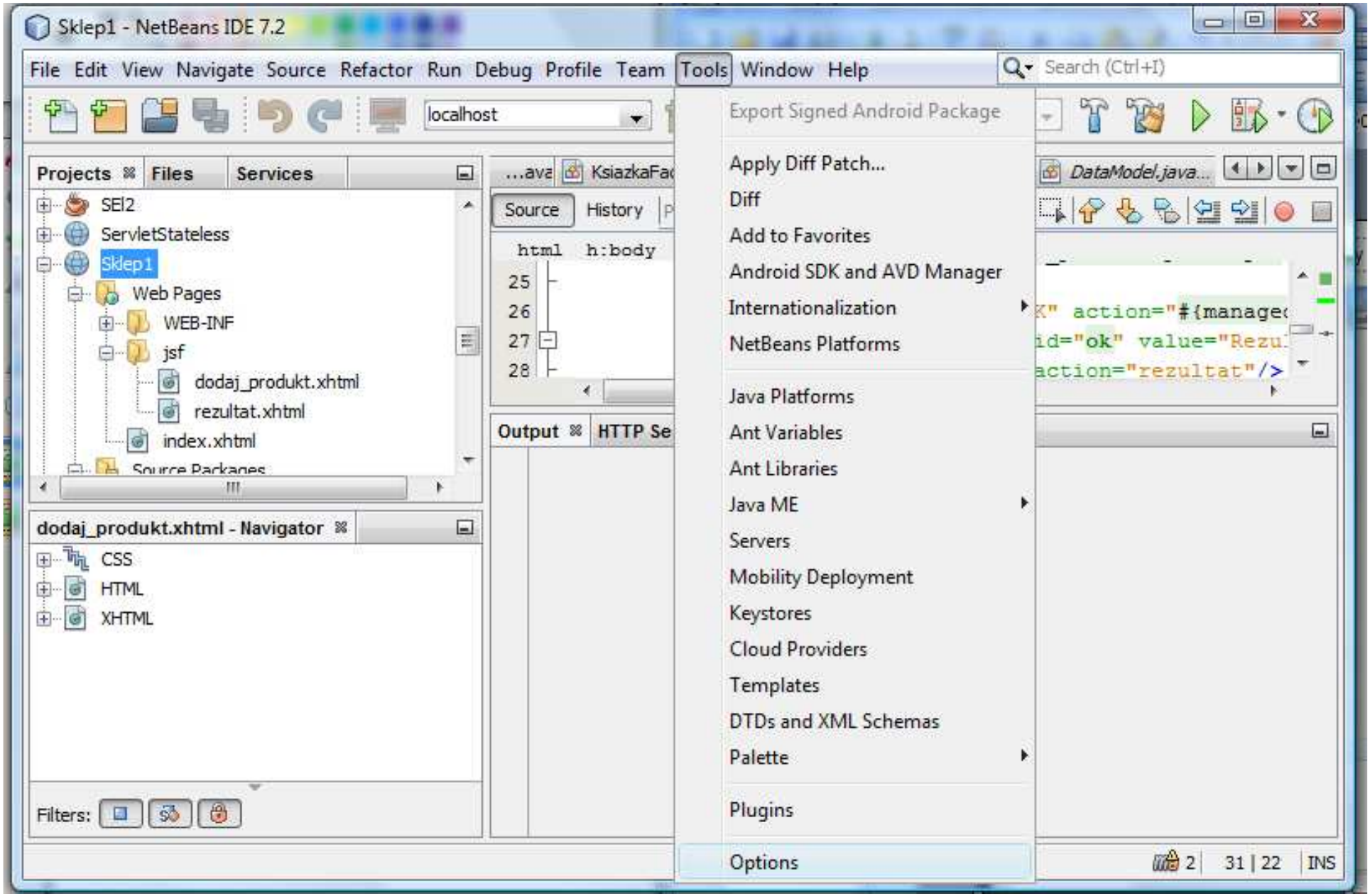
Budowa prostej aplikacji wielowarstwowej

Laboratorium 1

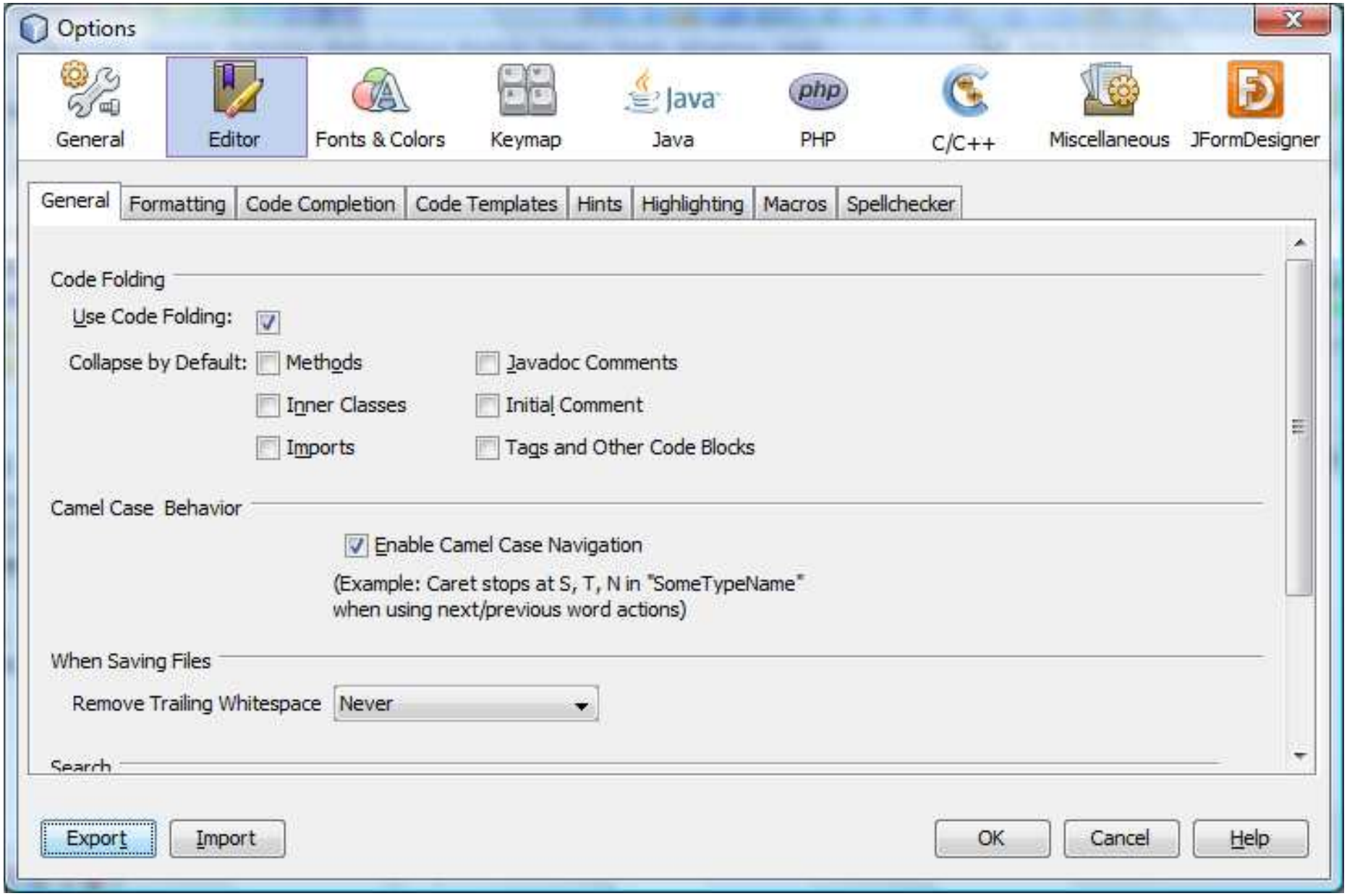
Programowanie komponentowe

Zofia Kruczkiewicz

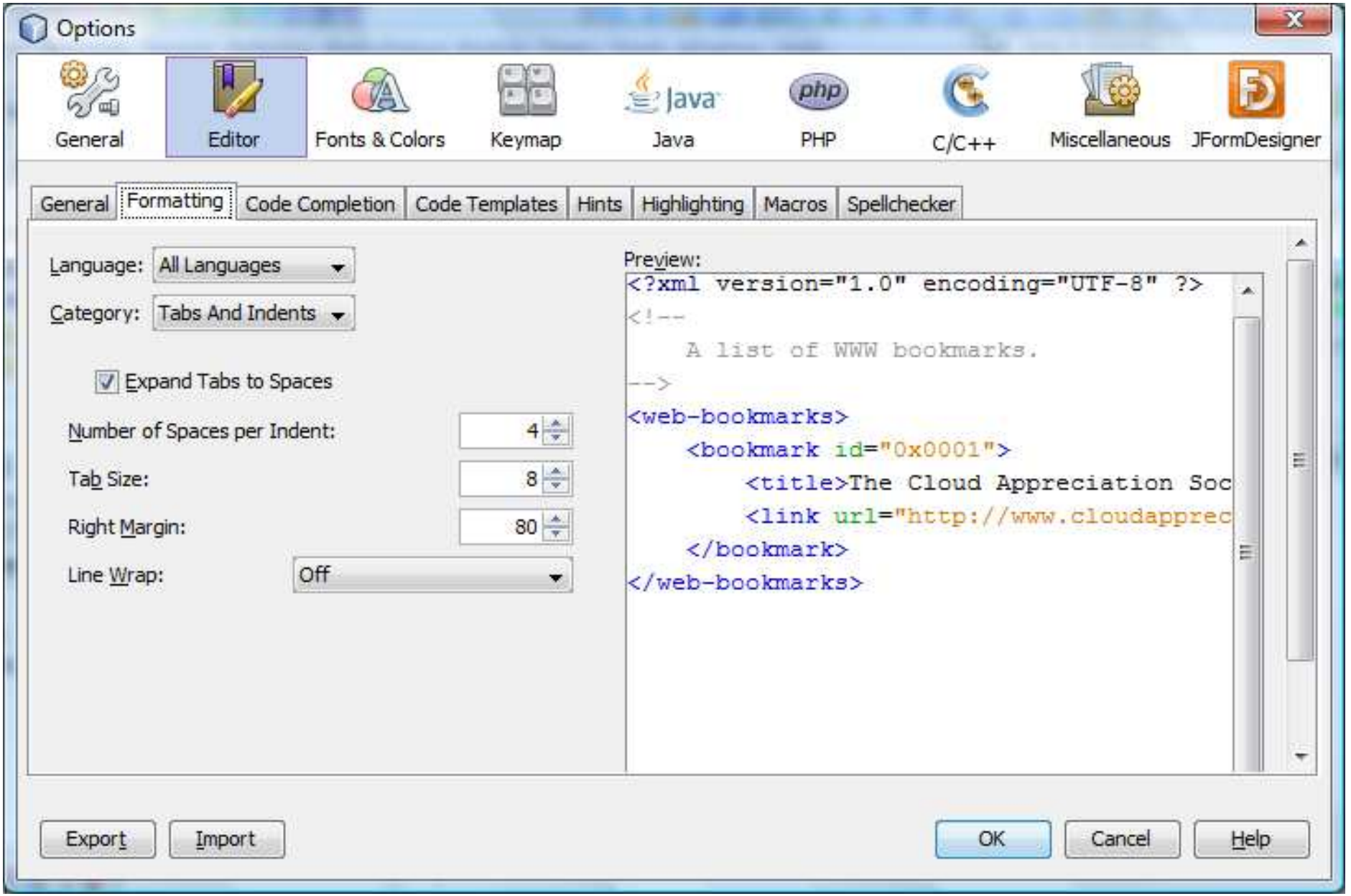
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



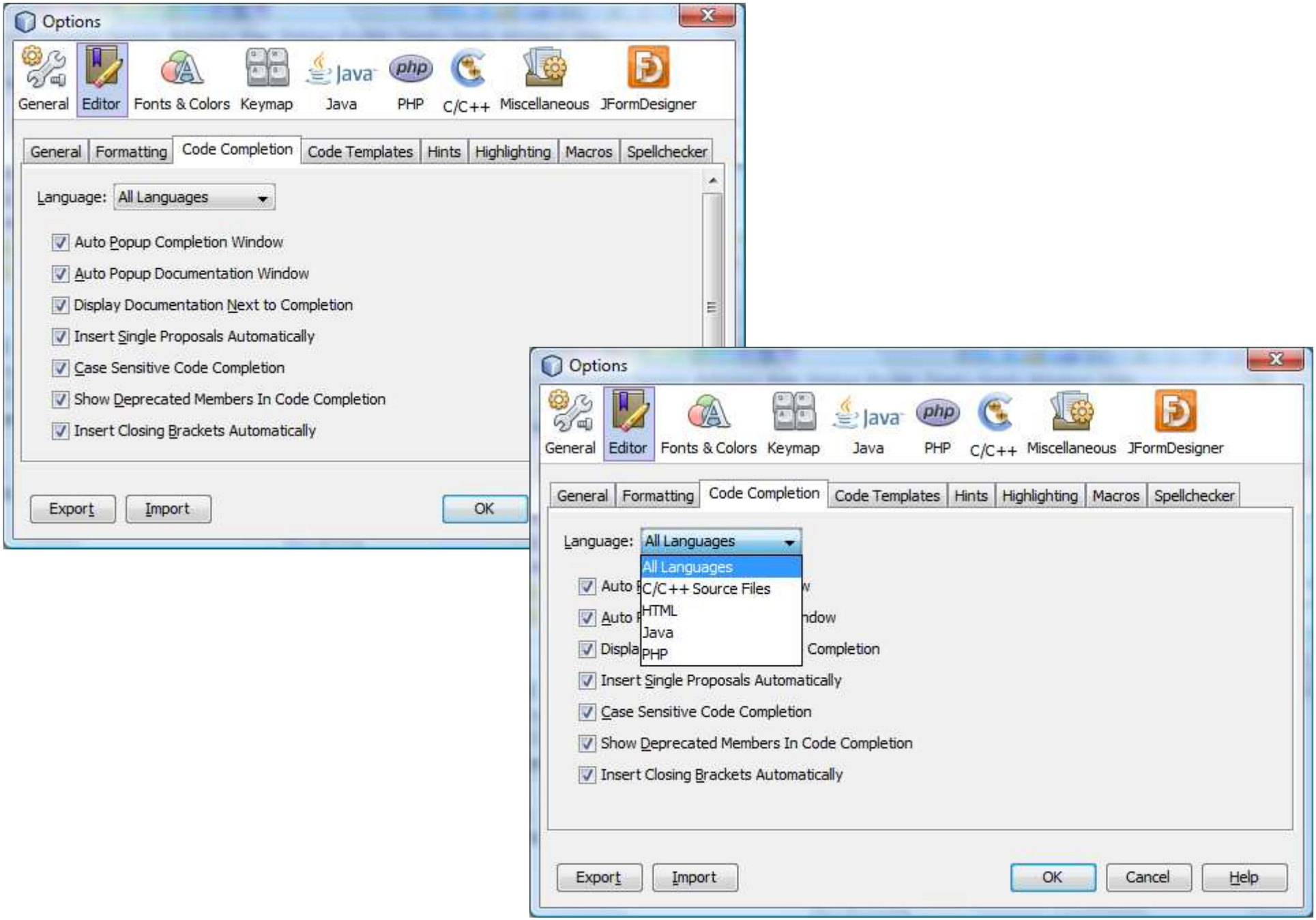
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



