

**Wielowarstwowa aplikacja
internetowa. Wykonanie widoku
typu tabela. Pliki typu properties.**

wg

<https://docs.oracle.com/javase/7/JEETT.pdf>

**Technologie internetowe 5
(1)**

Przykład 12 – wielowarstwowa aplikacja internetowa.

Przetwarzanie wielu obiektów typu Produkt.

Opis architektury aplikacji wielowarstwowej w materiałach do wykładu 1.

Rozdziały: 12, 32 oraz uzupełnie w rozdziałach 33, 34

Rozwijanie programu z przykładu 11 (wykład 4) – przetwarzanie wielu obiektów typu Produkt. Pełna prezentacja architektury aplikacji z przykładu 11.



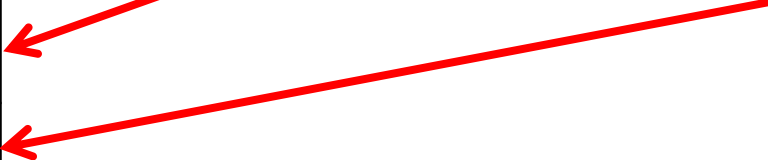
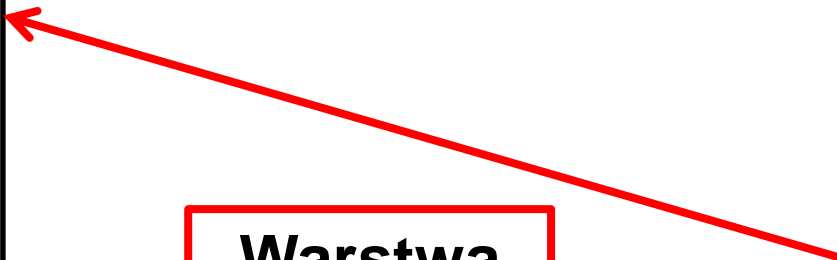
Struktura aplikacji z przykładu 11

- Web Pages
 - WEB-INF
 - jsf
 - dodaj_produkt2.xhtml
 - rezultat2.xhtml
 - resources
 - css
 - cssLayout.css
 - default.css
 - jsfcrud.css
 - obrazy
 - Dock.jpg
 - index2.xhtml
 - logo.xhtml
 - template.xhtml

- Source Packages
 - warstwabiznesowaejb
 - Fasada_warstwy_biznesowej.java
 - warstwabiznesowaejb.entity
 - Produkt.java
 - warstwainternetowajsf
 - Managed_produkt.java

Warstwa biznesowa

Warstwa internetowa



- Test Packages
- Libraries
 - JDK 1.8 (Default)
 - GlassFish Server
- Test Libraries
- Enterprise Beans
 - Fasada_warstwy_biznesowej1
- Configuration Files
 - MANIFEST.MF
 - web.xml

Warstwa internetowa aplikacji

Klasa typu Managed_produk

Metody służące do
powiązania danych z
komponentami JSF

```
package warstwainternetowajsf;  
import warstwabiznesowaejb.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;
```

```
    private int stan = 1;
```

```
    public Managed_produk() { ...2 lines }
```

```
    public Fasada_warstwy_biznesowej1 getFasada() { ...3 lines }
```

```
    public void setFasada(Fasada_warstwy_biznesowej1 fasada) { ...3 lines }
```

```
    public String getNazwa() { ...3 lines }
```

```
    public void setNazwa(String nazwa) { ...3 lines }
```

```
    public String getCena() { ...3 lines }
```

```
    public void setCena(String cena) { ...3 lines }
```

```
    public String getPromocja() { ...3 lines }
```

```
    public void setPromocja(String promocja) { ...3 lines }
```

```
    public String getCena_brutto() { ...3 lines }
```

```
    public void setCena_brutto(String cena_brutto) { ...3 lines }
```

```
    public int getStan() { ...3 lines }
```

```
    public void setStan(int stan) { ...3 lines }
```

```
    public String dodaj_produk() { ...6 lines }
```

```
    public void dane_produk() { ...12 lines }
```

```
public String dodaj_produkt() {  
    String[] dane = {nazwa, cena, promocja};  
    fasada.utworz_produkt(dane);  
    dane_produktu();  
    return "rezultat2";  
}
```

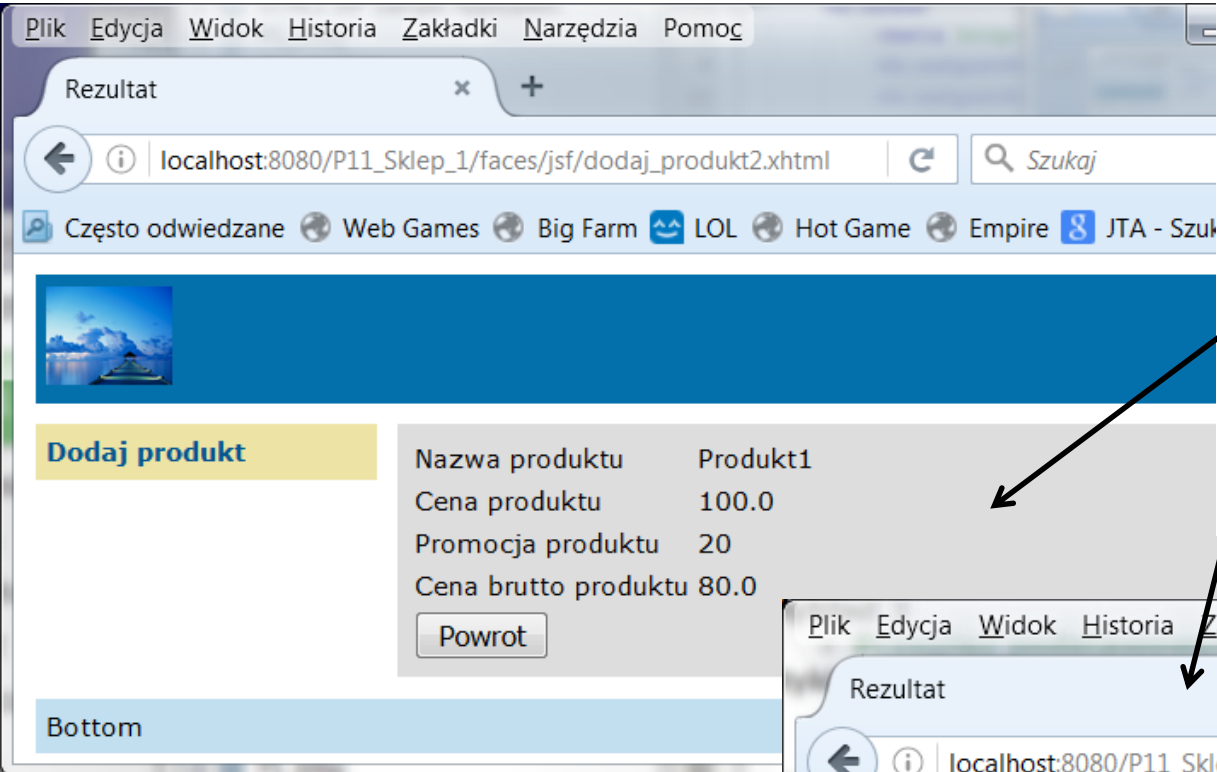
Klasa
Managed_produkt

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

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

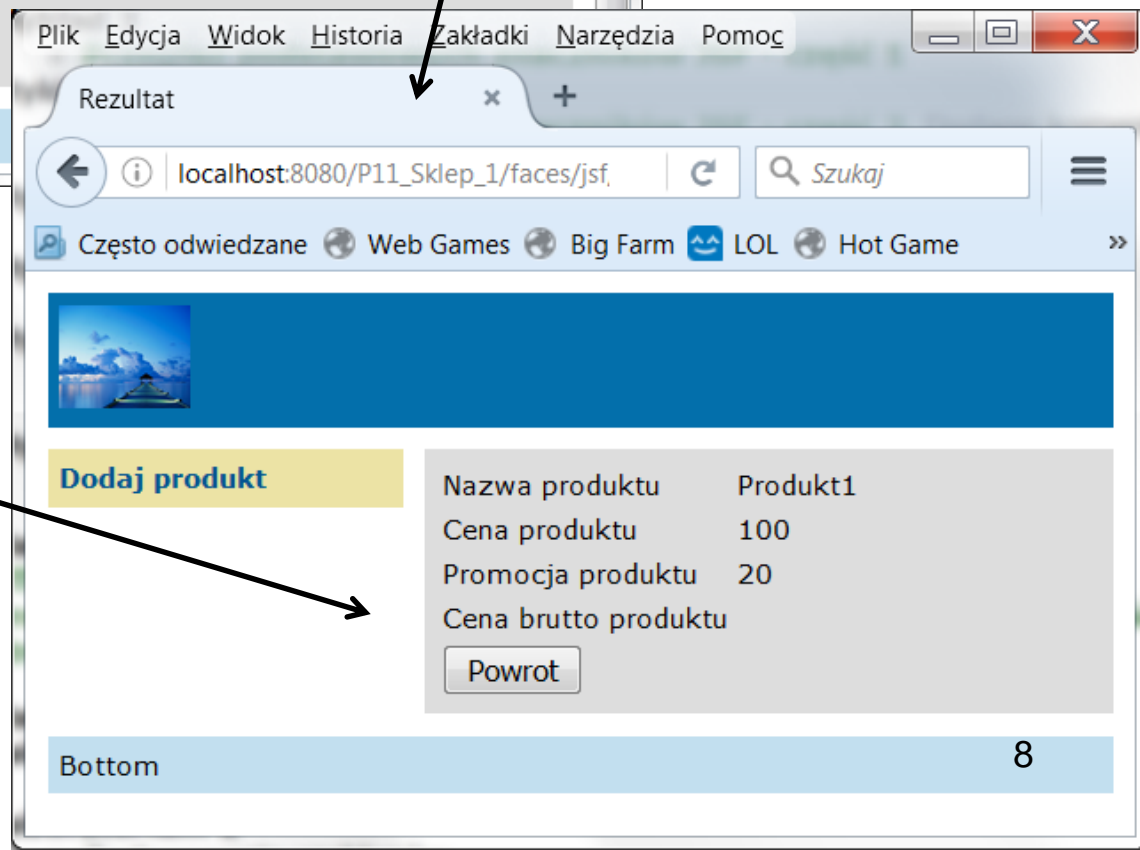
Metoda do obsługi
nawigacji pomiędzy
stronami JSF

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, czyli obiektu **fasada** klasy typu **Fasada_warstwy_biznesowej** oraz wyświetlanie danych za pomocą metody **dane_produktu**, pobranych z warstwy biznesowej od obiektu **fasada** typu EJB za pomocą metody **dane_produktu**



Dwukrotne wstawienie tych samych danych za pomocą strony **dodaj_produk2.xhtml** – wyświetlenie strony **rezultat2.xhtml**:

- 1) Po wstawieniu pierwszy raz danych
- 2) Po wstawieniu ponownie tych samych danych



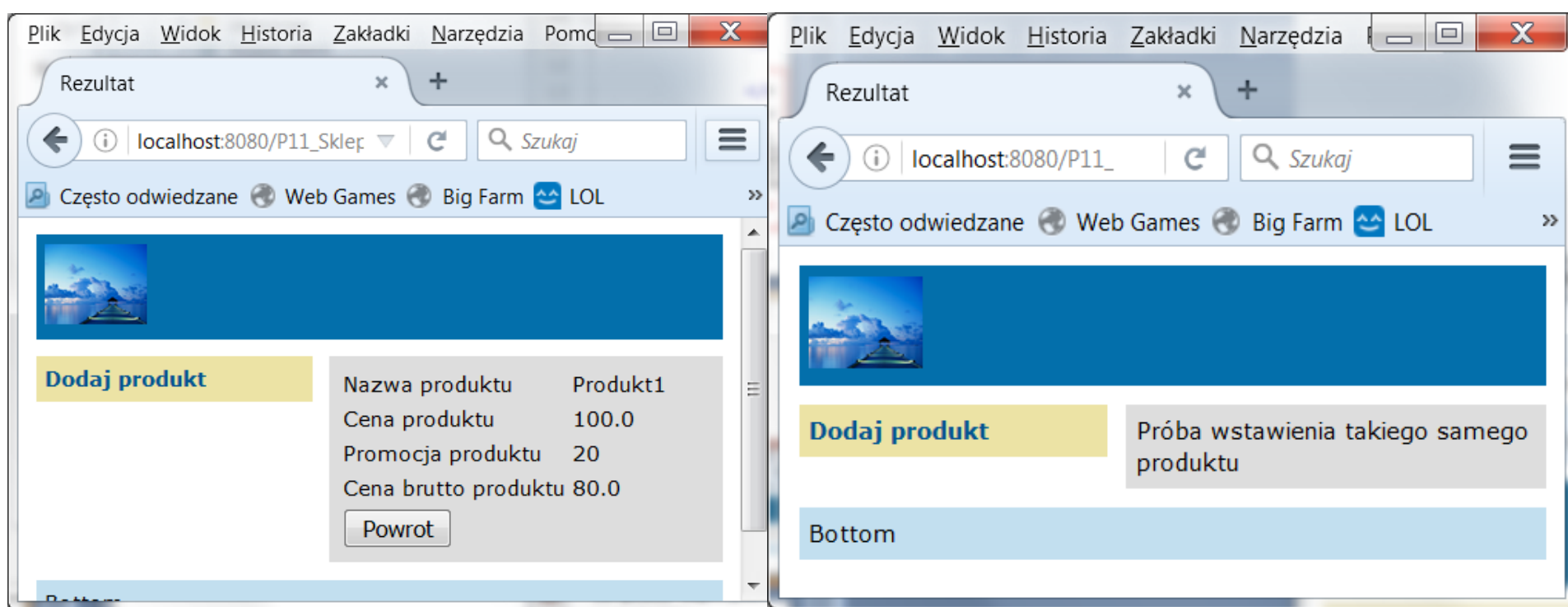
Brak danej **Cena brutto produktu**, ponieważ nie wstawiono drugi raz tego samego produktu. Pozostałe pola są wyświetlane z powodu przechowania ich wartości po wstawianiu danych w atrybutach obiektu typu **Managed_produk2**.

Strona rezultat2.xhtml – dodanie atrybutu **rendered** w celu warunkowego wyświetlania zawartości formularza

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
xmlns:h="http://xmlns.jcp.org/jsf/html">
<body>
<ui:composition template=" ../template.xhtml">
<ui:define name="title">Rezultat</ui:define>
<ui:define name="content">
<h:form>
<h:outputTest escape="false"
value= " Próba wstawienia takiego samego produktu "
rendered = "#{managrod_produkt.stan==0}"/>
<h:panelGrid columns="2"
rendered = "#{managrod_produkt.stan!=0}"/>
<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:commandButton id="powrot" value="Powrot" action="/faces/index2"/>
</h:panelGrid></h:form>
</ui:define>
</ui:composition>
</body>
</html>
```

Wprowadzenie do strony rezultat2.xhtml opartej na szablonie, do części oznaczonej jako **content** treść znacznika **h:form** z przykładów 8,9 ze strony **rezultat.xhtml**

Dzięki atrybutowi **rendered** strona **rezultat2.xhtml** nie wyświetla już danych produktu, który nie został wstawiony w aplikacji.

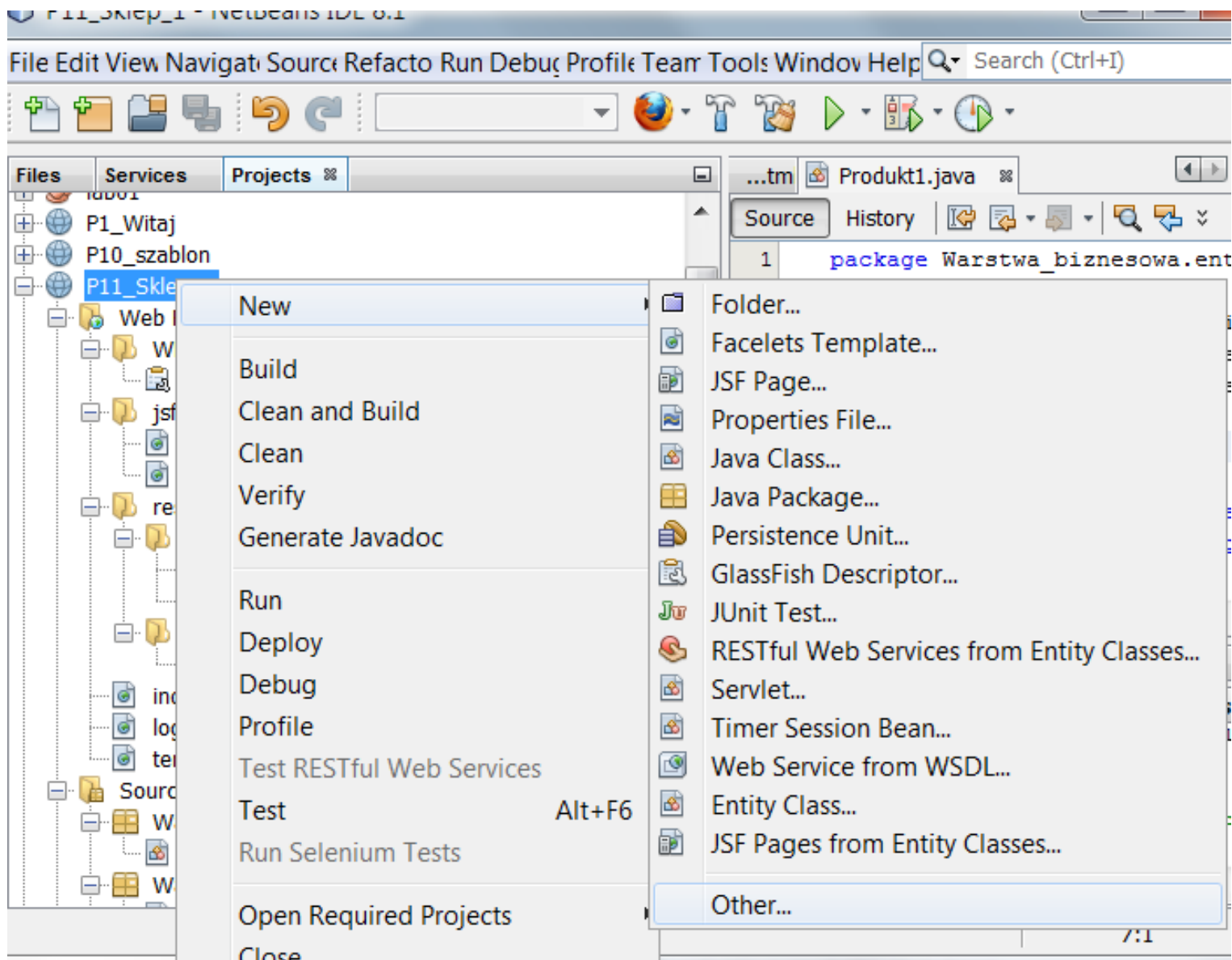


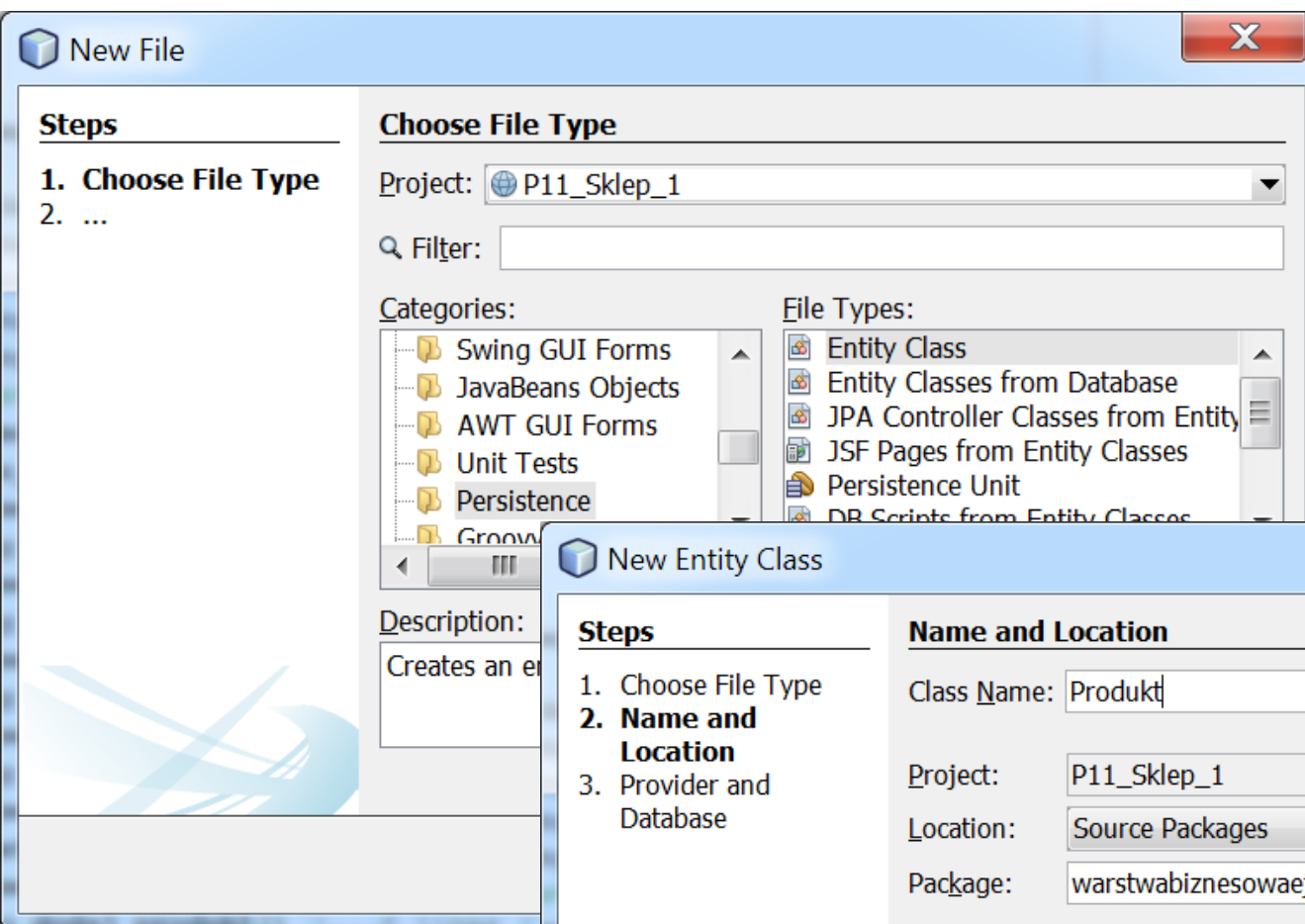
Warstwa biznesowa aplikacji

Budowa obiektów z warstwy biznesowej środowisku NetBeans

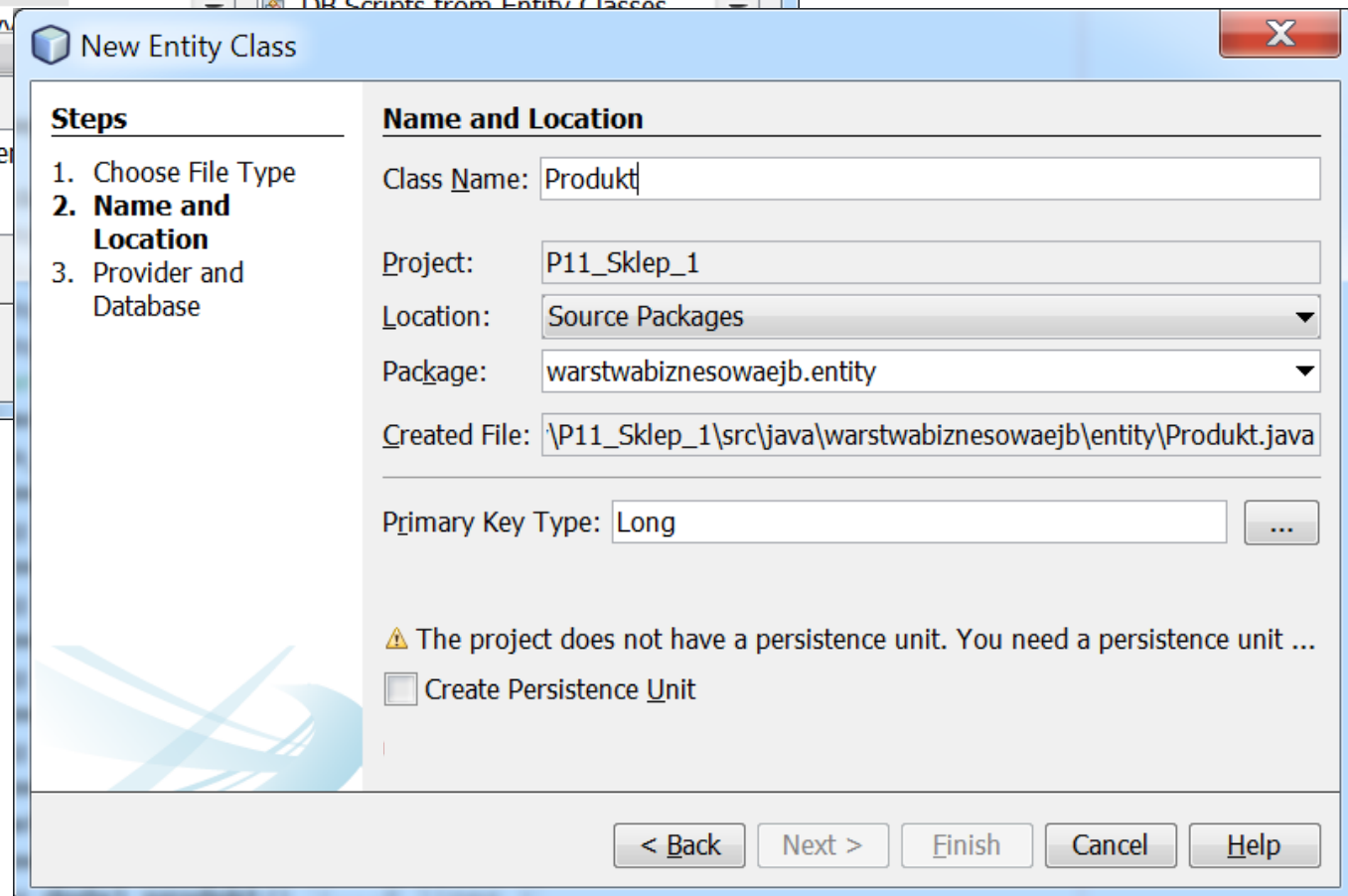
- Utworzono klasę **Produkt** reprezentującą klasę typu **Entity** tworzącą model danych warstwy biznesowej
- Utworzono klasę typu **EJB** reprezentującą klasę typu zdalna Fasada w warstwie biznesowej, która przetwarza dane wielu obiektów typu **Produkt**

Dodanie klasy **Produkt** typu **Entity** do projektu





Dodanie klasy
Produkt typu
Entity do
projektu (cd)