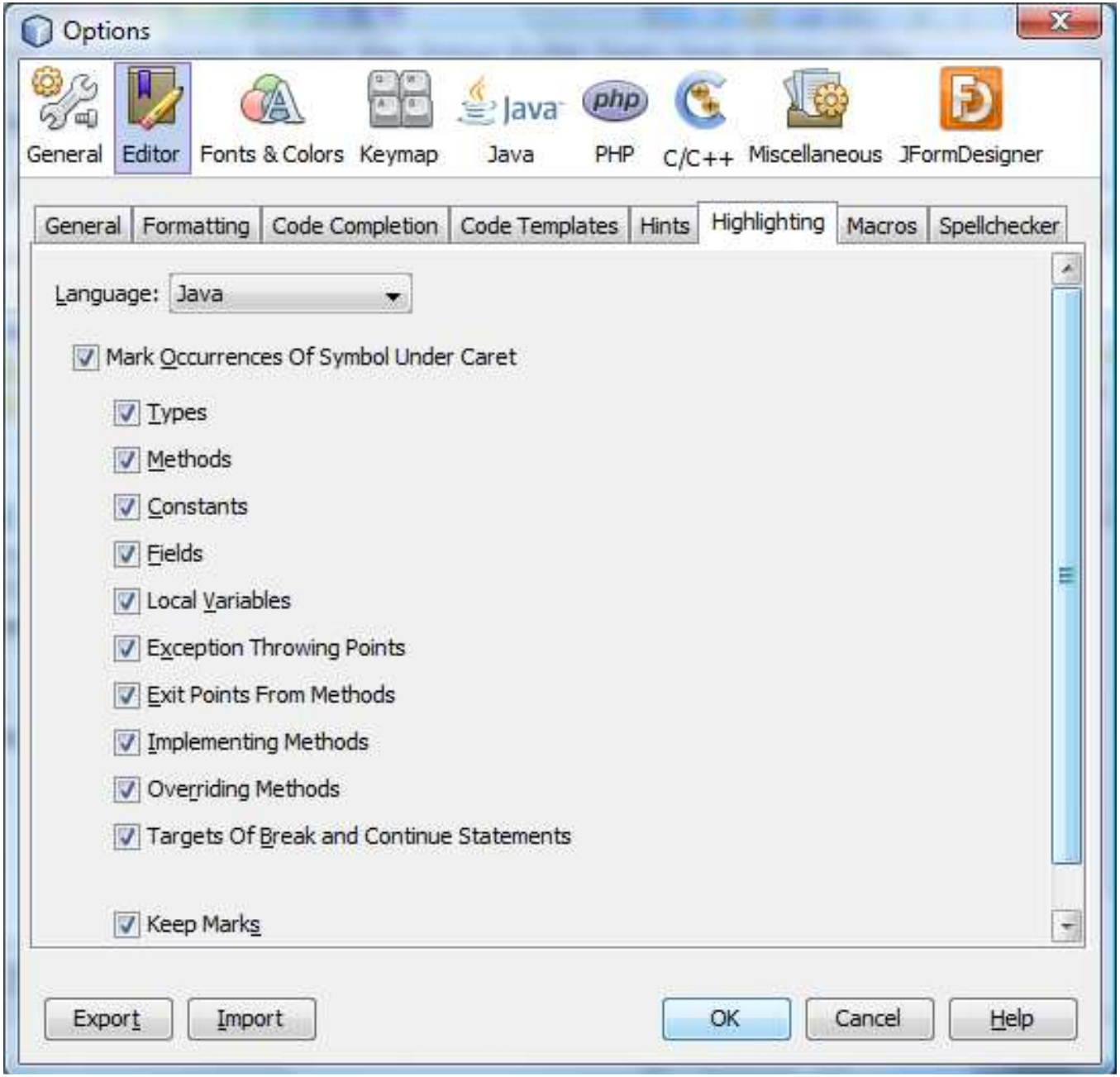
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



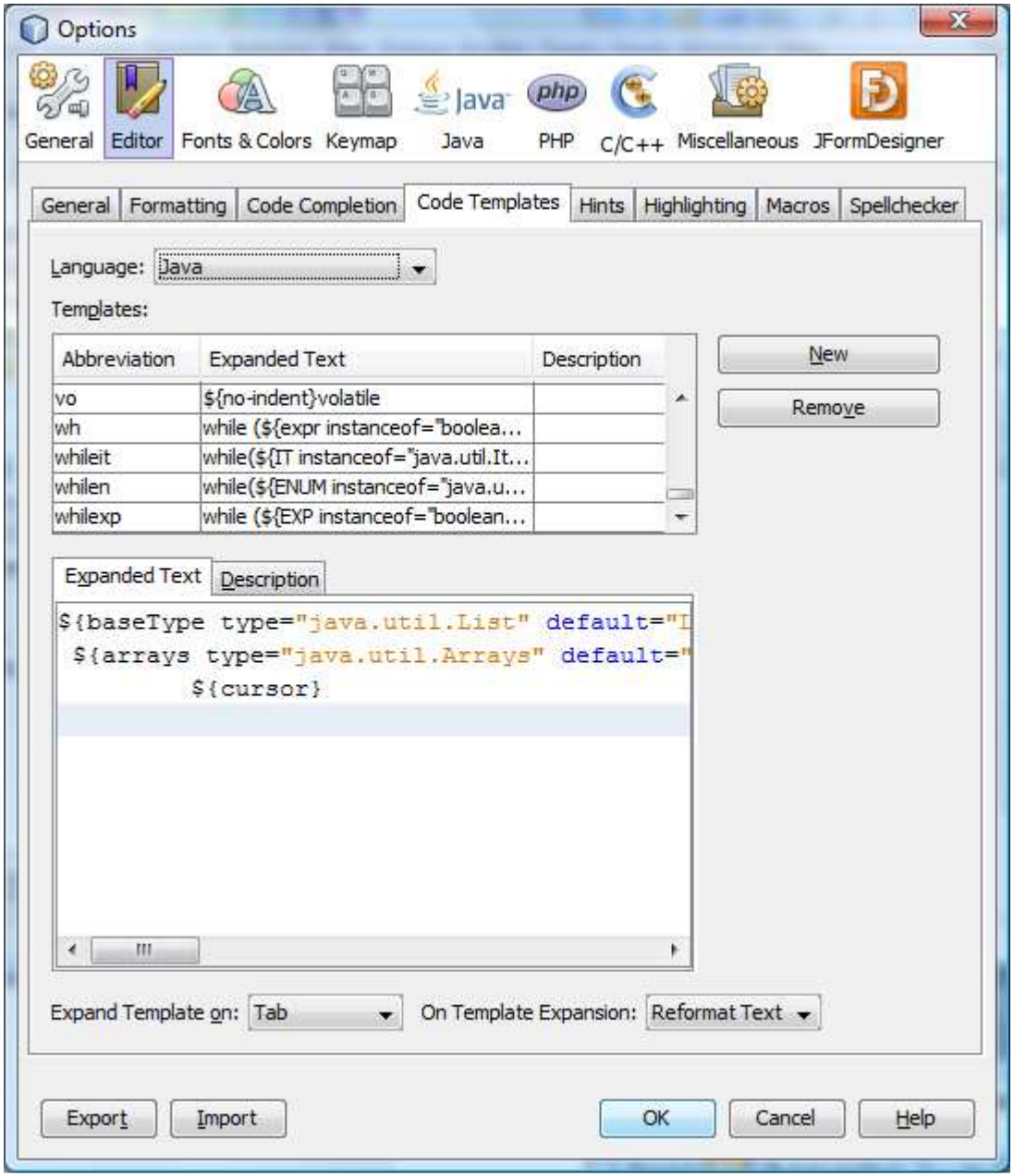
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



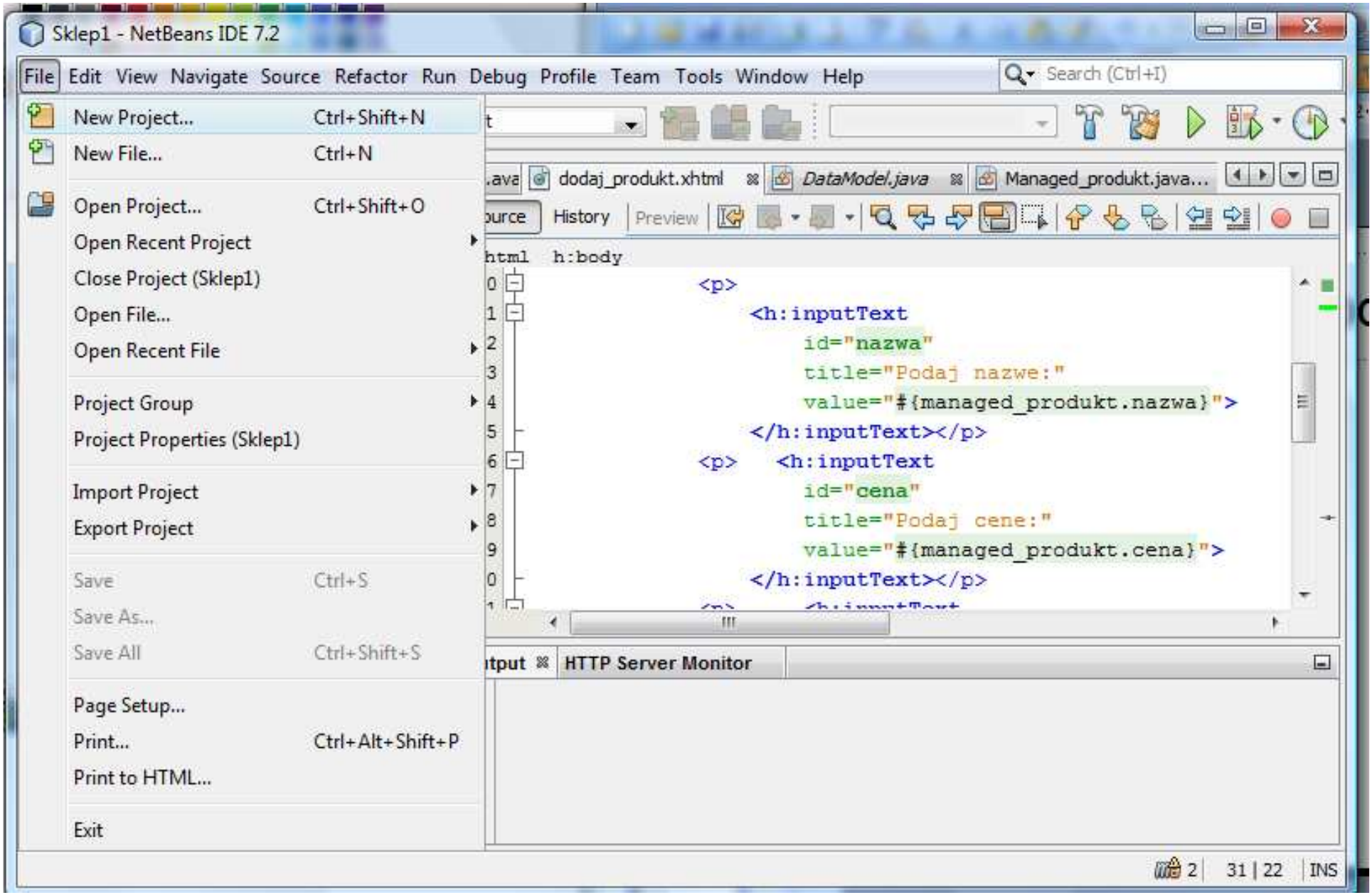
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



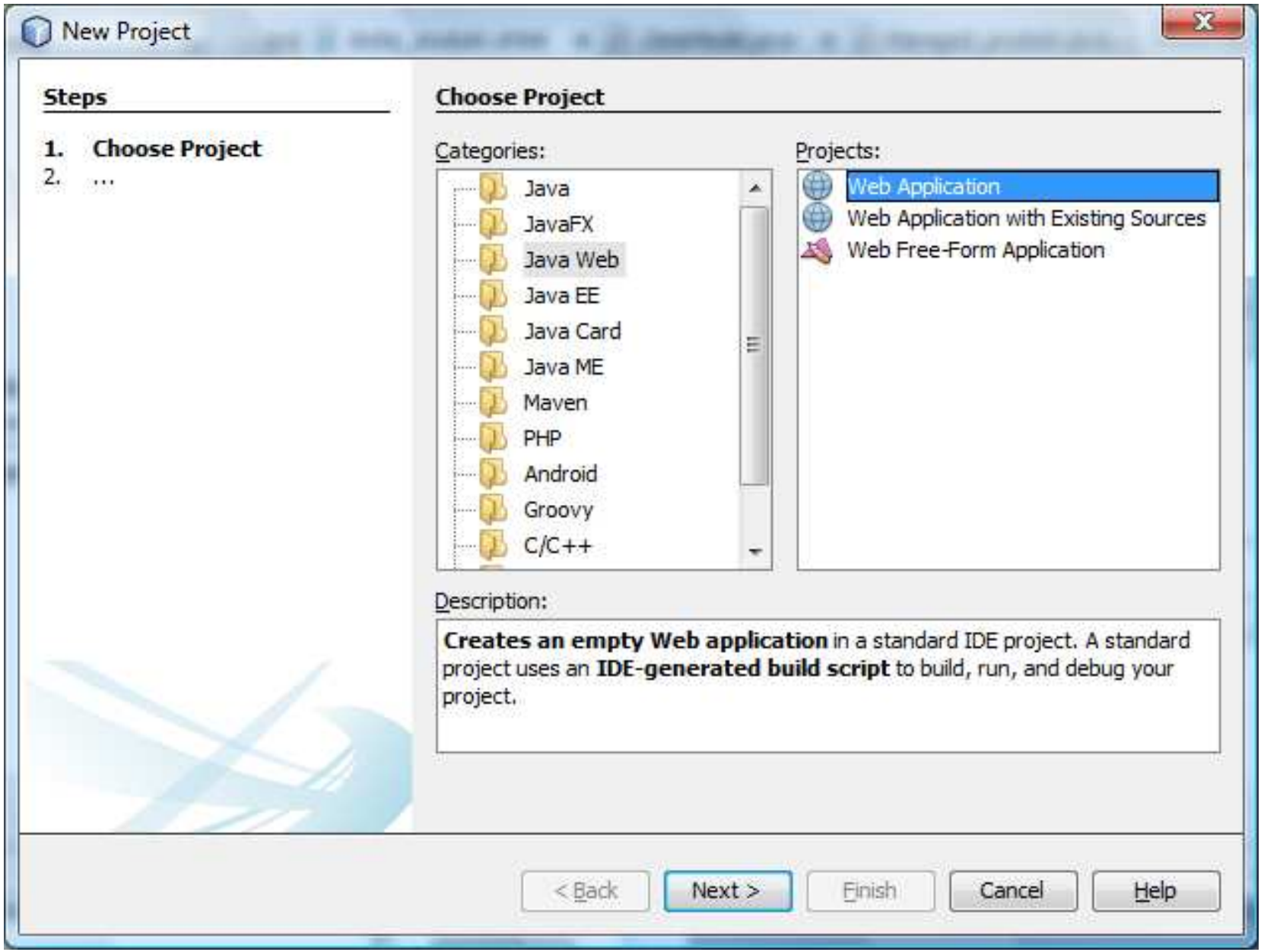
Konfigurowanie edytora programu za pomocą **Tools/Options/Editor**