```
package warstwabiznesowaejb.entity;
import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

Definicja klasy
Produkt typu
Entity

@Entity

```
public class Produkt implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String nazwa;
    private float cena;
    private int promocja;
    public Long getId() { ...3 lines }
    public void setId(Long id) { ...3 lines }
    public String getNazwa() { ...3 lines }
    public void setNazwa(String nazwa) { ...3 lines }
    public float getCena() { ...3 lines }
    public void setCena(float cena) { ...3 lines }
    public int getPromocja() { ...3 lines }
    public void setPromocja(int promocja) { ...3 lines }
```

Metody dostępu do
atrybutów klasy Produkt

Definicja klasy **Produkt** typu **Entity (cd)**

```
@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}
@Override
public boolean equals(Object object) {
    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)))
        if (!nazwa.equals(other.nazwa) || cena != other.cena ||
            promocja != other.promocja)
            return false;
    return true;
}
```

metoda wspierająca zachowanie
integralności danych



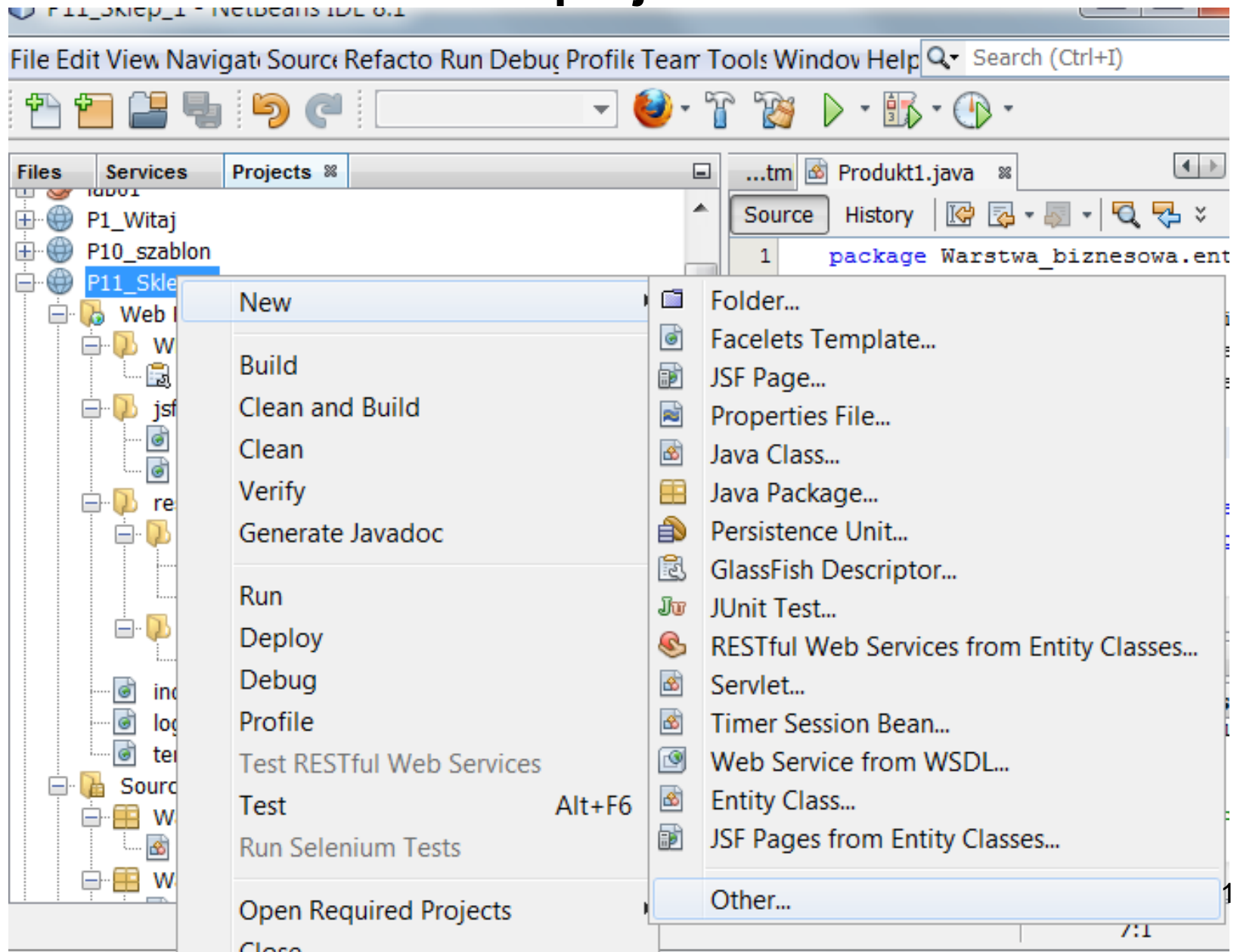
Definicja klasy **Produkt** typu **Entity** (cd)

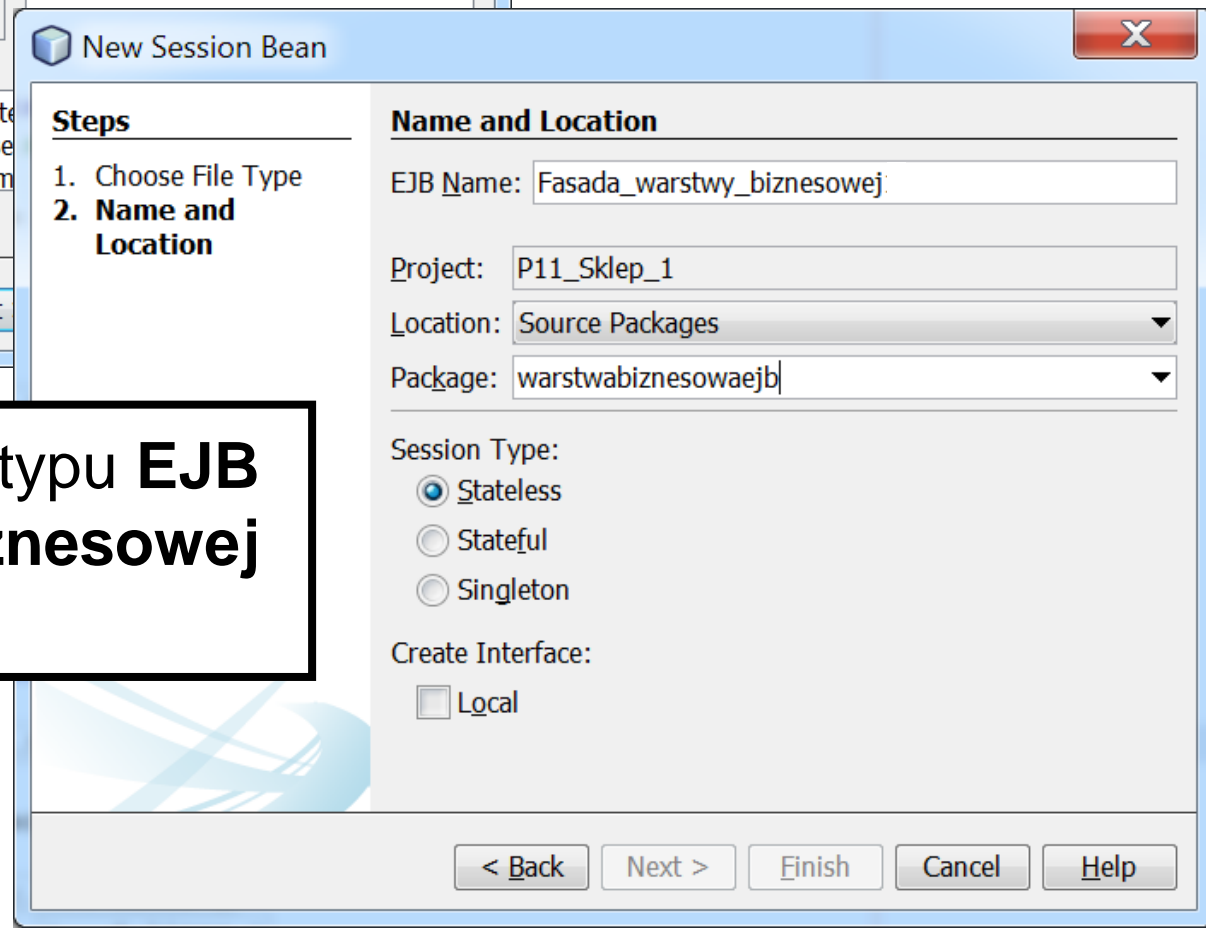
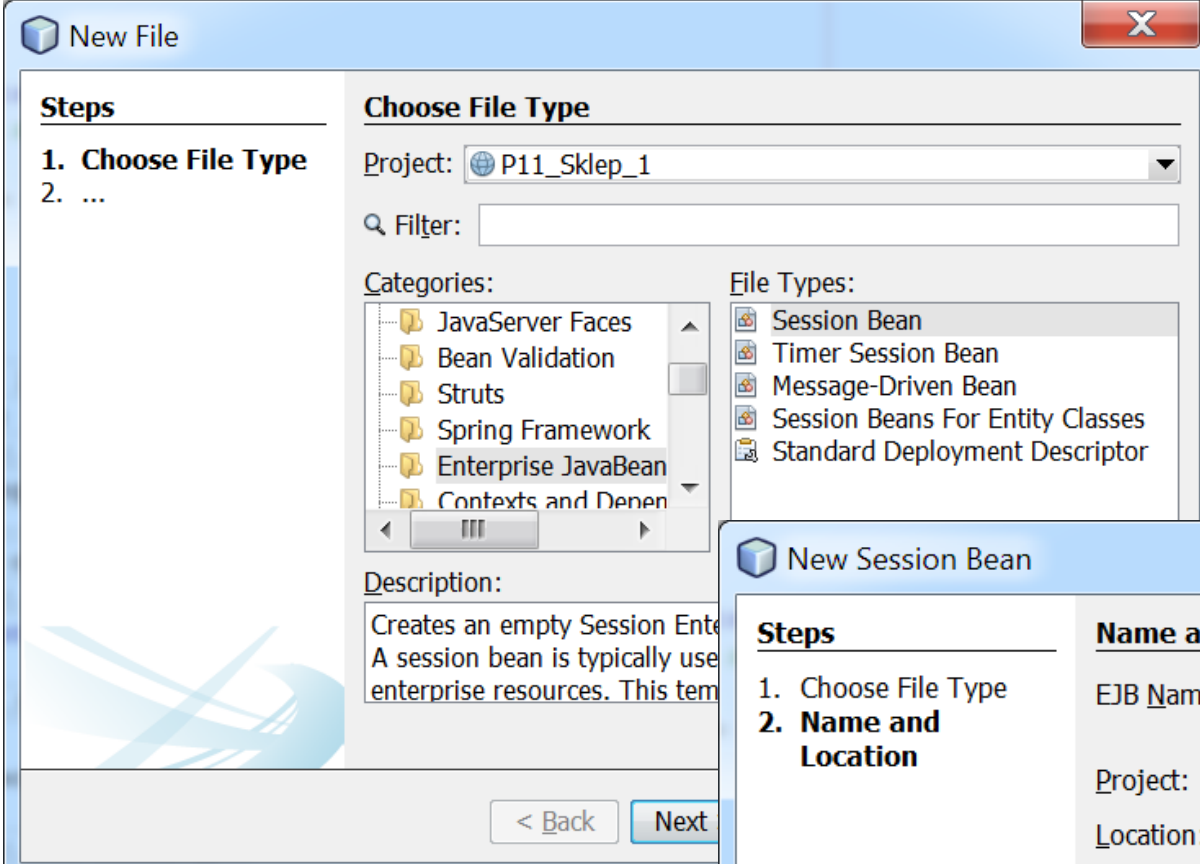
```
@Override  
public String toString() {  
    return "Produkt{" + "id=" + id + ", nazwa=" + nazwa +  
        ", cena=" + cena + ", promocja=" + promocja + '}';  
}
```

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

**metoda realizująca logikę biznesową
klasy**

Dodanie komponentu typu EJB – Fasada_warstwy_biznesowej do projektu





Dodanie komponentu typu **EJB**
Fasada_warstwy_biznesowej
do projektu

Definicja komponentu typu EJB Fasada_warstwy_biznesowej

```
package warstwabiznesowaejb;  
import java.util.ArrayList;  
import javax.ejb.Stateless;  
import warstwabiznesowaejb.entity.Produkt;
```

Zmienna typu **static** do nadawania unikatowych wartości id dla obiektu typu Produkt

```
@Stateless
```

```
public class Fasada_warstwy_biznesowej {
```

```
    static long klucz = 0;
```

```
    private ArrayList<Produkt> produkty = new ArrayList();
```

```
    boolean stan = false;
```

Jeśli zmienna stan ma wartość **false** – oznacza to próbę wprowadzenia produktu o danych, które nie są unikatowe

```
    public ArrayList<Produkt> getProdukty() { ...3 lines }
```

```
    public void setProdukty(ArrayList<Produkt> produkty) { ...3 lines }
```

Przechowywanie listy produktów o unikatowych danych

Definicja komponentu EJB Fasada_warstwy_biznesowej (cd)

```
public void utworz_produkt(String dane[]) {  
    Produkt produkt = new Produkt();  
    klucz++;  
    produkt.setId(new Long(klucz));  
    produkt.setNazwa(dane[0]);  
    produkt.setCena(Float.parseFloat(dane[1]));  
    produkt.setPromocja(Integer.parseInt(dane[2]));  
    dodaj_produkt(produkt);  
}  
  
protected void dodaj_produkt(Produkt produkt) {  
    if (!produkty.contains(produkt)) {  
        produkty.add(produkt);  
        stan = true;  
    } else  
        stan = false;  
}
```

Dodawanie nowego produktu

Dodawanie nowego produktu – sprawdzenie, czy nowy obiekt jest unikatowy. Wartość zmiennej stan równy **true** oznacza wprowadzenie danej

Definicja komponentu **EJB Fasada_warstwy_biznesowej** (cd)

Utworzenie modelu ostatnio wstawionego obiekt typu **Produkt**, który pobrany jest z kolekcji **produkty** i na podstawie jego danych wykonanie modelu danych widoków w postaci tablicy obiektów typu String

```
public String[] dane_produktu() {  
    if (stan) {  
        Produkt produkt = produkty.get(produkty.size() - 1);  
        String nazwa = produkt.getNazwa();  
        String cena = "" + produkt.getCena();  
        String promocja = "" + produkt.getPromocja();  
        String cena_brutto = "" + produkt.cena_brutto();  
        String dane[] = {nazwa, cena, promocja, cena_brutto};  
        return dane; }  
    return null;  
}
```

Dane ostatnio wprowadzonego produktu przeznaczone do prezentacji, Wartość **null** oznacza brak dodania nowego produktu

Przykład 13 – wielowarstwowa aplikacja internetowa.

Przetwarzanie wielu obiektów typu **Produkt**.

Zastosowanie komponentu typu **h:dataTable** do prezentacji wielu danych typu **Produkt** na stronie www

Warstwa biznesowa

Dodanie nowej metody do klasy **Fasada_warstwy_biznesowej**, generującej model danych dla komponentu typu **dataTable**

```
public ArrayList<ArrayList<String>> items() {  
    ArrayList<ArrayList<String>> dane = new ArrayList();  
    for (Produkt p : produkty) {  
        ArrayList<String> wiersz = new ArrayList();  
        wiersz.add(p.getId().toString());  
        wiersz.add(p.getNazwa());  
        wiersz.add("" + p.getCena());  
        wiersz.add("" + p.getPromocja());  
        wiersz.add("" + p.cena_brutto());  
        dane.add(wiersz);  
    }  
    return dane;  
}
```


Warstwa internetowa – klasa Managed_produk

```
package warstwainternetowajs;
```

```
import warstwabiznesowaejb.Fasada_warstwy_biznesowej;
```

```
import javax.ejb.EJB;
```

```
import javax.faces.bean.ManagedBean;
```

```
import javax.faces.bean.RequestScoped;
```

```
import javax.faces.model.DataModel;
```

```
import javax.faces.model.ListDataModel;
```

**Zmodyfikowana zawartość klasy
Managed_produk**

```
@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;
```

```
private DataModel items;
```

```
private int stan = 1;
```

```
public Managed_produk() { }
```

```
public Fasada_warstwy_biznesowej getFasada() { return fasada; }
```

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

DataModel – model danych komponentu
dataTable

Stan – zmienna oznaczająca warunki
renderowania

Metody dostępu do danych, podobnie jak w przykładzie 12

```
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;   }  
public int getStan()              {   return stan;   }  
public void setStan(int stan)      {   this.stan = stan;   }
```

Metody obsługujące strony **dodaj_produk2.xhtml** oraz zawartość strony **rezultat2.xhtml**, podobnie jak w przykładzie 12

```
public String dodaj_produk2() {  
    String[] dane = {nazwa, cena, promocja};  
    fasada.utworz_produk2(dane);  
    dane_produk2();  
    return "rezultat2";  
}
```

```
public void dane_produk2() {  
    stan = 1;  
    String[] dane = fasada.dane_produk2();  
    if (dane == null) {  
        stan = 0;  
    } else {  
        nazwa = dane[0];  
        cena = dane[1];  
        promocja = dane[2];  
        cena_brutto = dane[3]; }  
}
```

Nowe metody, obsługujące model widoku – **DataModel** jako obiekt **items**

```
public DataModel utworz_DataModel() {  
    return new ListDataModel(fasada.items());  
}
```

```
public DataModel getItems() {  
    if (items == null) {  
        items = utworz_DataModel();  
    }  
    return items;  
}
```

```
public void setItems(DataModel items) {  
    this.items = items;  
}
```

← Nowe metody
Utworzenie modelu komponentu **dataTable** na podstawie kolekcji zawierających elementy reprezentujące wiersz tabeli (kolekcja obiektów typu String reprezentująca atrybuty obiektu typu **Produkt** oraz cenę brutto). Kolekcja ta jest podawana przez metodę **items** wywołaną od obiektu **fasada** typu EJB (slajd 24)

Modyfikacja szablonu **template.xhtml** – dodanie w części przeznaczonyj na menu (id=left) linku do strony **lista_produkow.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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

<h:head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <h:outputStylesheet library= "css" name="default.css" />
  <h:outputStylesheet library= "css" name="cssLayout.css"/>
  <title><ui:insert name="title">Facelets Template</ui:insert></title>
</h:head>

<h:body>
  <div id="top">
    <ui:insert name="top">Top</ui:insert>
  </div>
```

Usunięto z szablonu strony logo.xhtml z części o id="top"

```
<div>
  <div id="left">
    <h:link outcome="/faces/jsf/dodaj_produkt2" value="Dodaj produkt"/><br/>
    <h:link outcome="/faces/jsf/lista_produktow" value="Lista produktow"/>
  </div>
  <div id="content" class="left_content">
    <ui:insert name="content">Content</ui:insert>
  </div>
</div>
<h:panelGroup id="messagePanel" layout="block">
  <h:messages errorStyle="color: red" infoStyle="color: green" />
</h:panelGroup>

<div id="bottom">
  <ui:insert name="bottom">Bottom</ui:insert>
</div>
</h:body>
</html>
```

**Dodany link do strony
wyświetlającej tabelę z
produktami
(slajdy 31-33)**

Zawartość **nowej strony** lista_produkow.xhtml do wyświetlania listy produktów dodana do projektu- **New/Other/JavaServer Faces/ Facelets Template Client...**

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
xmlns:h="http://xmlns.jcp.org/jsf/html,"
xmlns:f="http://xmlns.jcp.org/jsf/core">

<body>
<ui:composition template=" ../template.xhtml">
<ui:define name="title">
<h:outputText value="#{bundle.Lista_produkow_tytul}"></h:outputText>
</ui:define>

<ui:define name="content">
<h:form styleClass="jsfcrud_list_form">

<h:outputText escape="false" value="#{bundle.Lista_produkow_pusta}"
rendered="#{managed_produk.items.rowCount == 0}"/>

<h:panelGroup rendered="#{managed_produk.items.rowCount > 0}">
```

Jeżeli brak danych pobranych z modelu **items** typu `DataModel` (`rowCount==0`), wtedy wyświetla się napis **bundle.Lista_produkow_pusta** (czyli Brak danych), w przeciwnym wypadku wyświetla się tabelę `<h:dataTable>` (następny slajd)

```
<h:dataTable value="#{managed_produkt.items}" var="item" border="0"
```

Komponent typu **dataTable** zbindowany z obiektem **items** typu **DataModel**

```
cellpadding="2" cellspacing="0"  
rowClasses="jsfcrud_odd_row,jsfcrud_even_row"  
rules="all" style="border:solid 1px">
```

Item – element kolekcji **items** (zawierający dane atrybutów obiektu typu **Produkt** oraz **cenę brutto**)

```
<h:column>
```

```
<f:facet name="header"> Nagłówek kolumny tabeli dataTable
```

```
<h:outputText value="#{bundle.Lista_produktow_id}"/>
```

```
</f:facet>
```

```
<h:outputText value="#{item.get(0)}"/>
```

Kolejny element kolekcji **item** (zawierający dane atrybutu **id** obiektu typu **Produkt**), która jest elementem kolekcji **items** typu **DataModel**

```
</h:column>
```

```
<h:column>
```

```
<f:facet name="header">
```

```
<h:outputText value="#{bundle.Lista_produktow_nazwa}"/>
```

```
</f:facet>
```

```
<h:outputText value="#{item.get(1)}"/>
```

Kolejny element kolekcji **item** (zawierający dane atrybutu **nazwa** obiektu typu **Produkt**), która jest elementem kolekcji **items** typu **DataModel**

```
</h:column>
```

```
<h:column>
```

```
<f:facet name="header">
```

```
<h:outputText value="#{bundle.Lista_produktow_cena}"/>
```

```
</f:facet>
```

```
<h:outputText value="#{item.get(2)}"/>
```

Kolejny element kolekcji **item** (zawierający dane atrybutu **cena** obiektu typu **Produkt**), która jest elementem kolekcji **items** typu **DataModel**

```
</h:column>
```



```
<h:column>
  <f:facet name="header">
    <h:outputText value="#{bundle.Lista_produkow_promocja}"/>
  </f:facet>
  <h:outputText value="#{item.get(3)}"/>
</h:column>
```