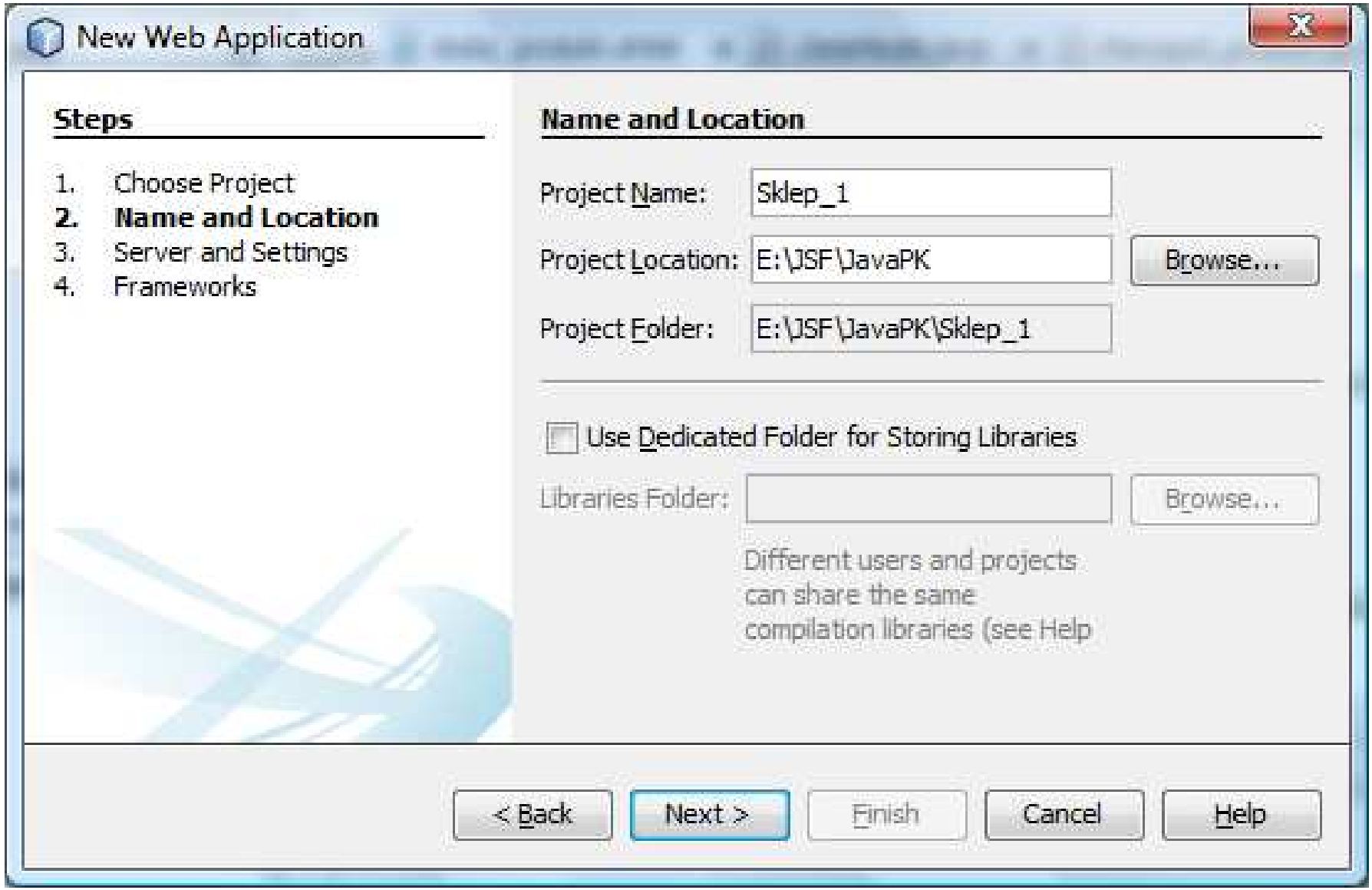
Zakładanie projektu typu Web Application (**File/New Project**)



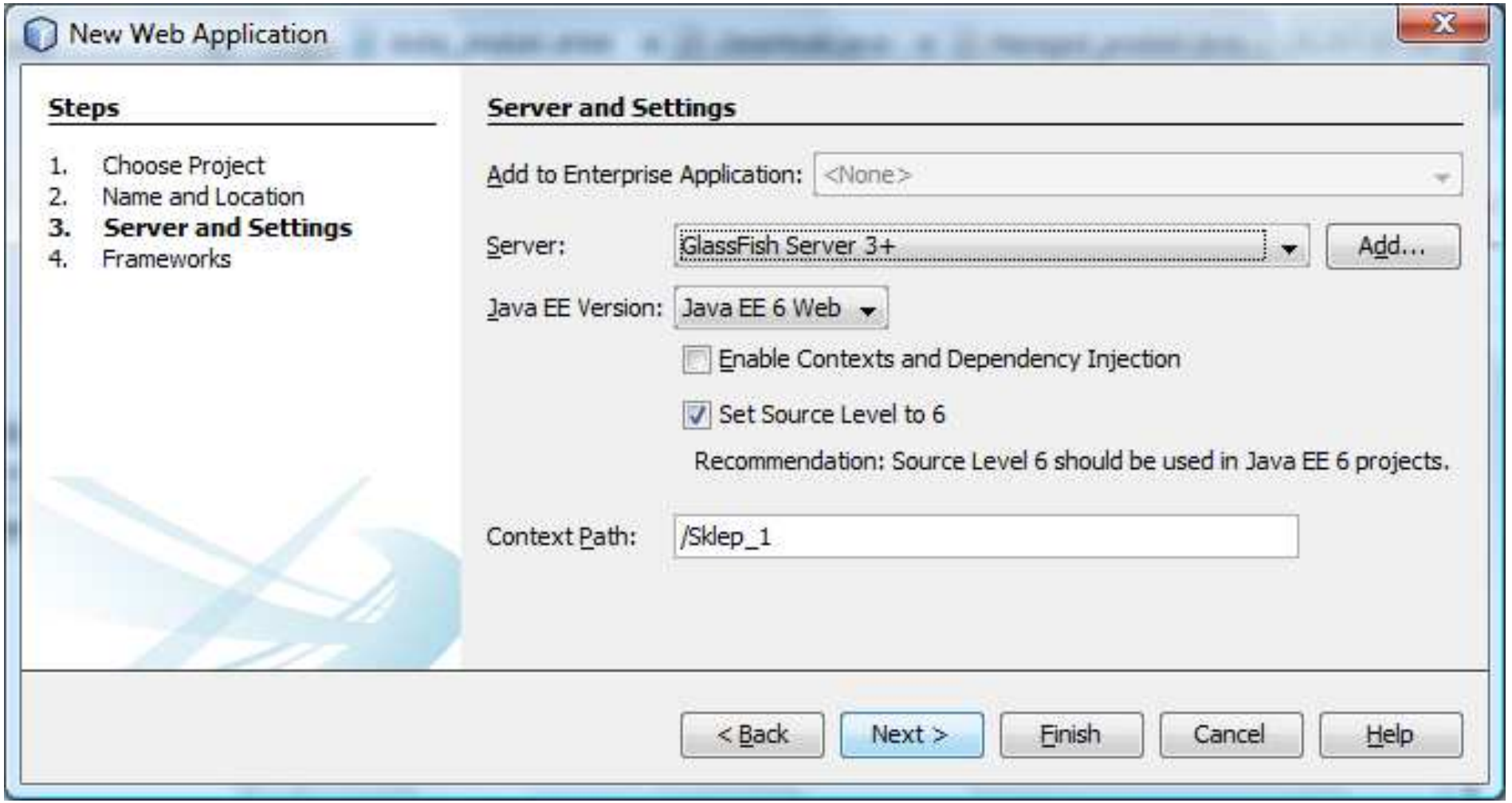
Zakładanie projektu typu Web Application (**Java Web/Web Application i Next**)



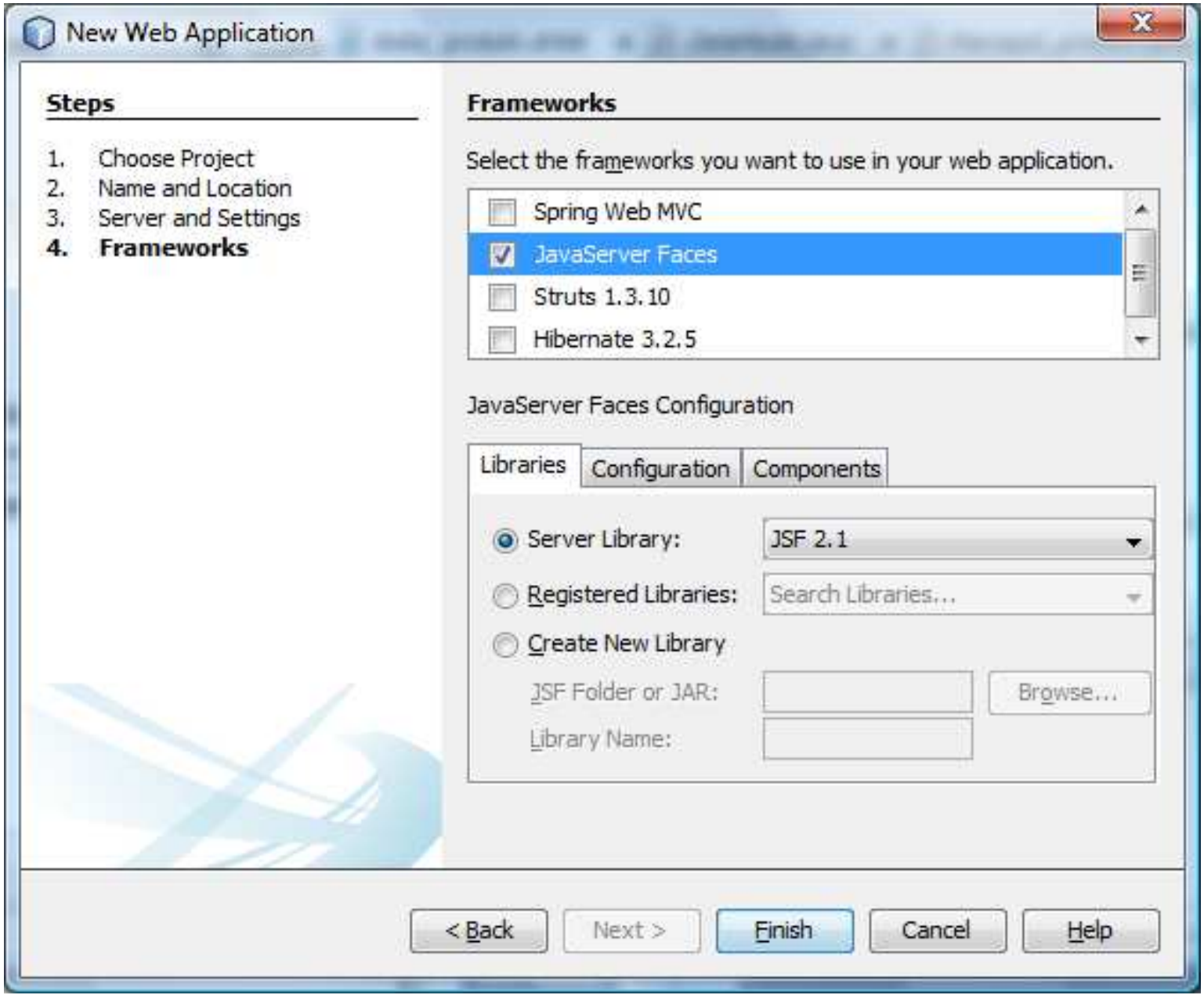
Zakładanie projektu typu Web Application: **Project Name: Sklep_1**, należy wybrać Project Location



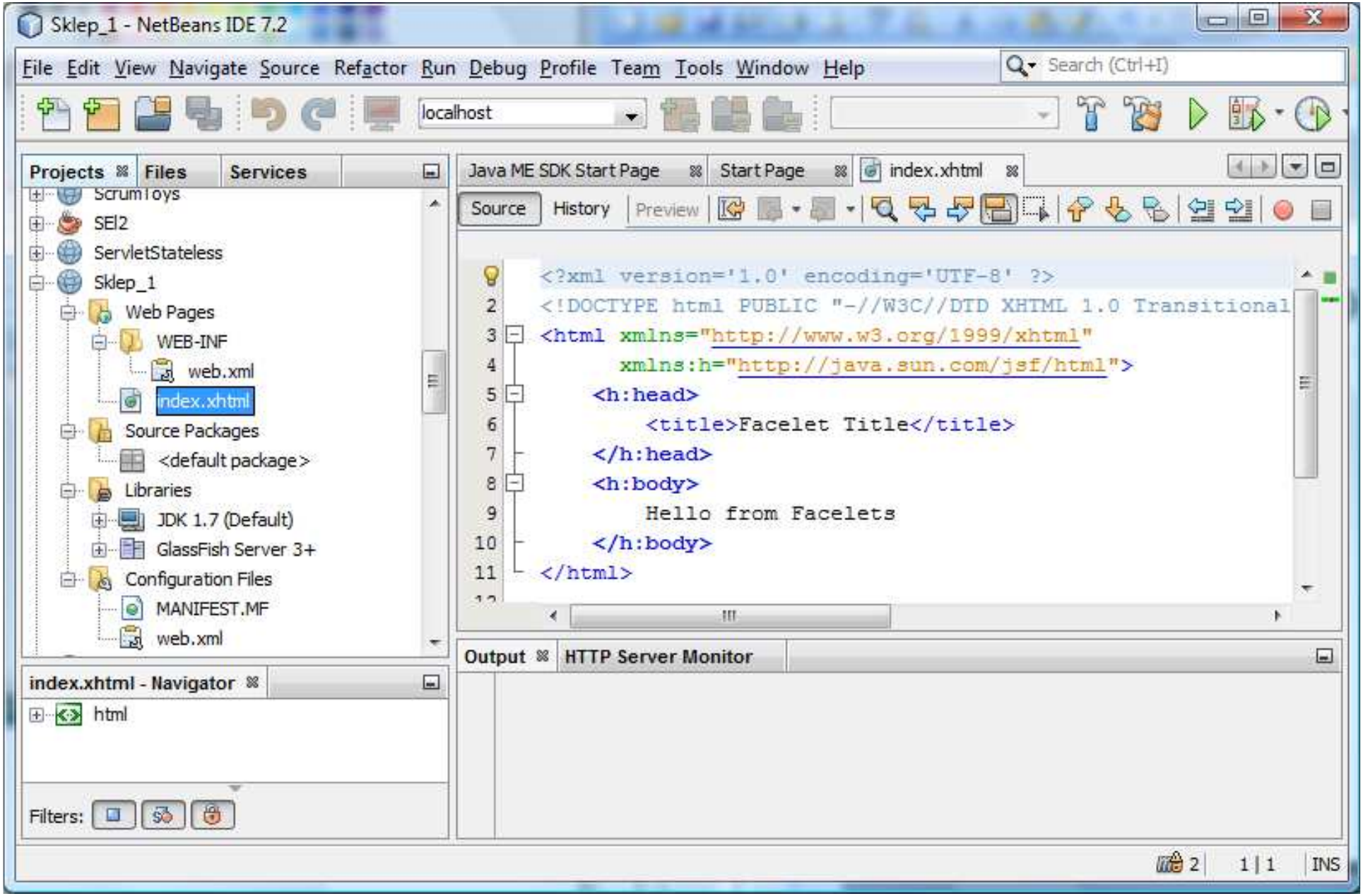
Zakładanie projektu typu Web Application: **Server: GlassFish 3+, Java EE Version: Java EE 6 Web i Next**