Kolejny element kolekcji **item** (zawierający dane atrybutu **promocja** obiektu typu **Produkt**), która jest elementem kolekcji **items** typu **DataModel**

```
<h:column>
  <f:facet name="header">
    <h:outputText value="#{bundle.Lista_produkow_cenabrutto}"/>
  </f:facet>
  <h:outputText value="#{item.get(4)}"/>
</h:column>
</h:dataTable>
```

Kolejny element kolekcji **item** (zawierający dane ceny brutto wyznaczonej przez metodę `cena_brutto` obiektu typu **Produkt**), która jest elementem kolekcji **items** typu **DataModel**

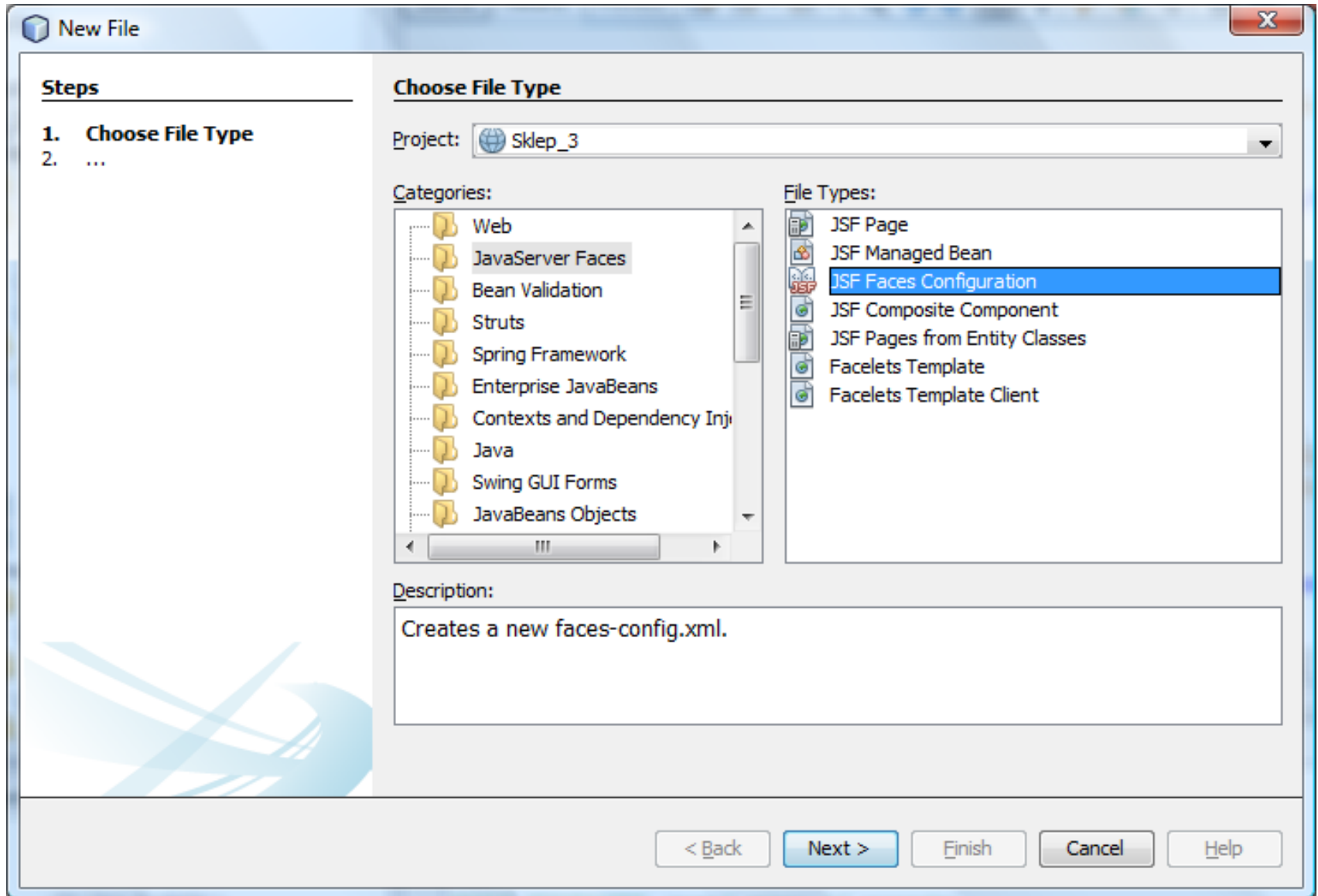
```
</h:panelGroup>
</h:form>
</ui:define>
```

```
</ui:composition>
```

```
</body>
</html>
```

Dodanie pliku konfiguracji projektu **faces-config.xml** (New/Other/JavaServer Faces/ JSF Faces Configuration)

Dokonano zmiany nazwy projektu na Sklep_3



Dodanie pliku konfiguracji projektu **faces-config.xml**

New JSF Faces Configuration

Steps

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

Name and Location

File Name:

Project:

Folder:

Created File:

< Back Next > **Finish** Cancel Help

Dodanie pliku typu **properties** do projektu: prawy klawisz na **Source Packages**, **New/Other/Other/Properties File**

The screenshot shows the NetBeans IDE 7.2 interface. The main window displays the project structure for 'Sklep_3'. The 'Source Packages' folder is selected, and the 'New' menu is open, showing the 'Other...' option. The 'New File' dialog is open, showing the 'Choose File Type' step. The 'Project' is set to 'Sklep_3'. The 'File Types' list is expanded, and 'Properties File' is selected. The 'Description' field shows: 'Creates a resource bundle (.properties) file suitable for internationalizing applications by separating out all human-visible text strings from your code. Resource bundle files can also be used to collect other types of strings, such as properties for Ant scripts. The created resource bundle contains only one locale,'.

Steps

1. Choose File Type
2. ...

Choose File Type

Project: Sklep_3

Categories:

- AWT GUI Forms
- Unit Tests
- Selenium
- Persistence
- Groovy
- Hibernate
- Web Services
- XML
- GlassFish
- WebLogic
- Other

File Types:

- SQL file
- HTML File
- XHTML File
- SVG File
- JavaScript File
- JSON File
- JNLP File
- Properties File**
- Cascading Style Sheet
- Ant Build Script
- YAML File
- INI File

Description:

Creates a resource bundle (.properties) file suitable for internationalizing applications by separating out all human-visible text strings from your code. Resource bundle files can also be used to collect other types of strings, such as properties for Ant scripts. The created resource bundle contains only one locale,

36

< Back Next > Finish Cancel Help

Dodanie pliku typu **properties** do projektu

New Properties File

Steps

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

Name and Location

File Name:

Project:

Folder:

Created File:

< Back Next > **Finish** Cancel Help

Zawartość pliku **Bundle.properties** zawierająca treść komunikatów. Należy dodatkowo wkleić plik **jsfcrud.css** do katalogu **resources/css** pobrany ze strony <http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/javapk/jsfcrud.css>

The screenshot shows an IDE window with the following components:

- Project Explorer (Left):** Shows a project named 'Sklep_3' with a tree structure including 'Web Pages', 'WEB-INF', 'jsf', 'resources', 'Source Packages', and 'Bundle.properties'.
- Bundle.properties (Center):** Contains the following text:

```
1 # To change this template, choose Tools | Templates
2 # and open the template in the editor.
3
4 Lista_produkow_tytul=Lista produktow
5 Lista_produkow_pusta=Brak danych
6 Lista_produkow_id=Id produktu
7 Lista_produkow_nazwa=Nazwa produktu
8 Lista_produkow_cena=Cena netto produktu
9 Lista_produkow_promocja=Promocja produktu
10 Lista_produkow_cenabrutto=Cena brutto
11 Lista_produkow_niedodano=Taki produkt juz istnieje
```
- Output (Bottom):** Shows the 'HTTP Server Monitor' with the following log:

```
run-deploy:
Browsing: http://localhost:26537/Sklep2
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 16 seconds)
```
- Properties (Right):** Shows the 'Bundle.properties' file with details: Name: Bundle, Extension: ..., All Files: E..., File Size: 92, Modification: 201...

Zawartość pliku **Bundle.properties** zawierająca treść komunikatów używanych na stronie **lista_produkow.xhtml**. Należy w taki sam sposób zastąpić komunikaty w pozostałych plikach.xhtml

Lista_produkow_tytul=Lista produktow

Lista_produkow_pusta=Brak danych

Lista_produkow_id=Id produktu

Lista_produkow_nazwa=Nazwa produktu

Lista_produkow_cena=Cena netto produktu

Lista_produkow_promocja=Promocja produktu

Lista_produkow_cenabrutto=Cena brutto

Lista_produkow_niedodano=Taki produkt juz istnieje

Deklaracja pliku **Bundle.properties** w pliku konfiguracyjnym **faces-config.xml**.

The screenshot shows an IDE window with the following components:

- Projects View:** Shows a project named "Sklep_3" with a tree structure including "Web Pages", "WEB-INF" (containing "faces-config.xml" and "web.xml"), "jsf" (containing "dodaj_produk2.xhtml", "lista_produk2.xhtml", "rezultat2.xhtml"), "resources" (containing "css" and "index2.xhtml", "template.xhtml"), and "Source Packages" (containing "Bundle.properties", "warstwabiznesowaejb", "Fasada_warstwy_biznesowej.java", and "warstwabiznesowaejb.entity").
- Editor:** Displays the content of "faces-config.xml" with line numbers 1-17. The XML code is as follows:

```
1 <?xml version='1.0' encoding='UTF-8'?>
2
3 <!-- ===== FULL CONFIGURATION FILE =====
4
5 <faces-config version="2.1"
6     xmlns="http://java.sun.com/xml/ns/javaee"
7     xmlns:xsi="http://www.w3.org/2001/XMLSchema-imp
8     xsi:schemaLocation="http://java.sun.com/xml/ns/
9
10 <application>
11     <resource-bundle>
12         <base-name>/Bundle</base-name>
13         <var>bundle</var>
14     </resource-bundle>
15 </application>
16 </faces-config>
17
```
- Properties View:** Shows a table with the following data:

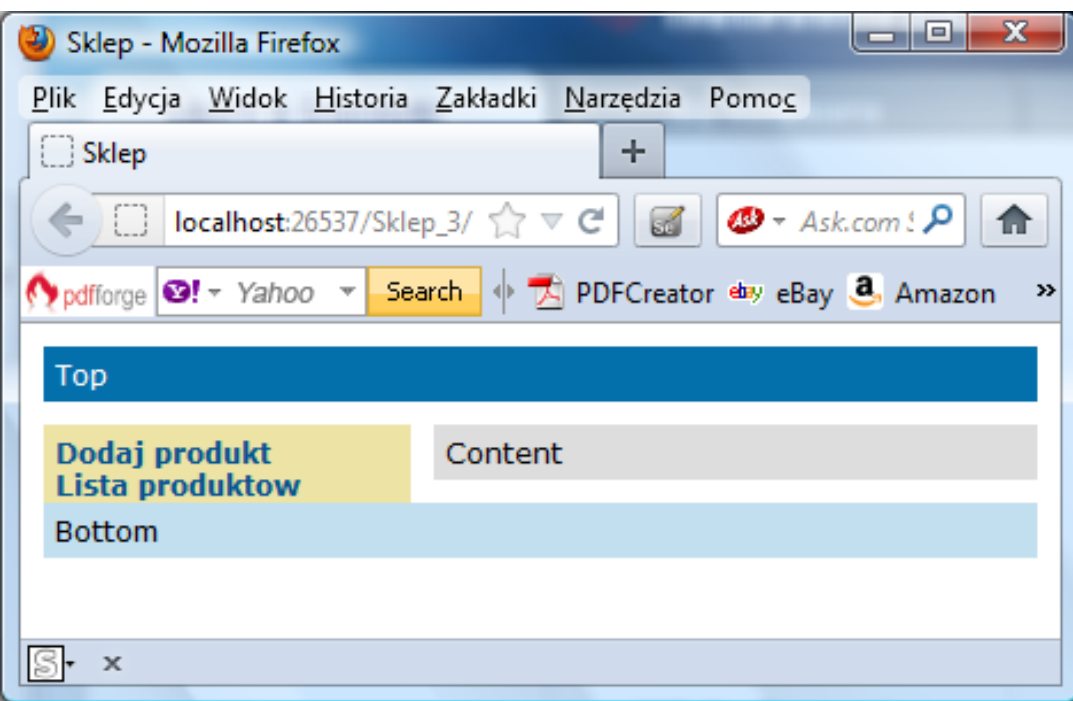
Name	fac...
Extension	xml
All Files	E...
File Size	400
Modification	201...
- Output View:** Shows a "Java DB Database Process" and "GlassFish Server 3+" running.

Deklaracja strony startowej index2.xhtml w pliku deskryptora web.xml

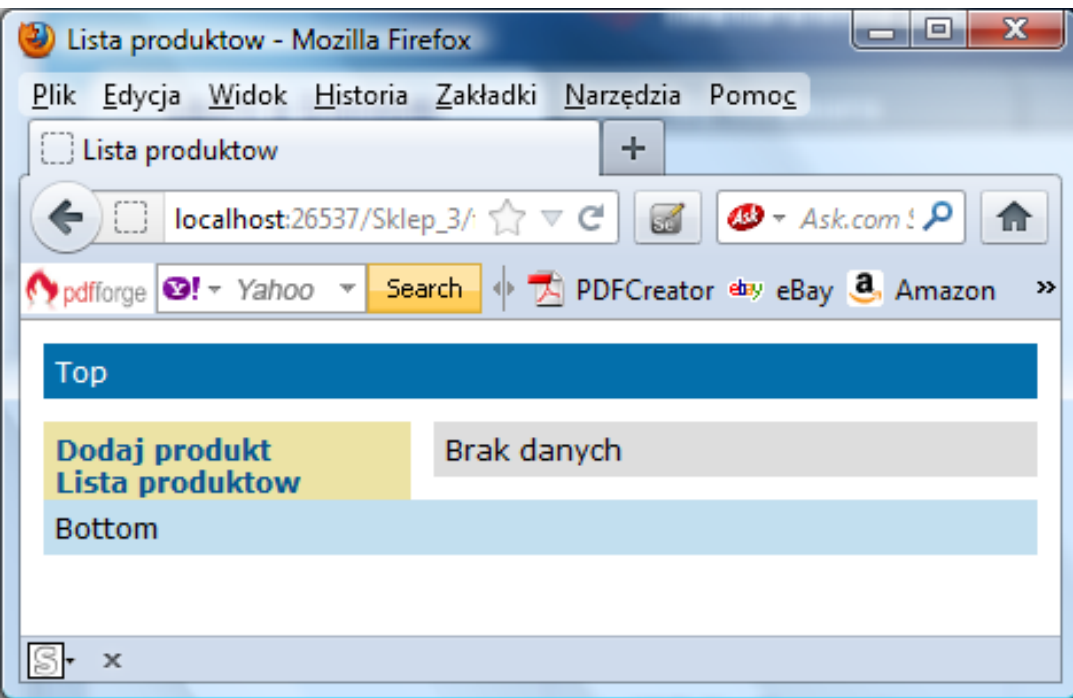
The screenshot displays an IDE window with the following components:

- Projects View:** Shows a project named 'Skep_3' with a 'Web Pages' folder containing 'WEB-INF' (with 'faces-config.xml' and 'web.xml'), 'jsf' (with 'dodaj_produk2.xhtml', 'lista_produkow.xhtml', and 'rezultat2.xhtml'), and 'resources' (with 'css' folder containing 'cssLayout.css', 'default.css', and 'jscrud.css'). Other files include 'index2.xhtml', 'template.xhtml', and 'Source Packages'.
- Code Editor:** Shows the 'web.xml' file with the following XML content:

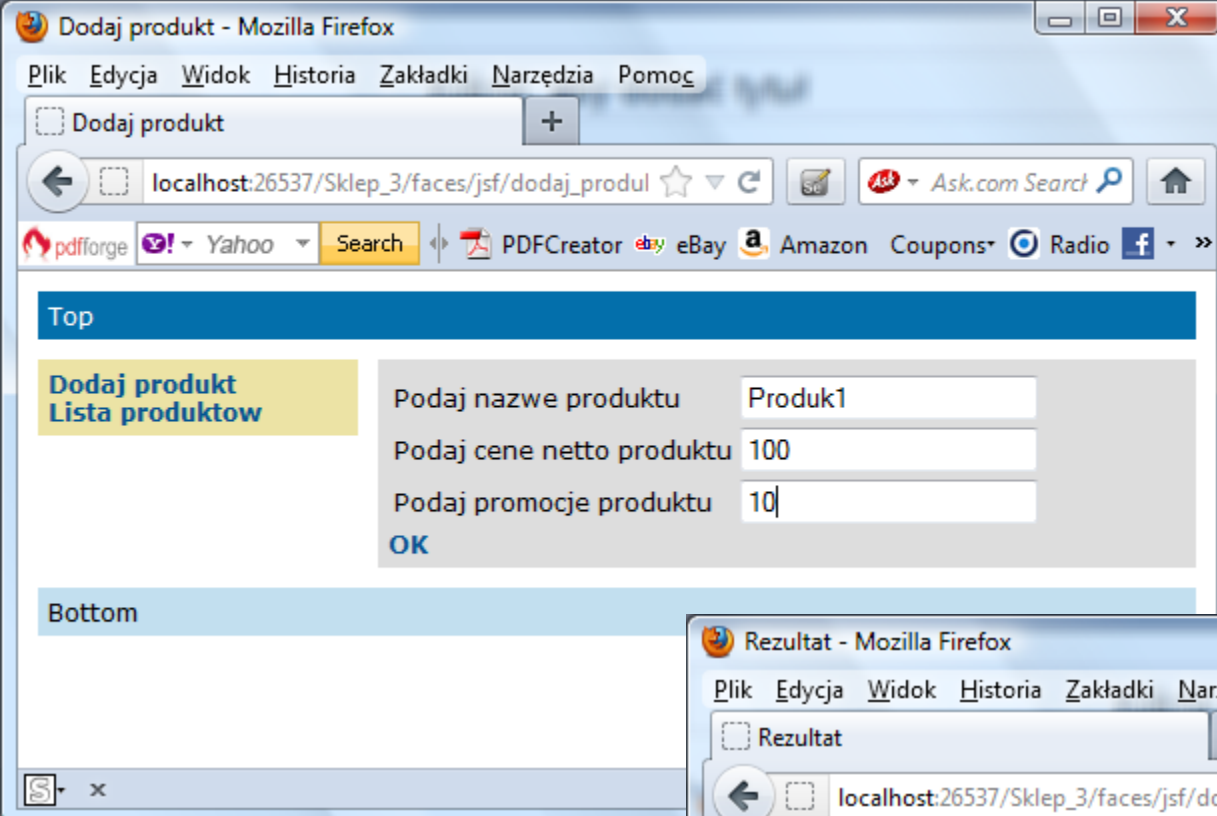
```
7 <servlet>
8     <servlet-name>Faces Servlet</servlet-name>
9     <servlet-class>javax.faces.webapp.FacesServlet</
10    <load-on-startup>1</load-on-startup>
11 </servlet>
12 <servlet-mapping>
13     <servlet-name>Faces Servlet</servlet-name>
14     <url-pattern>/faces/*</url-pattern>
15 </servlet-mapping>
16 <session-config>
17     <session-timeout>
18         30
19     </session-timeout>
20 </session-config>
21 <welcome-file-list>
22     <welcome-file>faces/index2.xhtml</welcome-file>
23 </welcome-file-list>
24 /web-app>
25
```
- Properties View:** Shows the 'web.xml' file properties: Name: web, Extension: xml, All Files: E..., File Size: 974, Modification: 201..., Servlet Spec: 3.0.
- Output View:** Shows 'HTTP Server Monitor' with a 'Sklep2 (run)' process running on 'GlassFish Server 3+'.