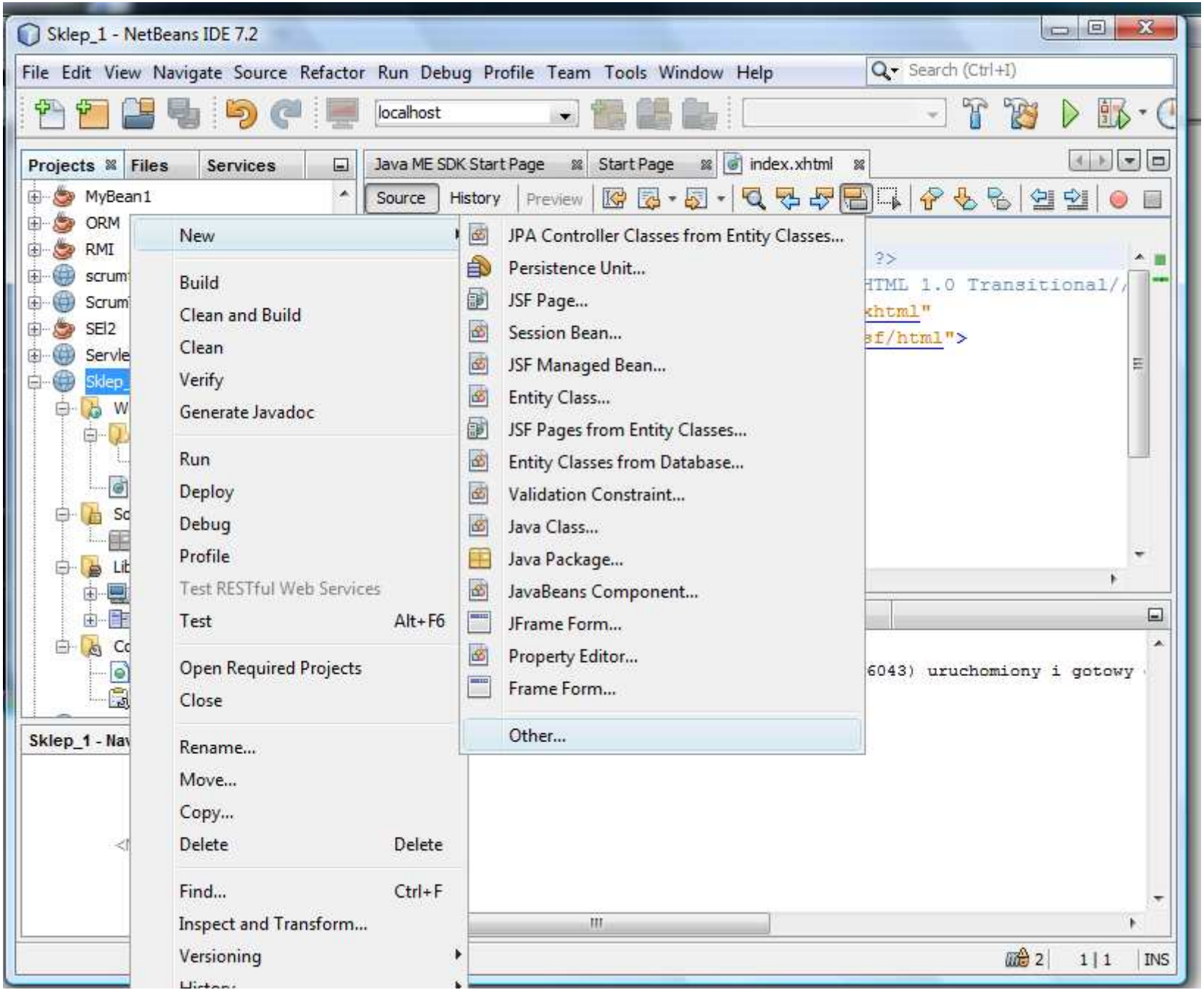
Zakładanie projektu typu Web Application- zaznaczyć **checkbox JavaServer Faces i Finish**



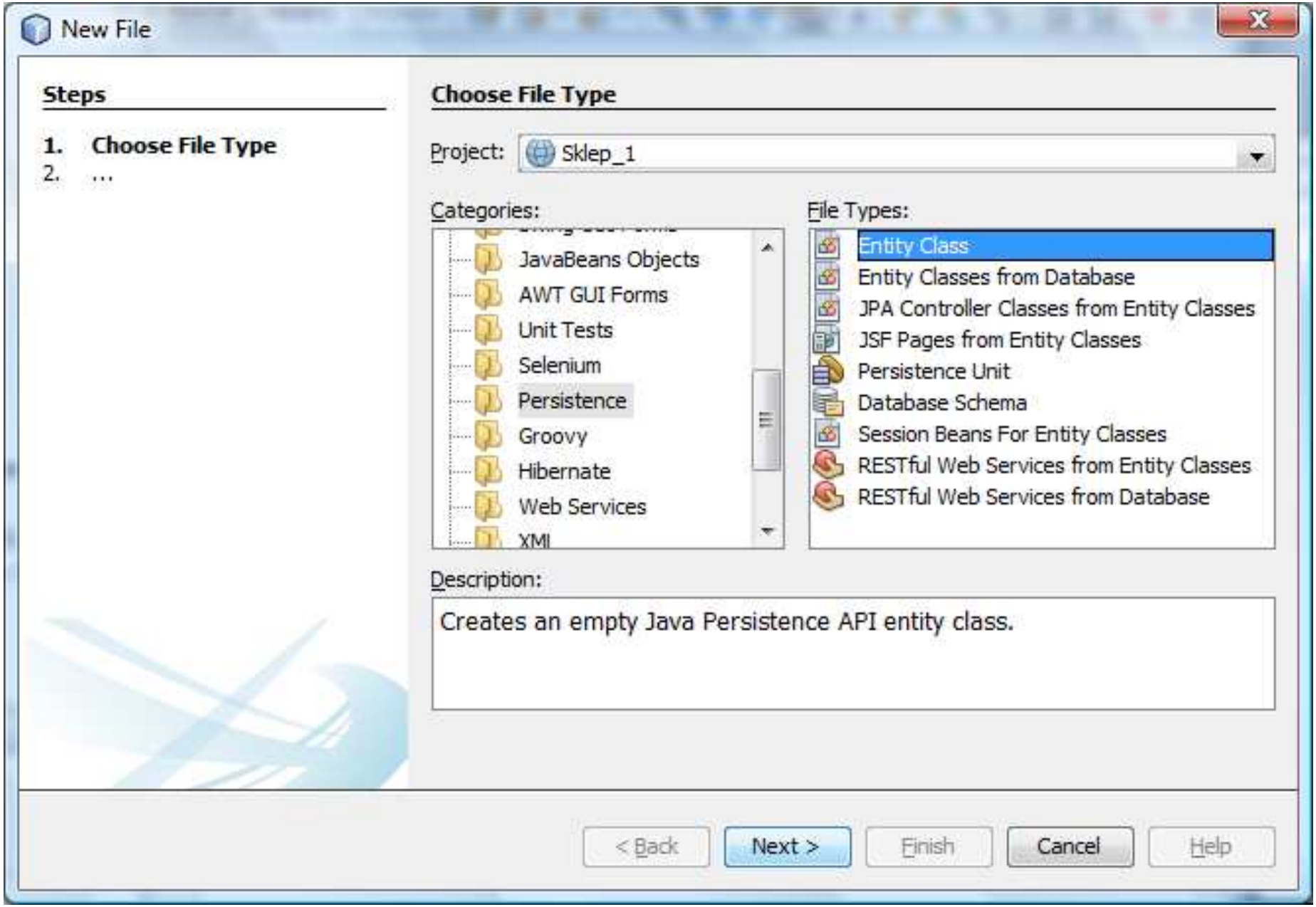
Zakładanie projektu typu Web Application – projekt ze stroną startową **index.xhtml**



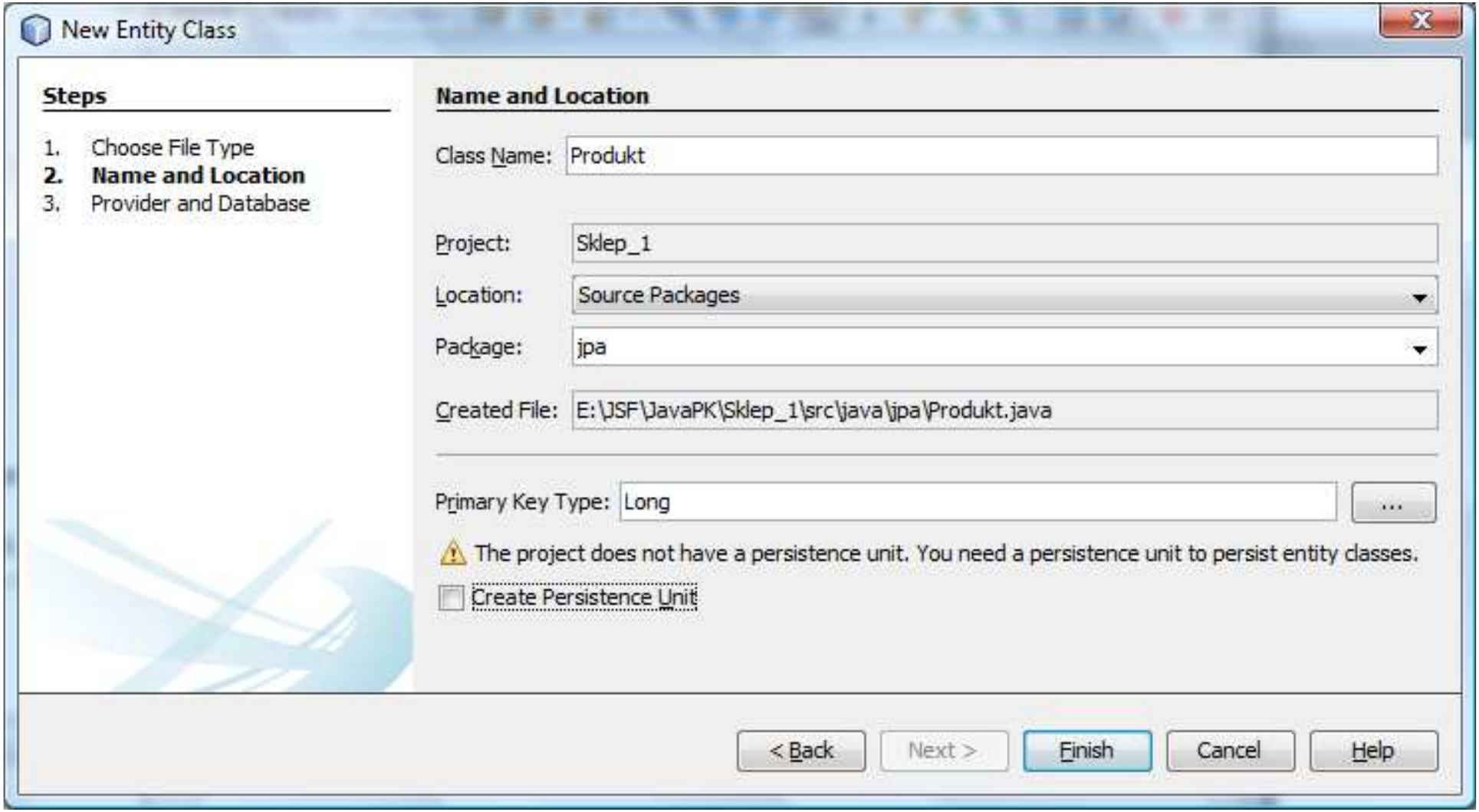
Dodanie klasy **typu Entity** przechowującej dane produktu i wyznaczającej cenę brutto (**New/Other**)



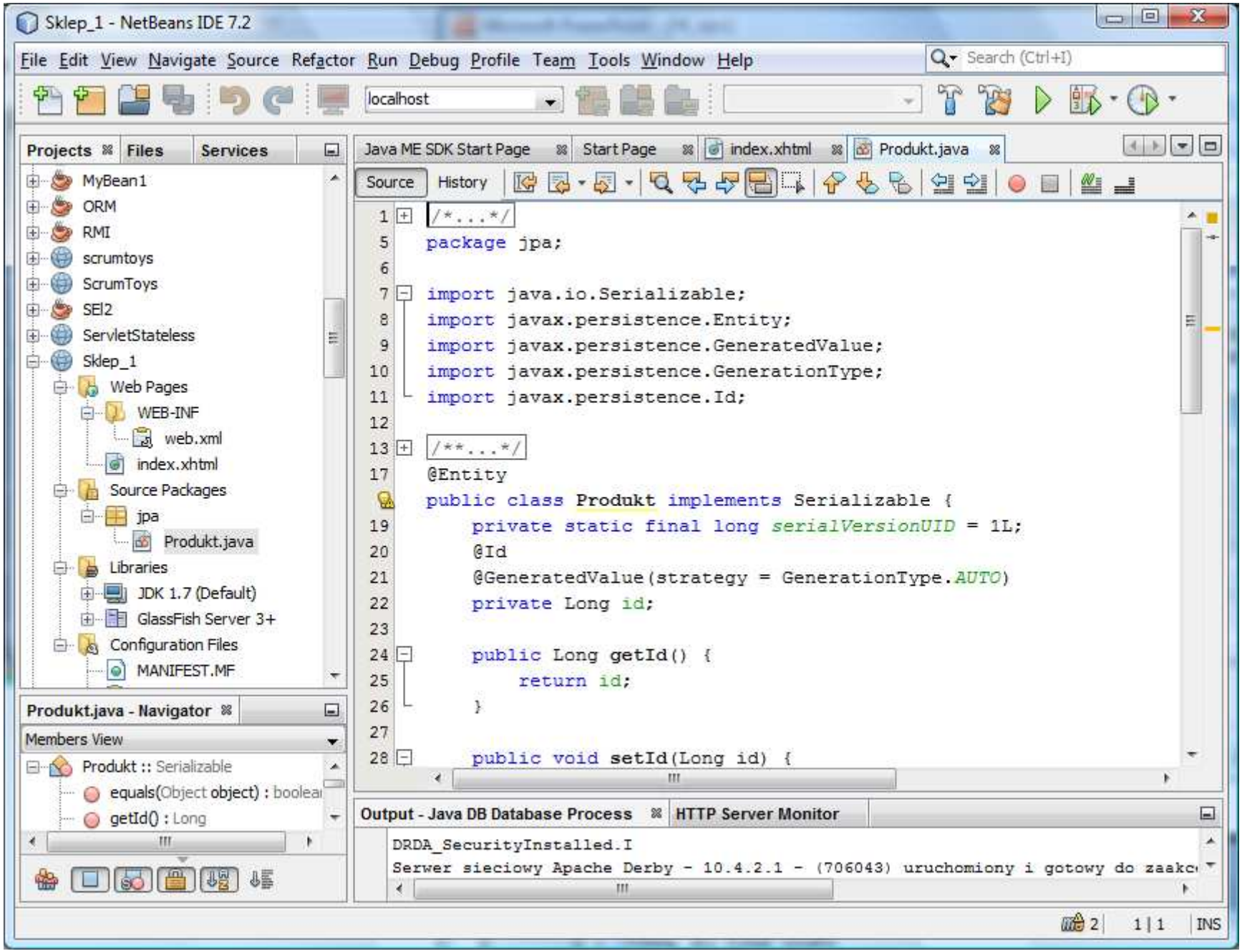
Dodanie klasy **typu Entity** przechowującej dane produktu i wyznaczającej cenę brutto – **Persistence/ Entity Class i Next**



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



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



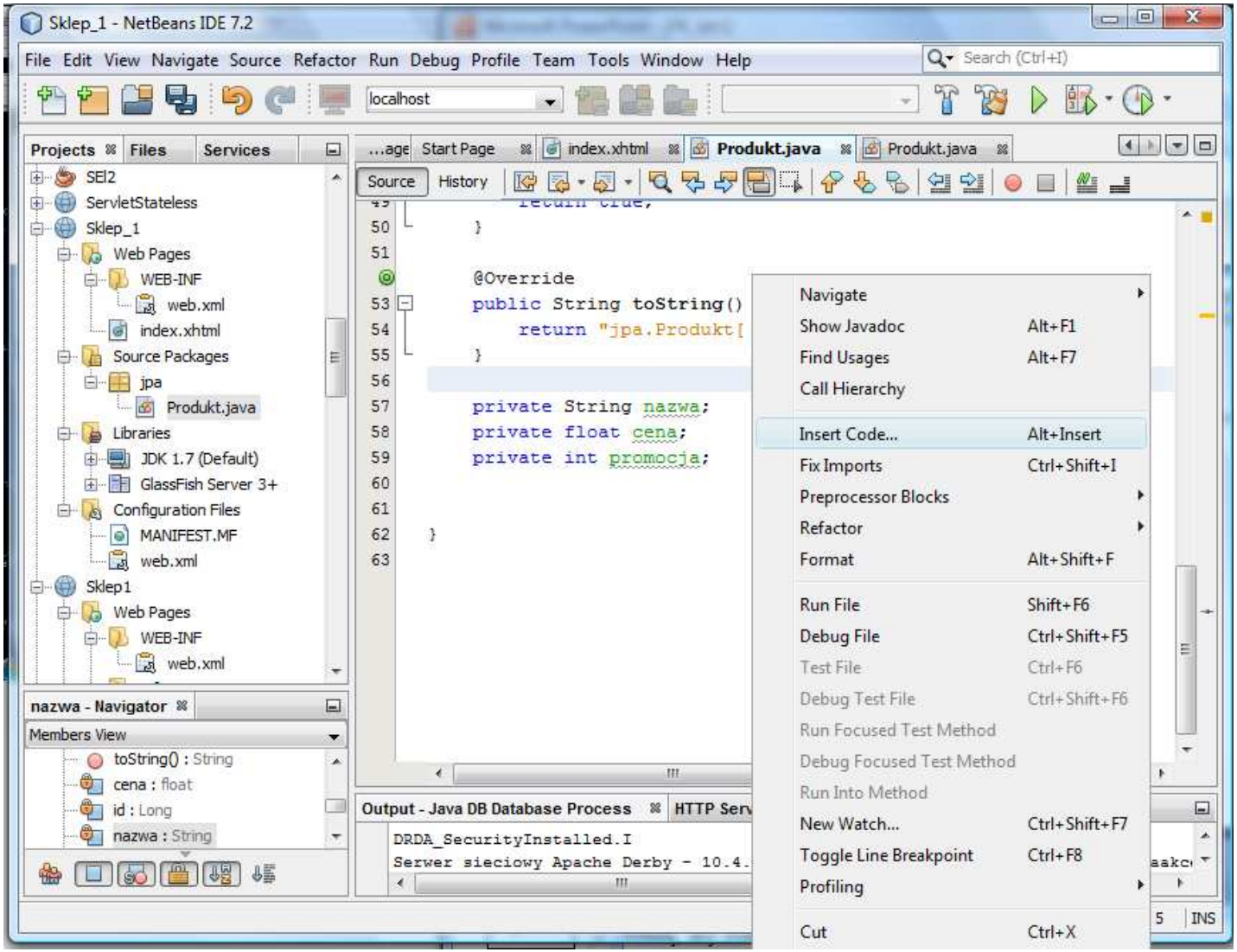
Wygenerowany kod klasy **Produkt**

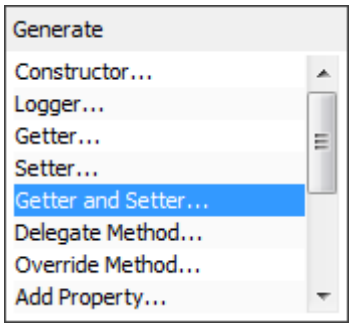
```
package jpa;

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…**



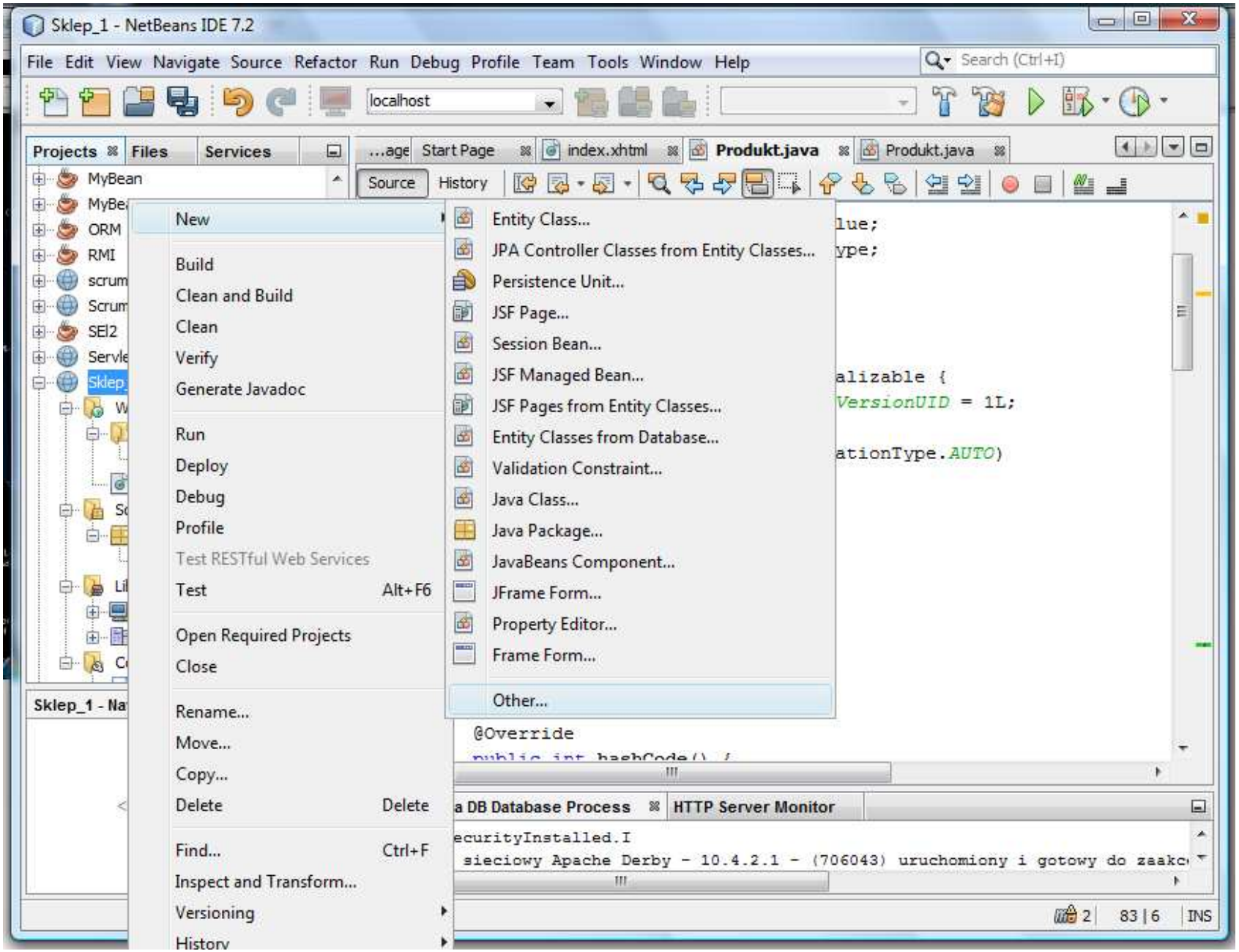


Zaznaczyć atrybuty do wygenerowania metod dostępu do atrybutów

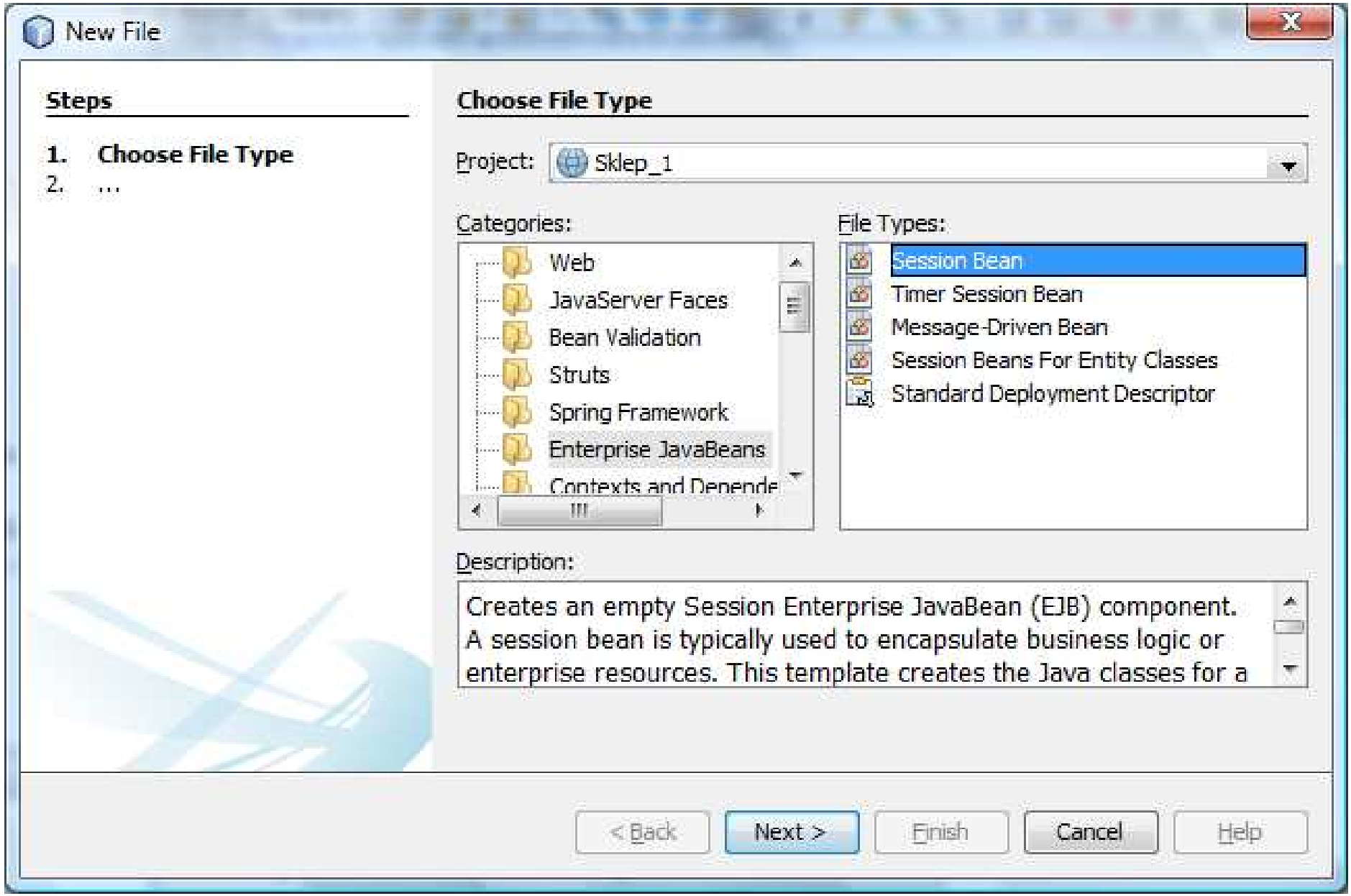
Oprócz wygenerowanych metod dodano metodę **cena_brutto()** wyznaczającą cenę brutto na podstawie ceny netto i promocji