Uruchomienie aplikacji

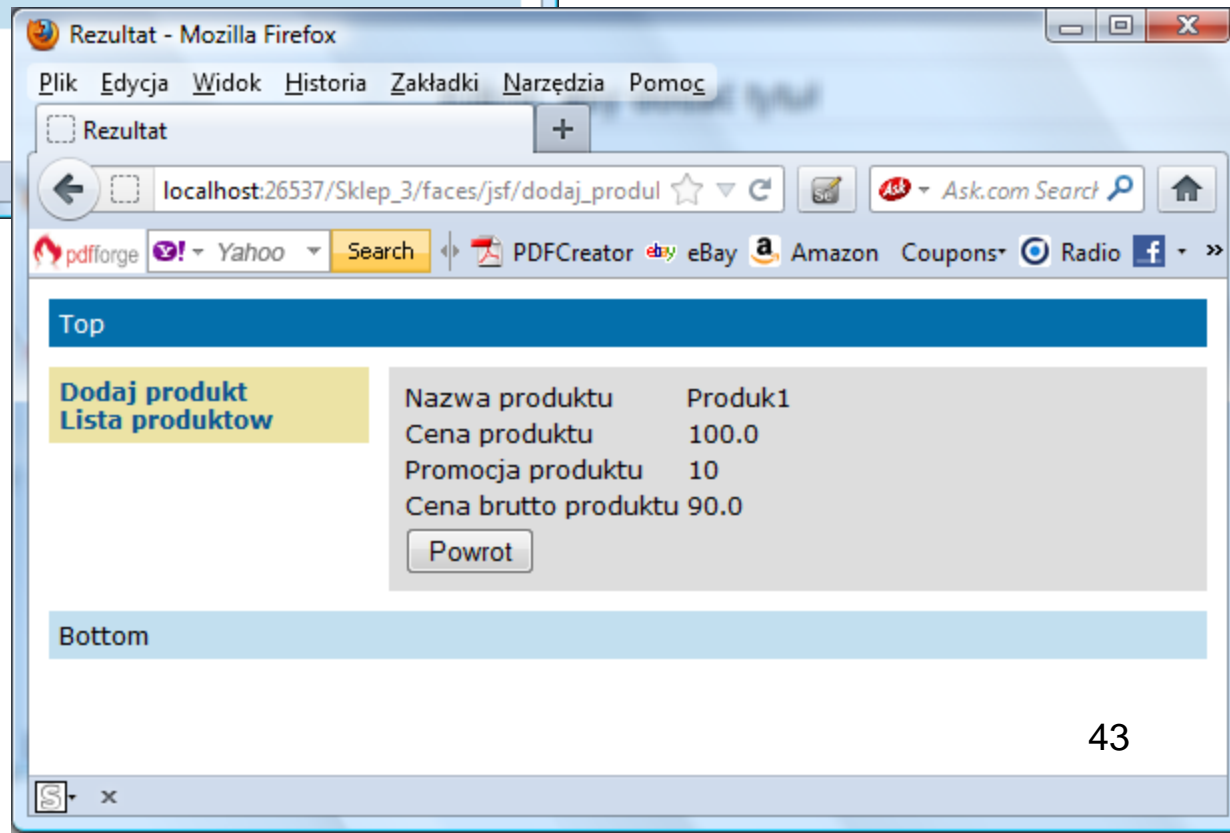


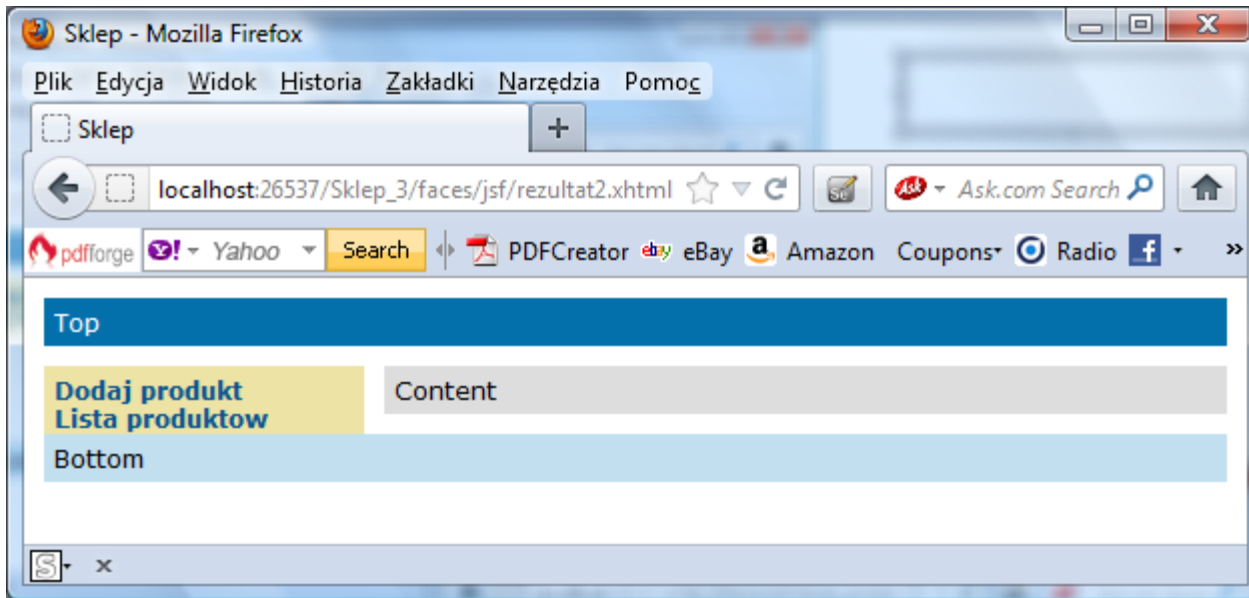
Widok po kliknięciu na
Lista produktów



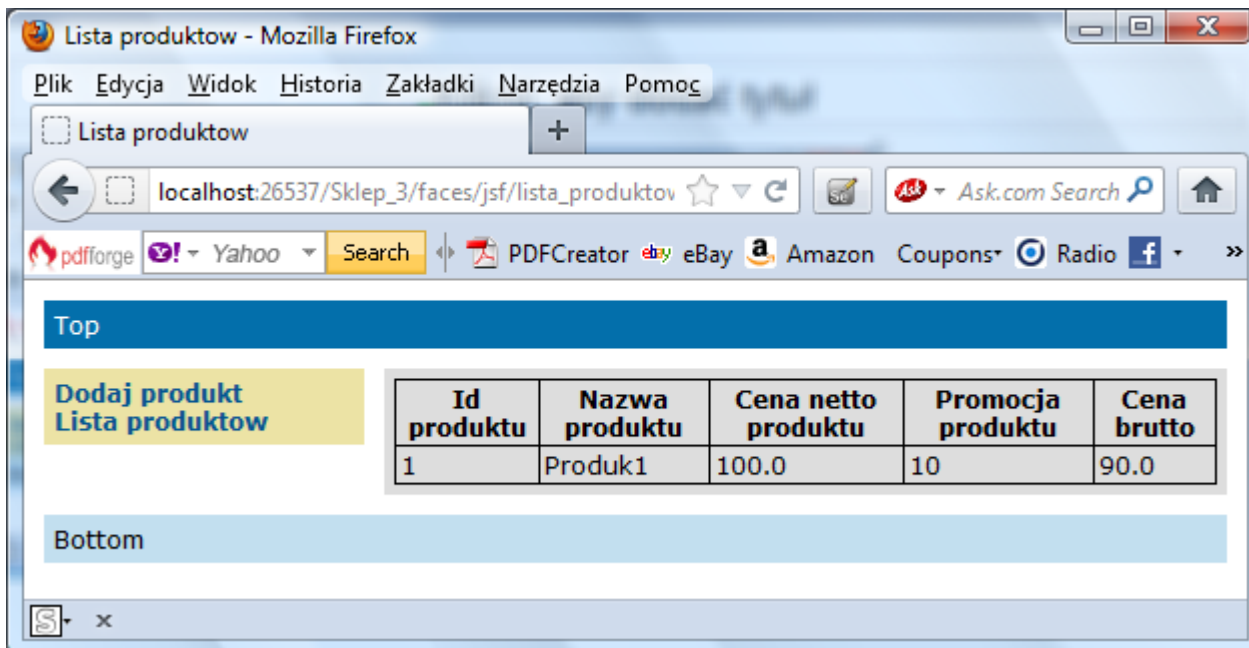
Widok po kliknięciu na
Dodaj produkt

Widok po kliknięciu na
Ok

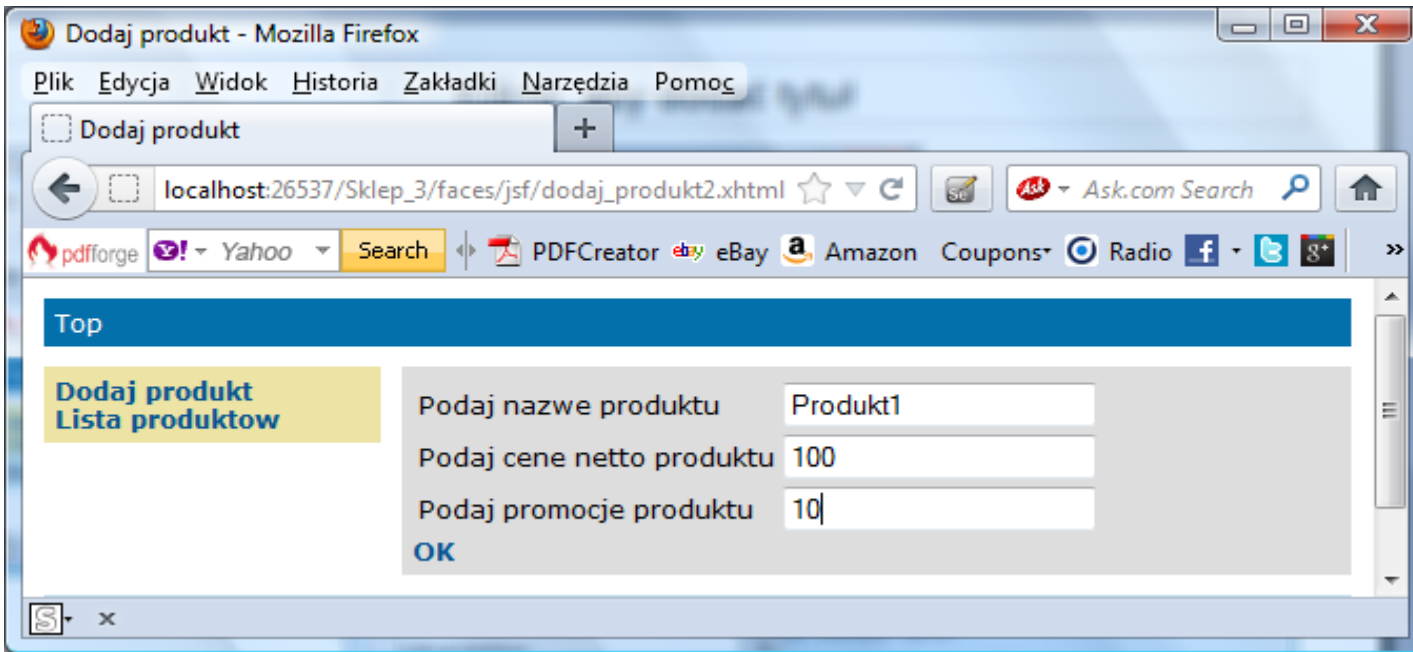




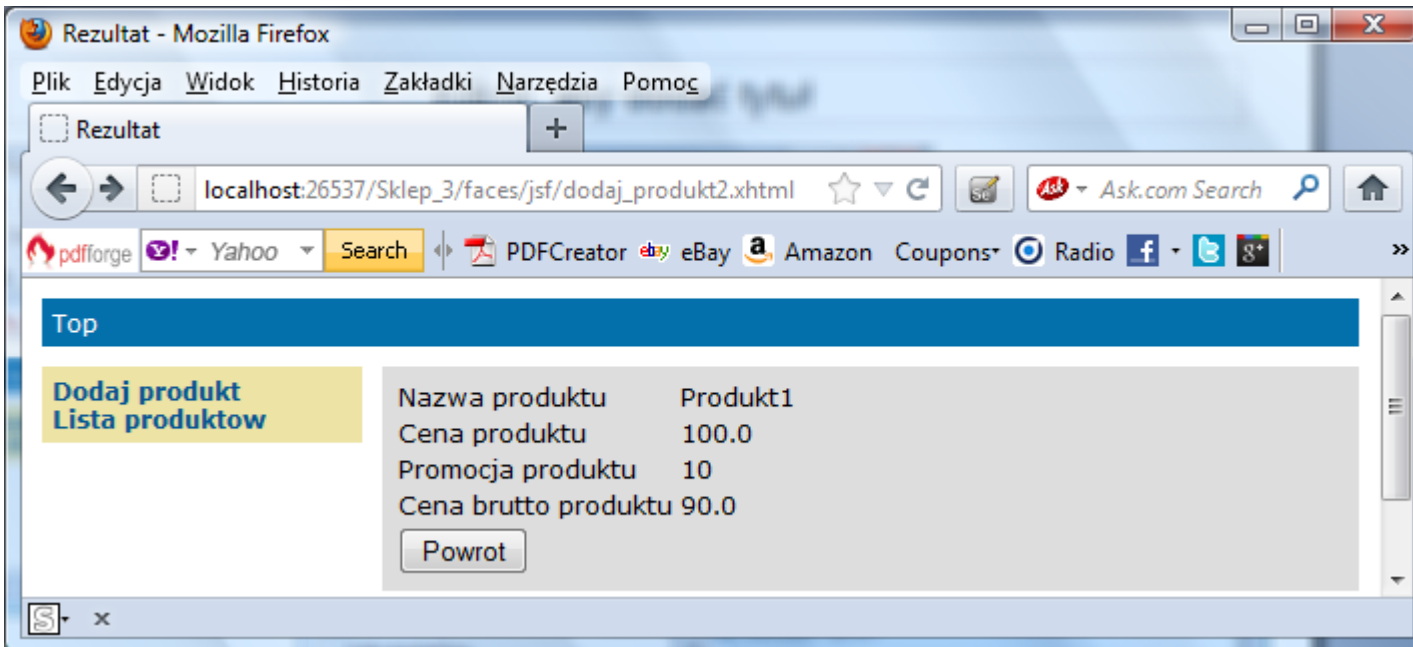
Widok po kliknięciu na
Powrot



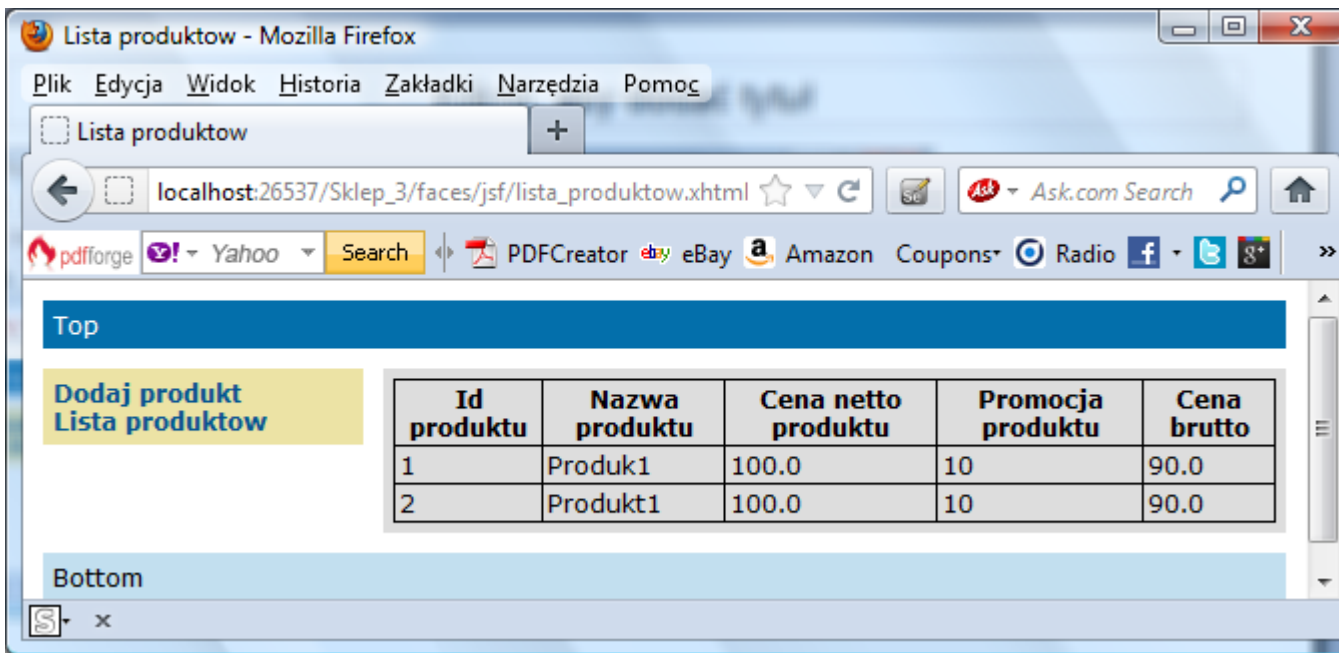
Widok po
kliknięciu na
Lista produktow



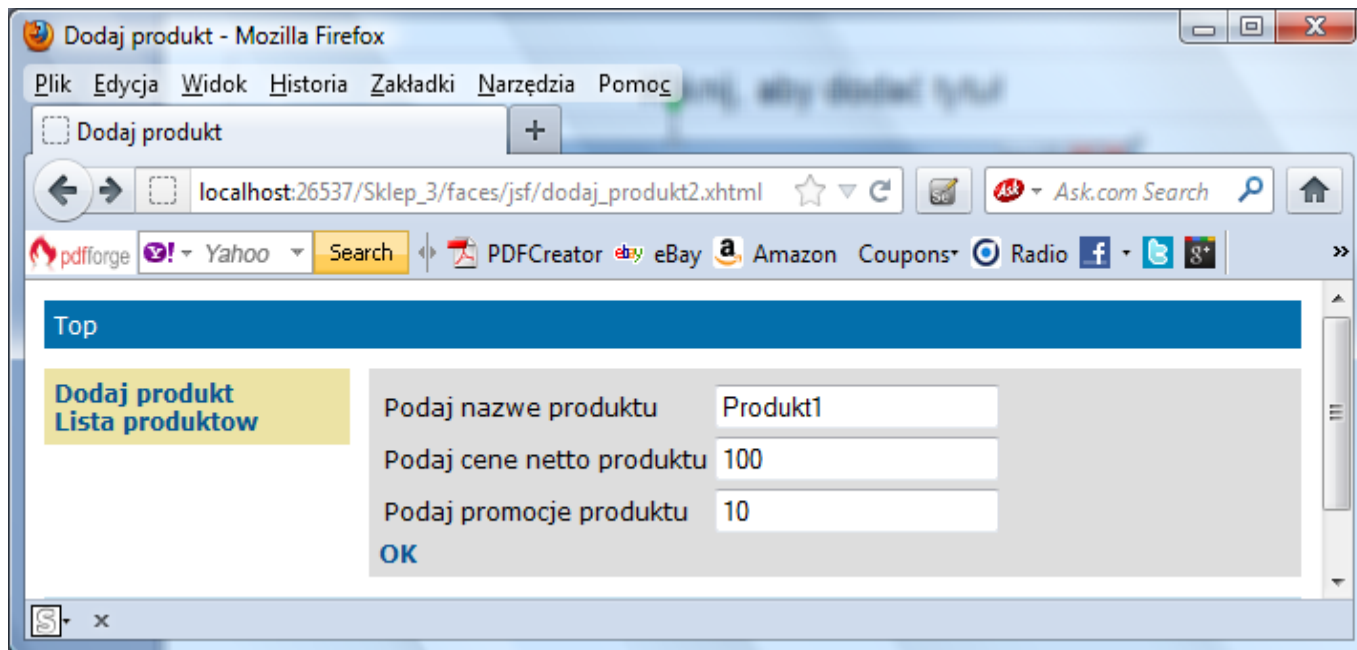
Widok po
kliknięciu na
Dodaj produkt



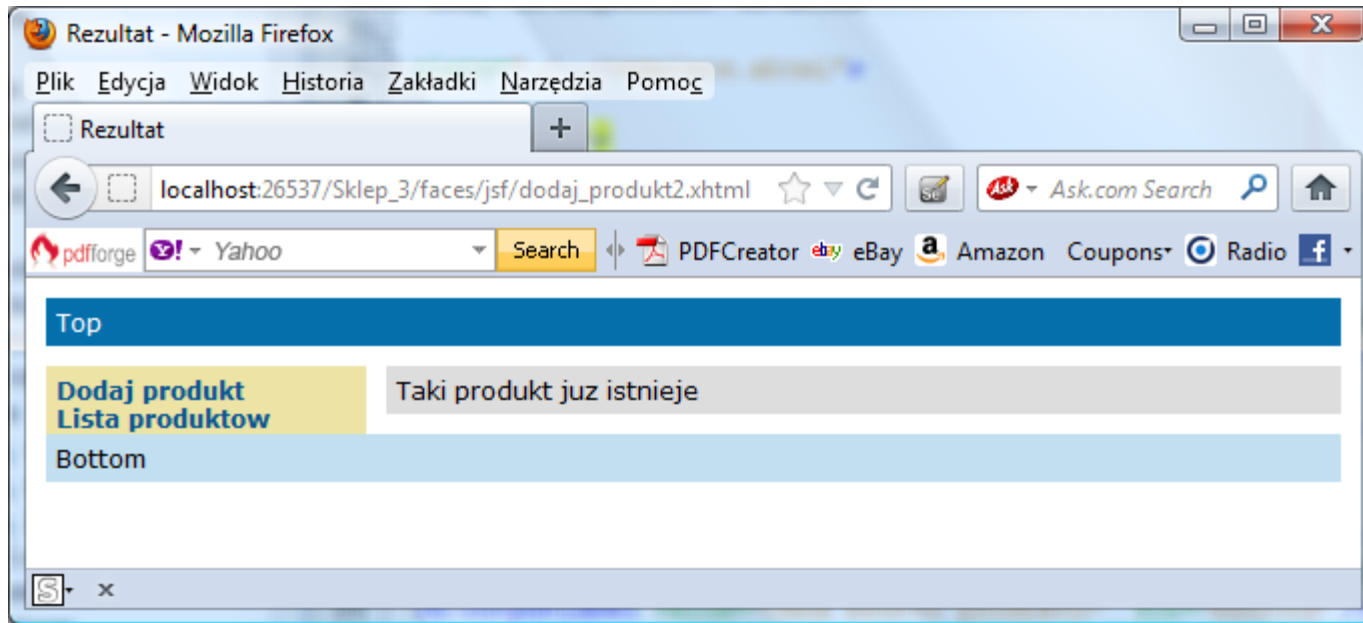
Widok po
kliknięciu na
Ok



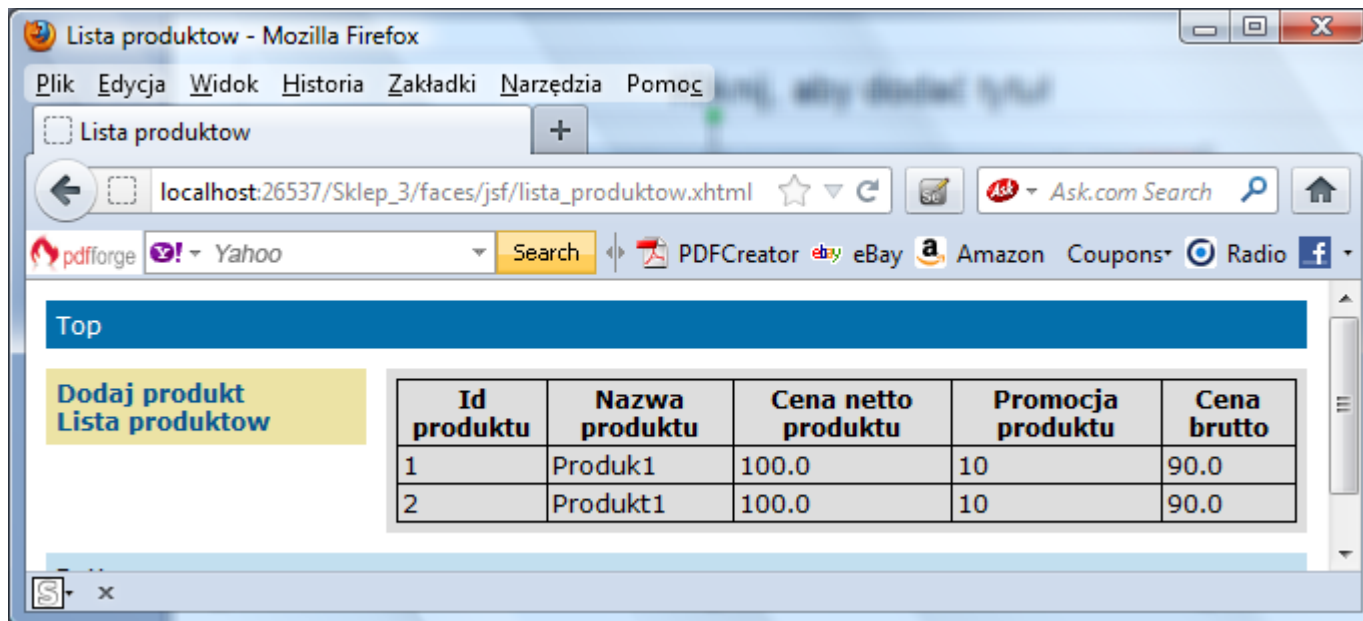
Widok po kliknięciu
na
Lista produktow



Widok po
kliknięciu na
Dodaj produkt



Widok po kliknięciu na
Ok



Widok po kliknięciu
na
Lista produktów

**Znaczniki typu f:
używane podczas obsługi walidacji,
konwersji, obsługi zdarzeń oraz
uzupełniają obsługę innych
znaczników**

podrozdział 10.3

1. Podstawowe znaczniki JSF – f:

Znaczniki do obsługi zdarzeń	Funkcja
f:actionListener	Dodaje słuchacza zdarzeń typu „action listener” do macierzystego komponentu
f:phaseListener	Dodaje słuchacza zdarzeń typu „PhaseListener” do strony
f:setPropertyActionListener	Rejestruje specjalnego słuchacza zdarzeń, którego jedynym celem jest przekazanie wartości do obiektu typu Managed Bean, gdy zawartość formularza jest przekazywana do serwera www.
f:valueChangeListener	Dodaje słuchacza zdarzeń typu „value-change listener” do macierzystego komponentu

Znaczniki reprezentujące elementy listy	Funkcja
f:selectItem	Reprezentuje jedną pozycję z listy
f:selectItems	Reprezentuje listę

2. Podstawowe znaczniki JSF – f: (cd)

Znaczniki do obsługi konwersji	Funkcja
f:converter	Dodaje dowolny konwerter do macierzystego komponentu