```
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;
    }
```

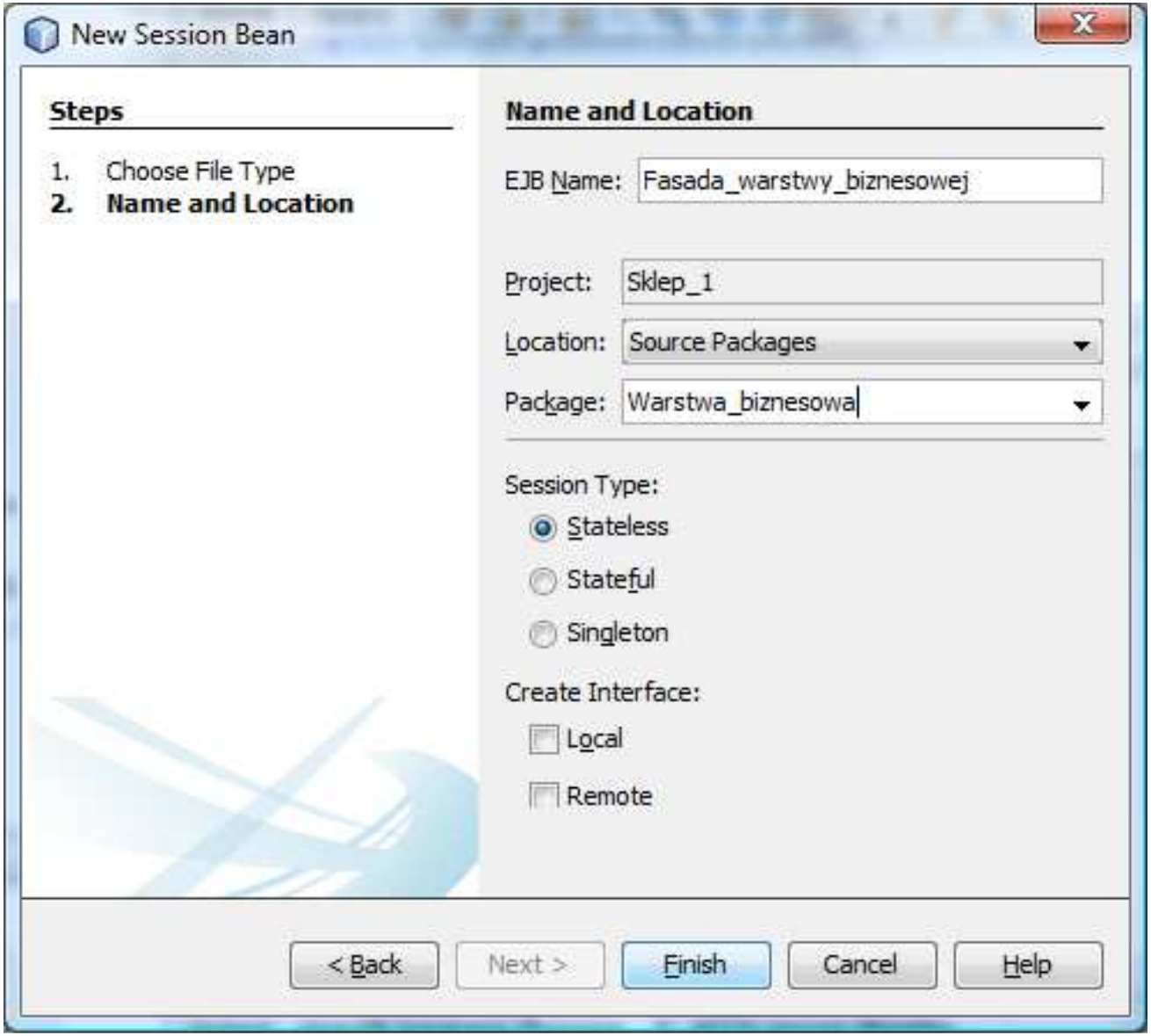
Należy dodać ziarno EJB do przetwarzania obiektu typu Entity (Produkt) – **New/Other**



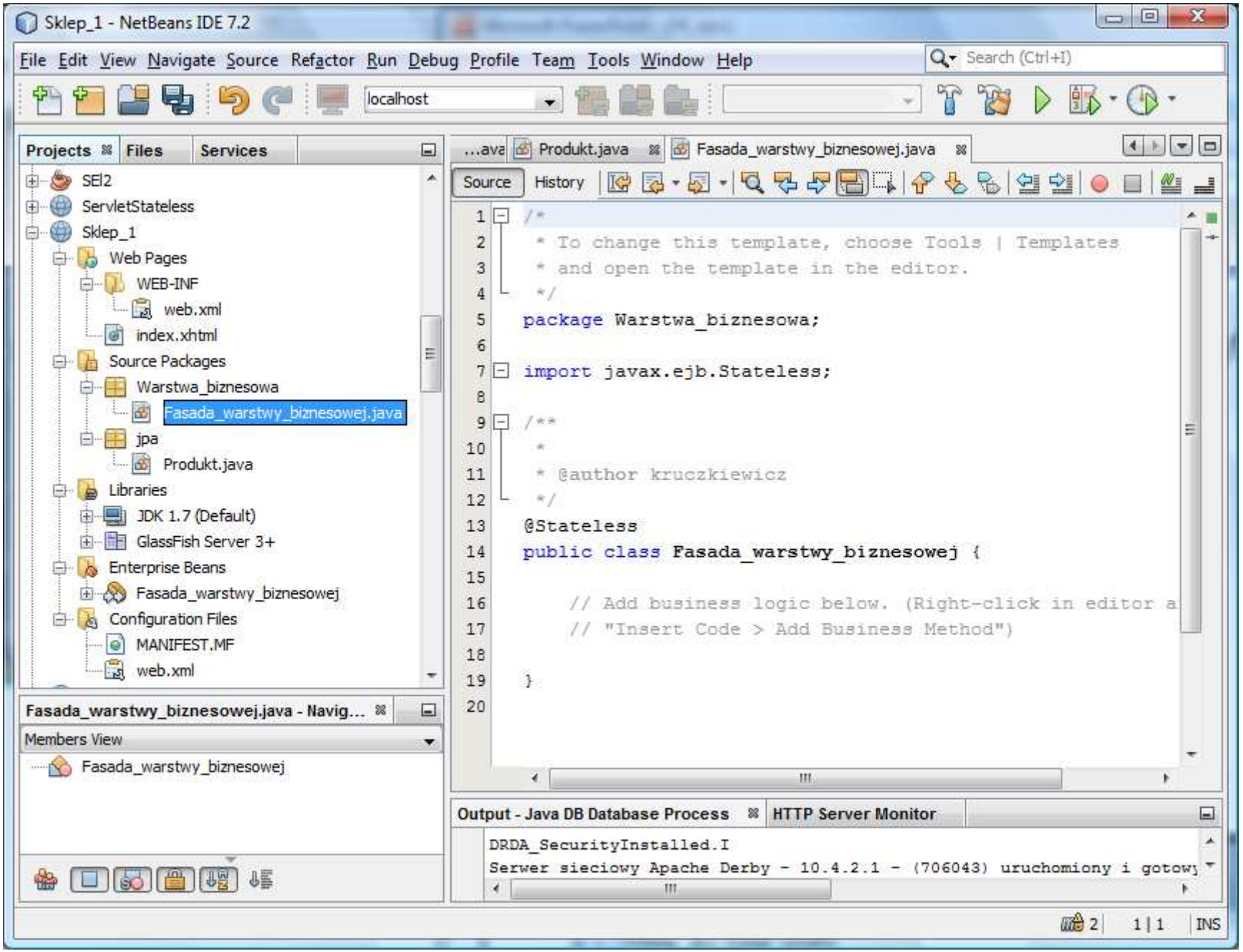
Należy dodać ziarno EJB do przetwarzania obiektu typu Entity – **Enterprise JavaBean/ Session Bean i Next**



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



Kod wygenerowany ziarna EJB do przetwarzania obiektu typu Entity

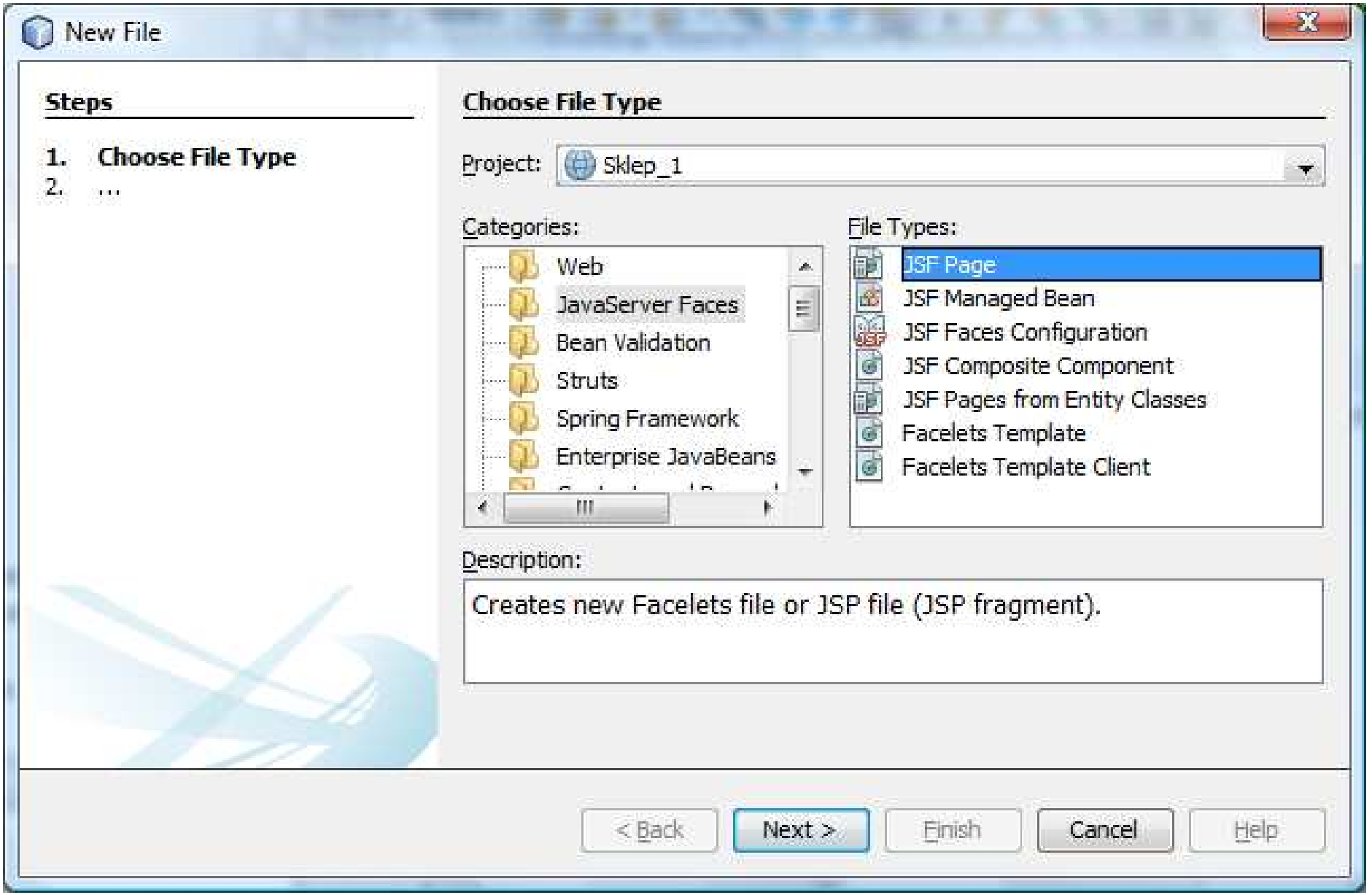


Uzupełniono kod klasy **Fasada_warstwy_biznesowe**j: dodano atrybut produkt typu Produkt, metody typu set i get (za pomocą opcji **Insert Code** - (slajdy 20-21)) oraz metody (następny slajd): **utworz_produkt**, która tworzy produkt nadając mu dane pobrane z tablicy **dane** przekazanej do metody oraz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

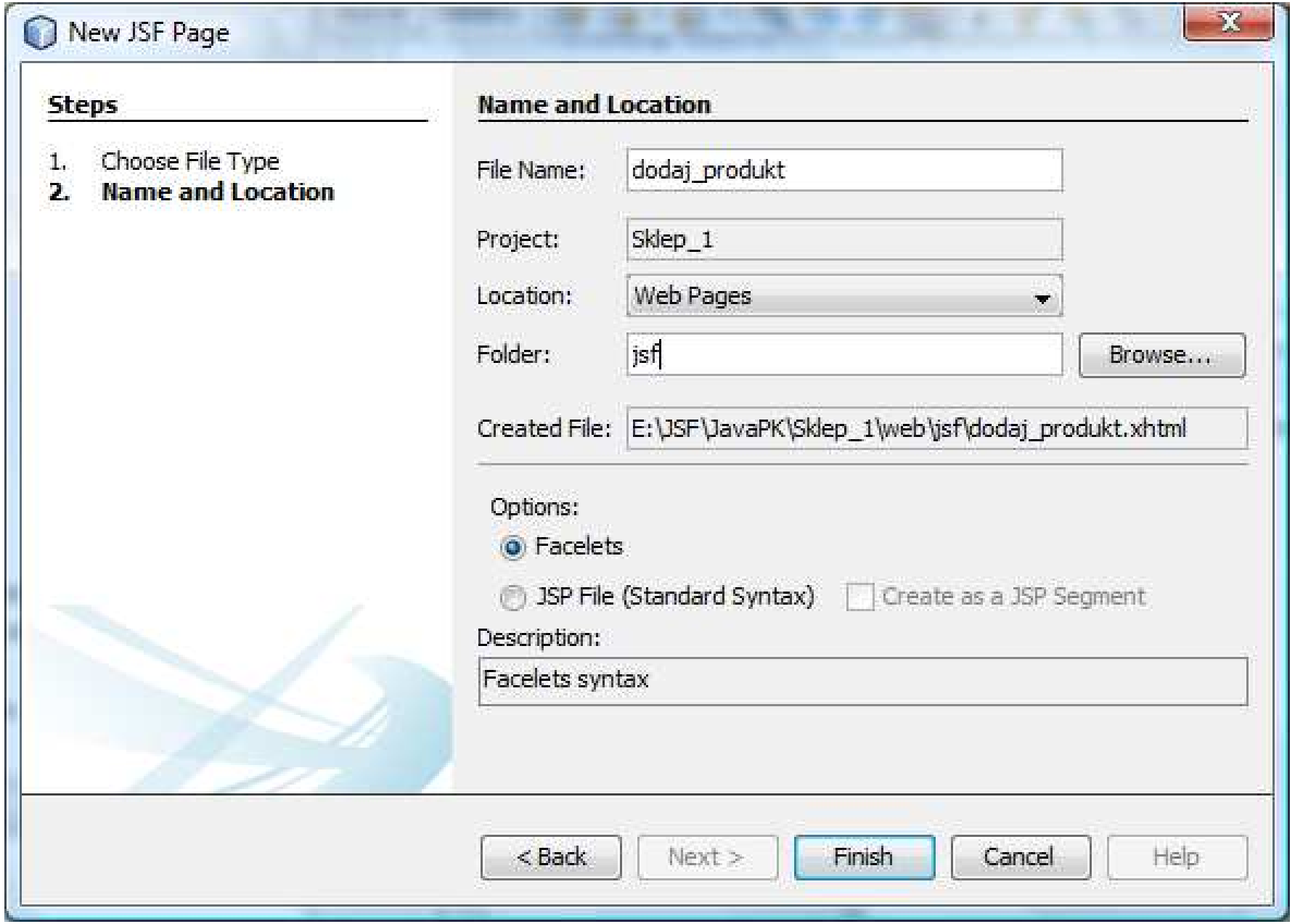
```
package Warstwa_biznesowa;

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;
    }
}
```

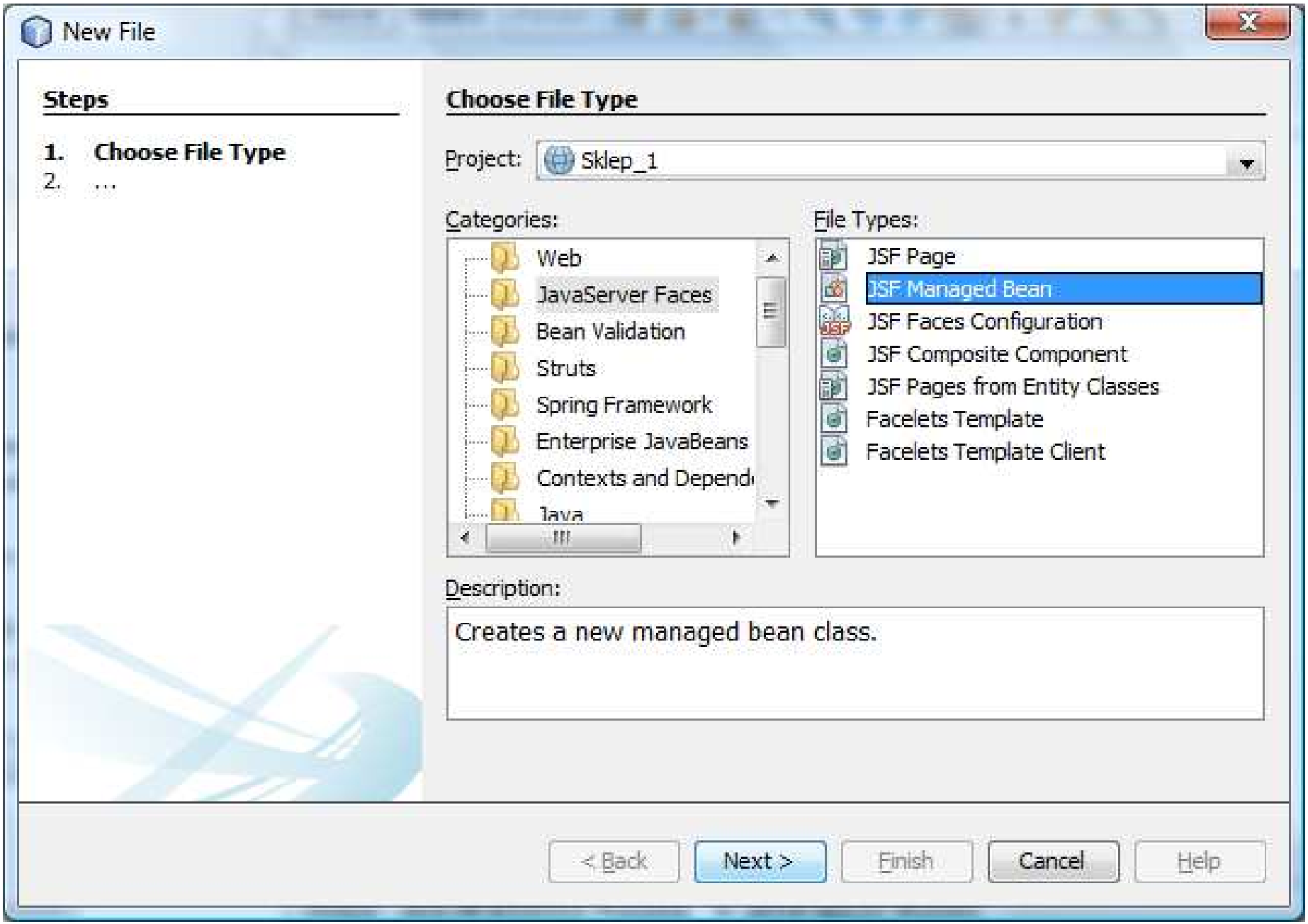
Dodanie strony typu JavaServer Faces do wstawiania danych produktu:
New/Other/JavaServer Faces/JSF Page



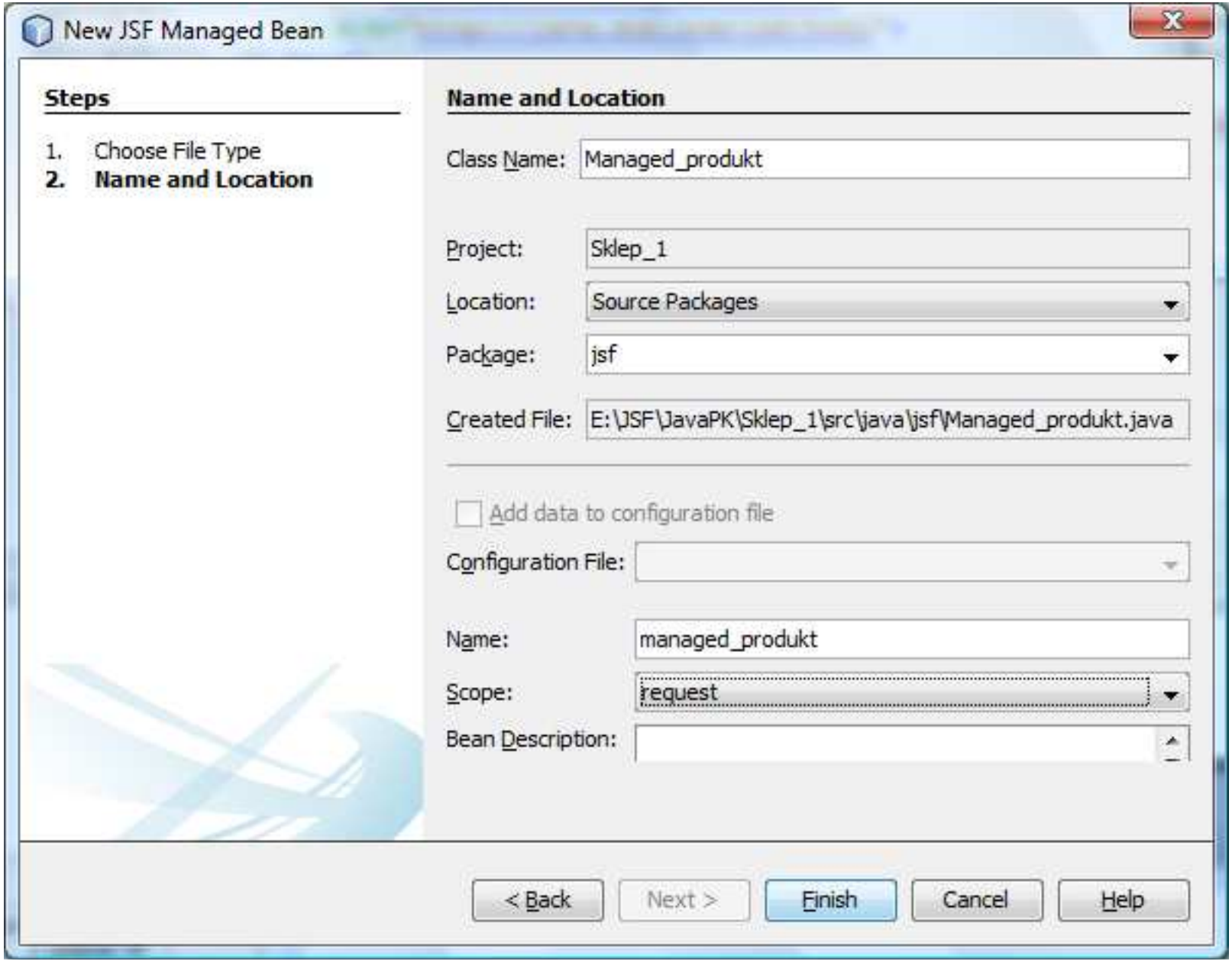
Dodanie strony typu JavaServer Faces do wstawiania danych produktu: **File Name: dodaj_produkt, Folder: jsf i Finish**



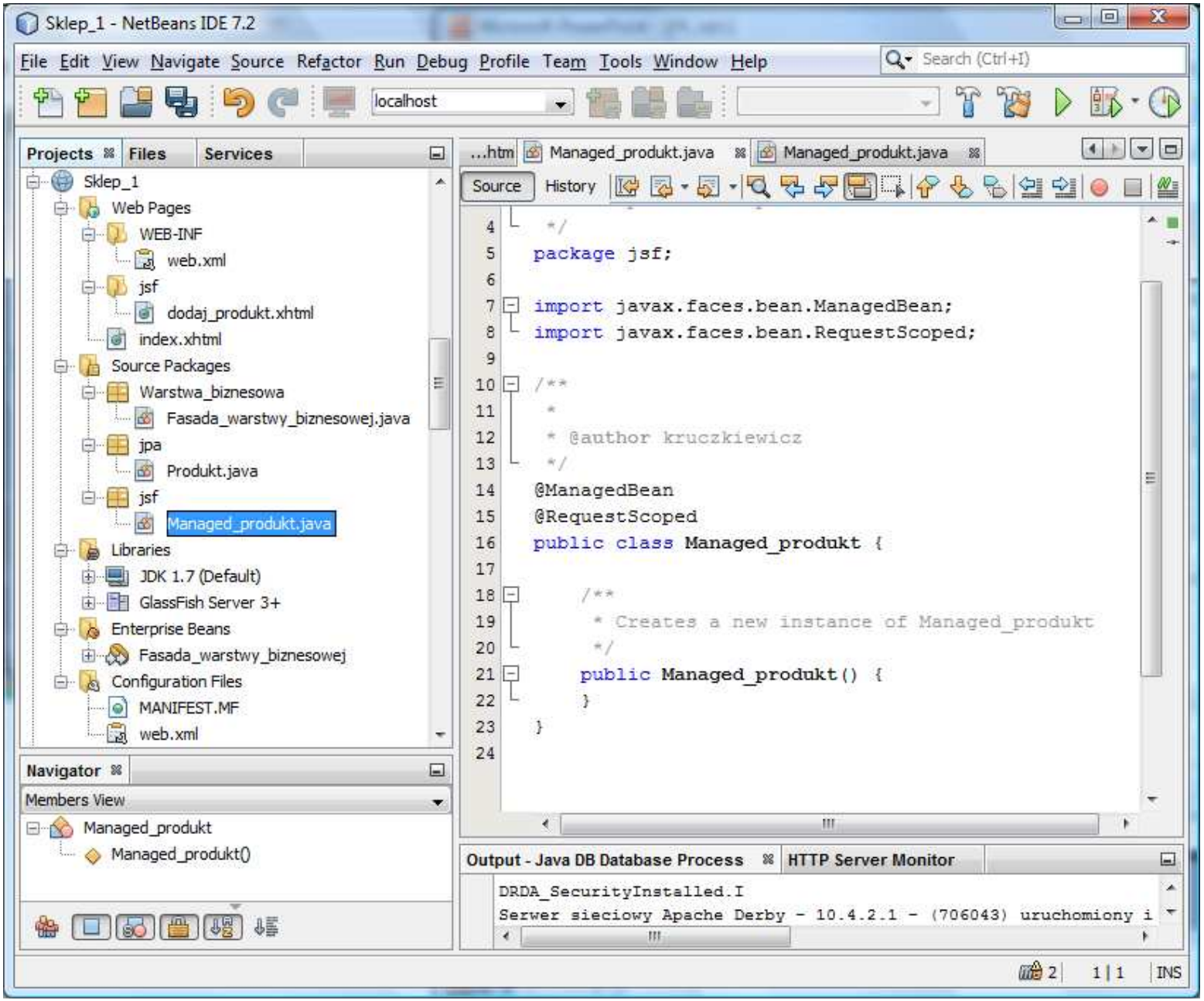
Dodanie klasy typu Managed Bean: **New/Other/JavaServer Faces/JSF Managed Bean i Next**



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



Wygenerowany kod klasy Managed Bean



```
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_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: 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_produkt obsługujących dodawanie produktu (**dodaj_produkt**) po pobraniu danych z formularza za pomocą atrybutów: nazwa, cena, promocja i wywołaniu metody **utworz_produkt** ziarna EJB z obiektu fasada klasy typu Fasada_warstwy_biznesowej oraz wyświetlanie danych za pomocą metody **dane_produktu** pobranych z warstwy biznesowej od obiektu typu EJB fasada za pomocą metody **dane_produktu**

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

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

Uzupełniona treść strony **dodaj_produkt**

```
<?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** umożliwiająca wprowadzanie danych produktu do obiektu typu Managed_produkt w dwóch kolumnach za pomocą komponentów **outputLabel** oraz **inputText**.

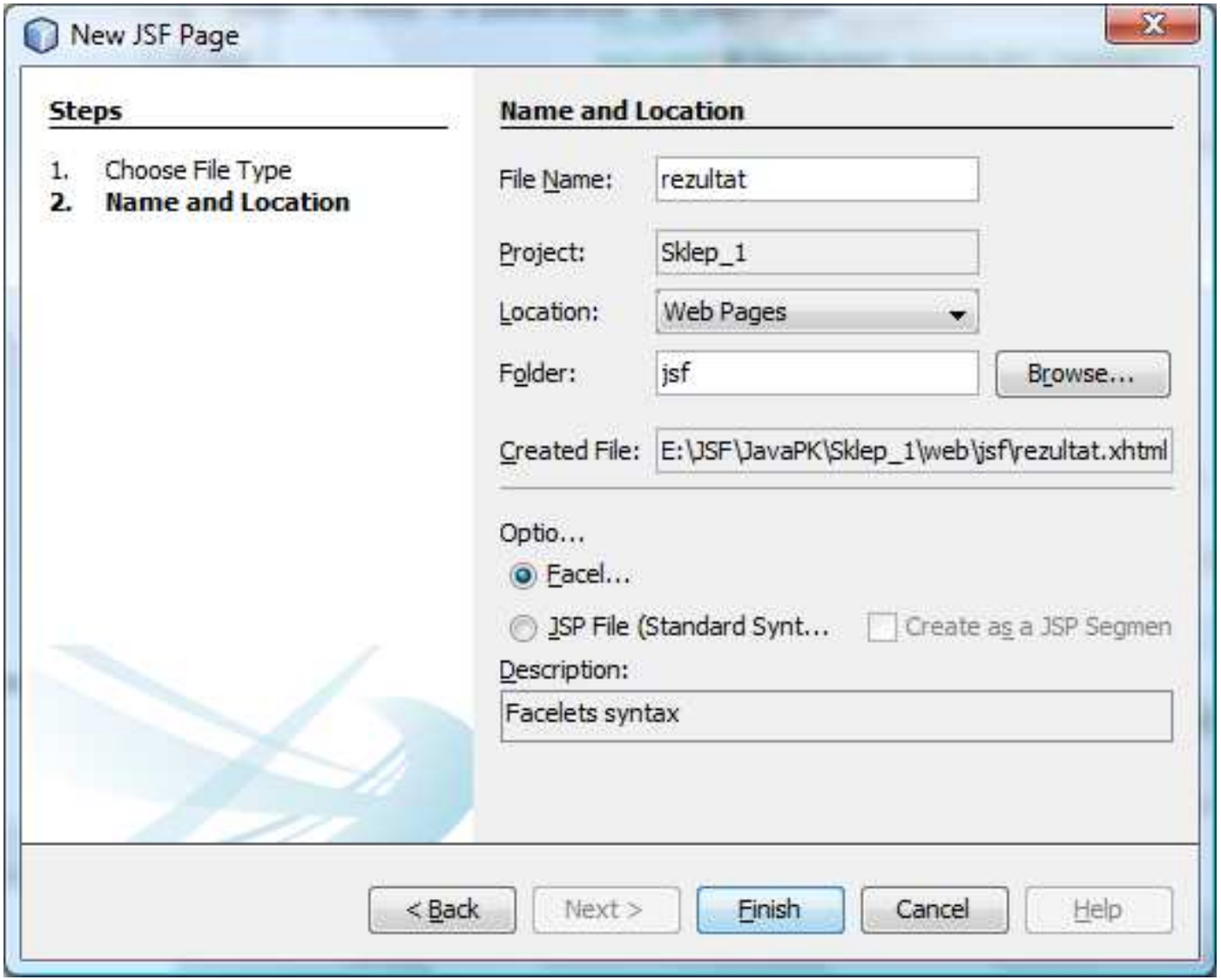
Atrybuty **required** i **requiredMessage** obsługują błąd wynikający z braku wprowadzenia danych do komponentów typu **inputText**

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

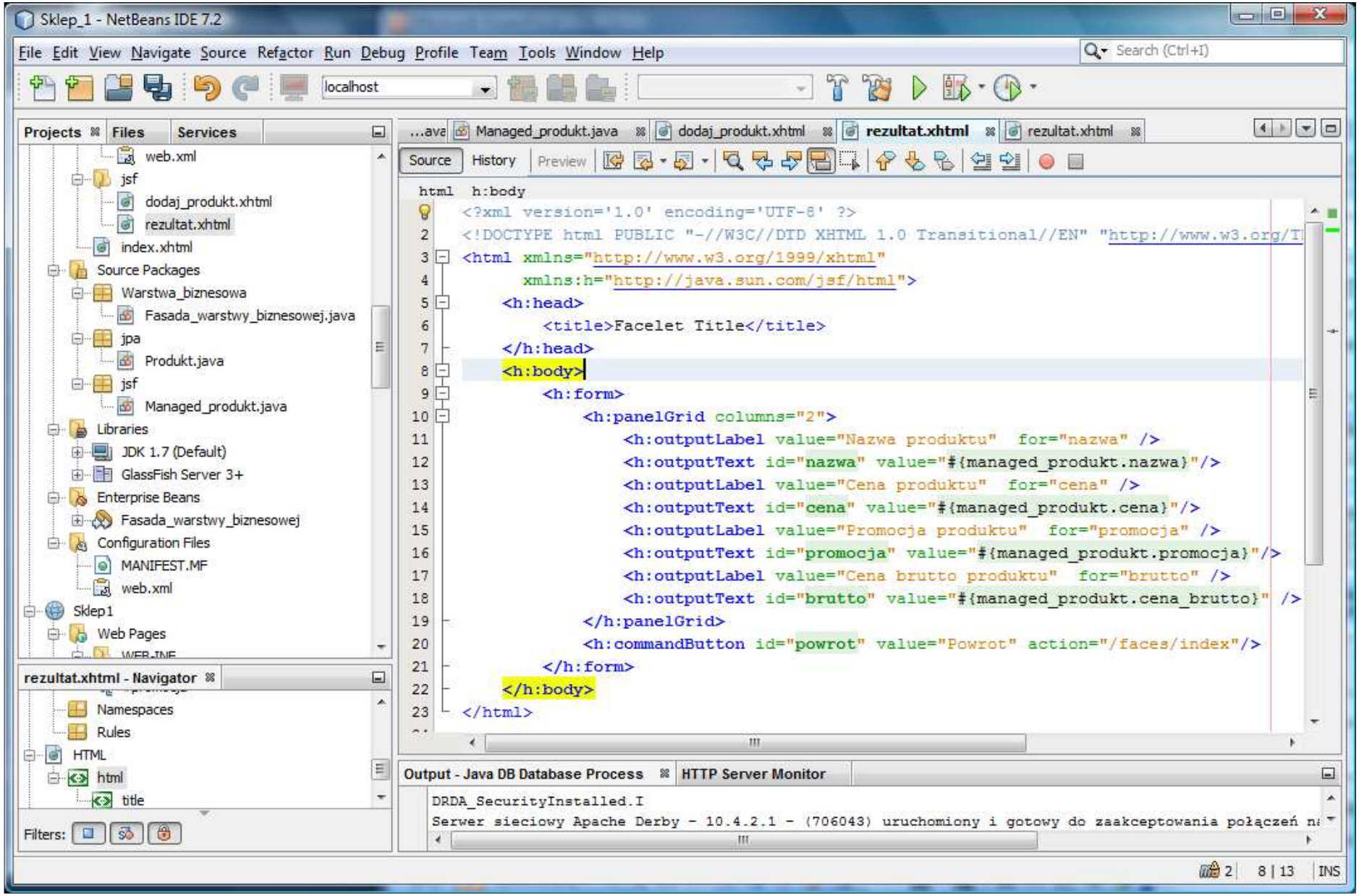
Znacznik **<h:commandLink** pozwala powrócić do strony, której nazwę zwraca bezparametrowa metoda dodaj_produkt z obiektu klasy Managed_produkt (wartość atrytbutu **action**) – jest to strona **rezultat.xhtml**:

```
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_produkt typu Managed_produkt



```
<?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>
```

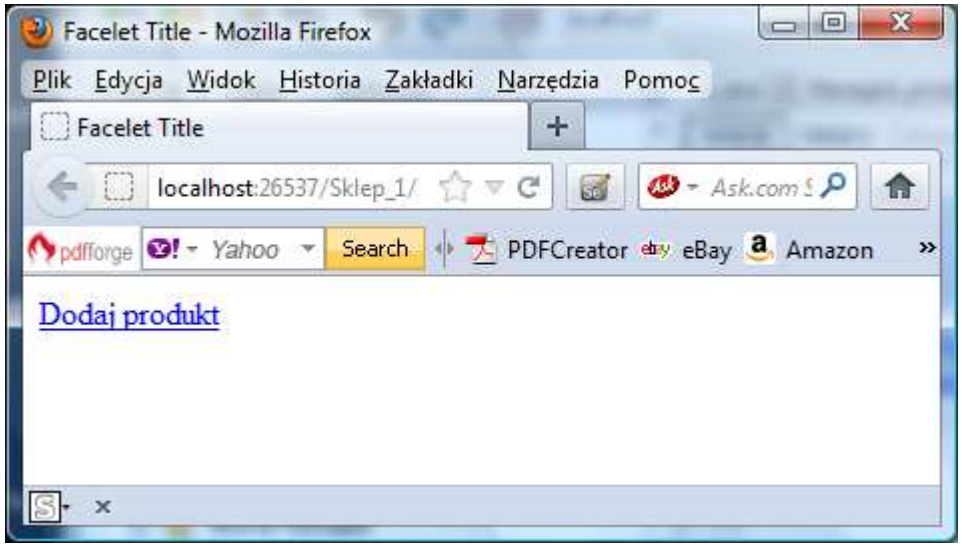
Siatka **panelGrid** umożliwiająca wyświetlanie danych produktu pobranych z obiektu typu Managed_produkt w dwóch kolumnach za pomocą komponentów **outputLabel** i **outputText**

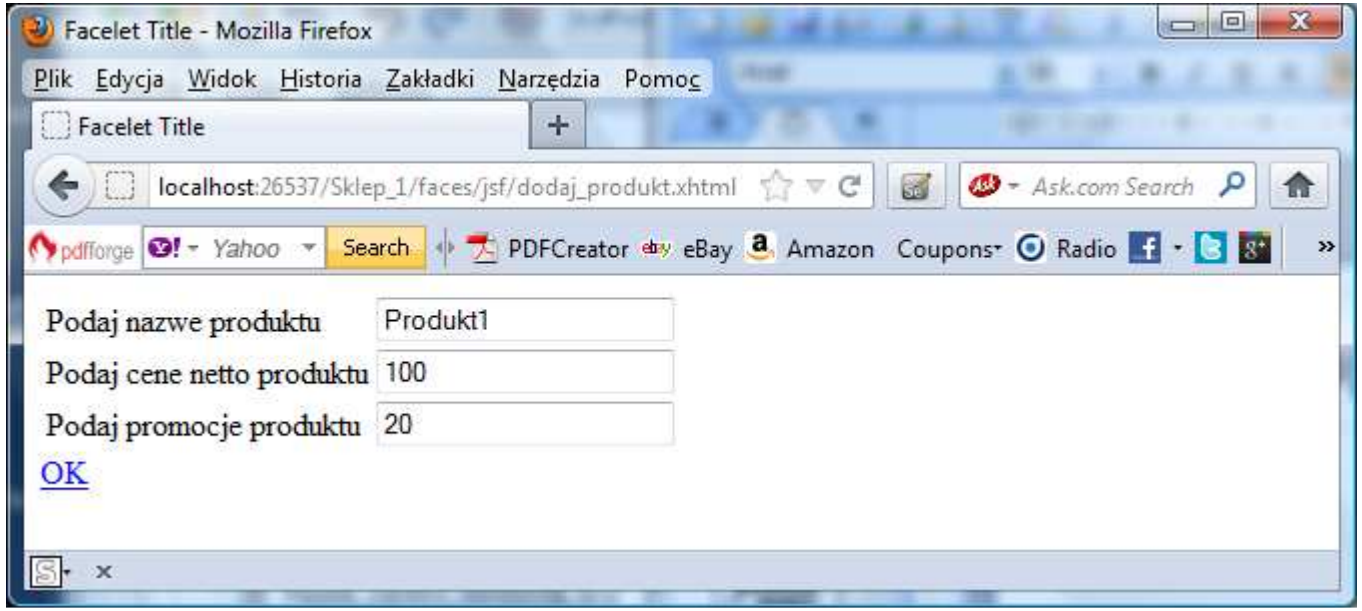
Znacznik **<h:commandButton** pozwala powrócić do strony głównej **index** (wartość atrytbutu **action**)

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

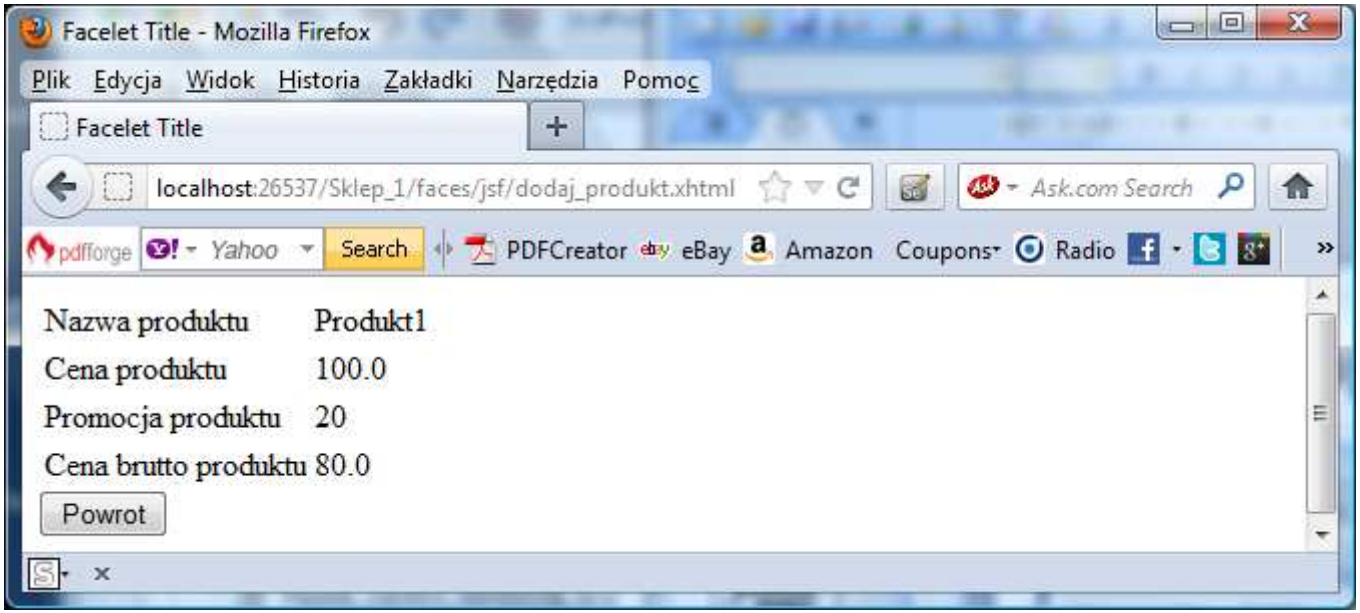
```
<?xml version='1.0' encoding='UTF-8' ?><!DOCTYPE html PUBLIC "-
    //W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
      <h:link outcome="/jsf/dodaj_produkt" value="Dodaj produkt"/>
  </h:body>
</html>
```

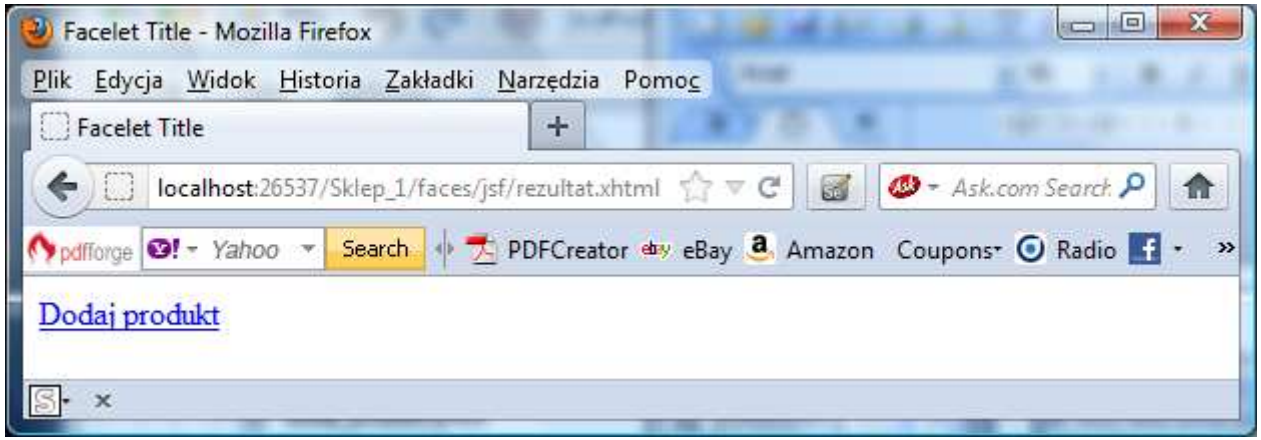
Stany aplikacji – uruchomienie strony głównej. Po wybraniu linku Dodaj produkt przejście do strony dodaj_produkt

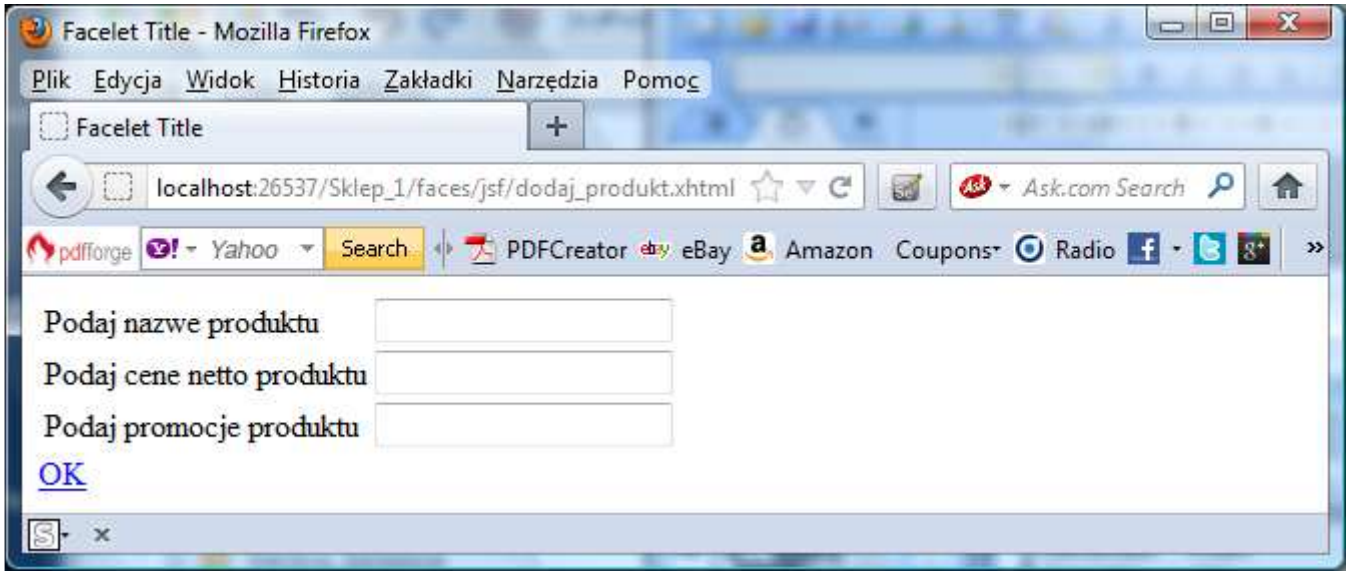


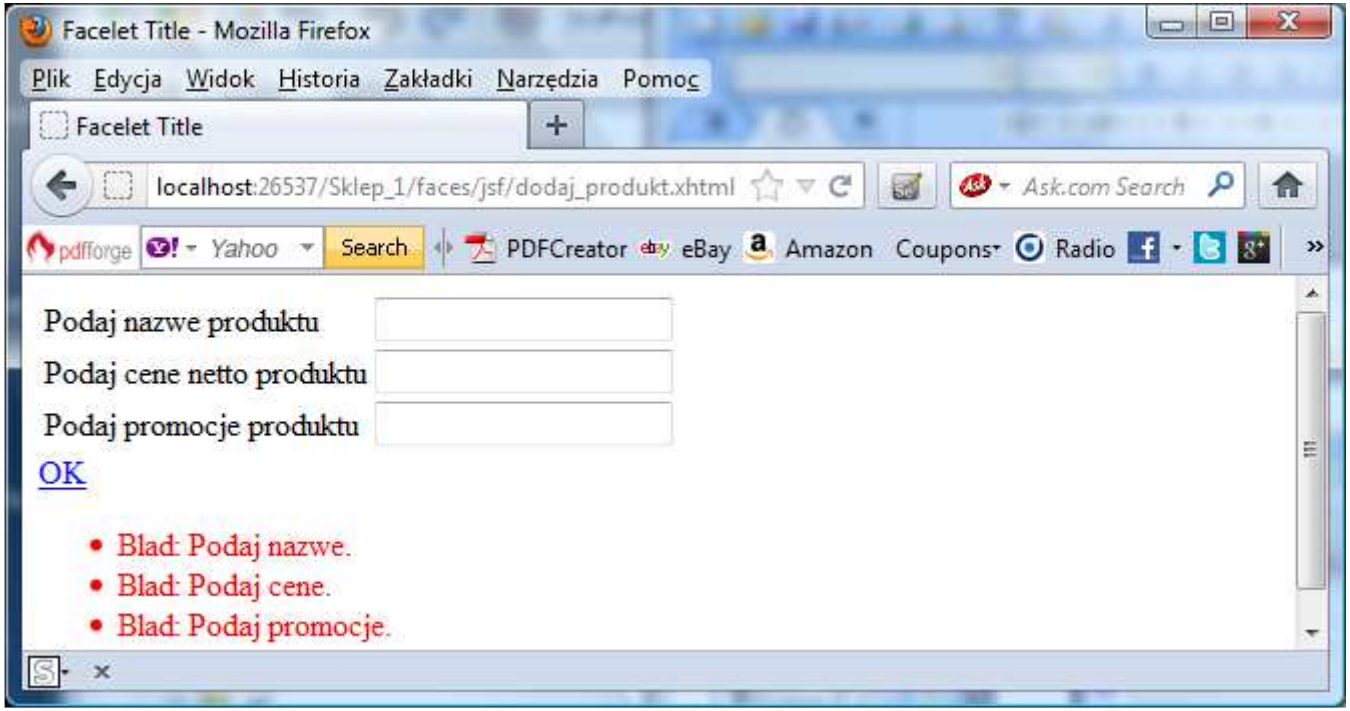


Po kliknięciu OK. na stronie **dodaj_produkt** przejście do strony **rezultat** i wyświetlenie danych. Po kliknięciu na **Powrót** przejście do strony **głównej** index





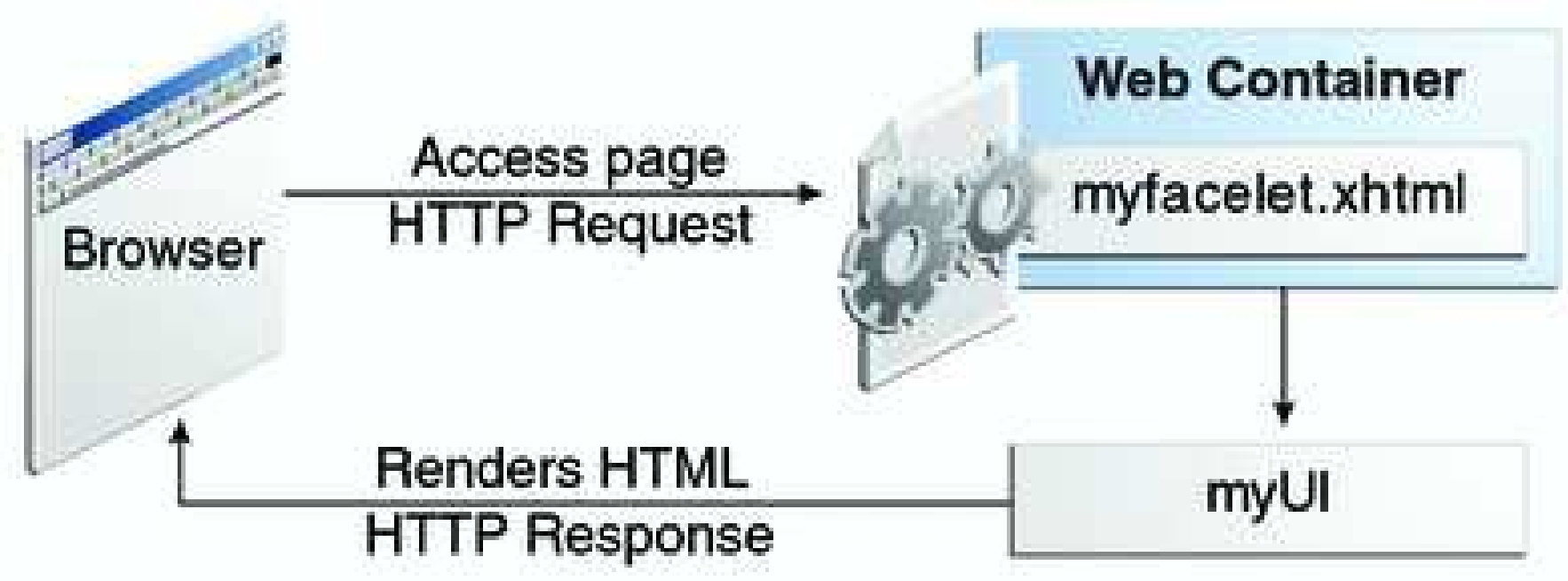




Działanie walidacji typu **required** kontrolującej brak danych

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

Interakcja między warstwa 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 obiektów, takich jak:

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