f:convertDateTime	Dodaje instancję konwertera typu DateTimeConverter do macierzystego komponentu
f:convertNumber	Dodaje instancję konwertera typu NumberConverter do macierzystego komponentu

Znaczniki aspektowe	Funkcja
f:facet	Dodaje zagnieżdżony komponent, który ma specjalne powiązanie ze znacznikiem zagnieżdżającym (macierzystym)
f:metadata	Rejestruje znacznik typu „facet” w macierzystym komponencie

3. Podstawowe znaczniki JSF – f(cd)

Znaczniki validatorów	Funkcje
f:validateDoubleRange	Dodaje validator typu DoubleRangeValidator do komponentu
f:validateLength	Dodaje validator typu LengthValidator do komponentu
f:validateLongRange	Dodaje validator typu LongRangeValidator do komponentu
f:validator	Dodaje validator zdefiniowany przez programistę do komponentu
f:validateRegEx	Dodaje validator typu RegExValidator do komponentu
f:validateBean	Zapewnia validację lokalnej warstości (atrybut value) za pomocą validatora typu BeanValidator
f:validateRequired	Wymusza obecność wartości (atrybut value) w komponencie

4. Podstawowe znaczniki JSF –f (cd)

Kategoria znacznika	Znaczniki różne	Funkcje
atrybut konfiguracji	f:attribute	Dodaje atrybuty konfiguracji do komponentu typu ojciec
lokalizacja	f:loadBundle	Specyfikuje komponent typu ResourceBundle , który jest reprezentowany w postaci obiektu typu Map
Parametr zastępowania	f:param	Zastępuje parametry w komponencie typu MessageFormat i dodaje parę nazwa-wartość do adresu URL
Ajax	f:ajax	Łączy zdarzenia Ajax z pojedynczym komponentem lub grupą komponentów opartych na rozmieszczeniu
zdarzenie	f:event	Pozwala na instalację słuchacza zdarzeń ComponentSystemEventListener w komponencie

Praktyczne porady dotyczące stosowania znaczników typu f:

Znaczniki	Gdzie używać znaczniki
Znaczniki do obsługi zdarzeń	Rejestrowanie słuchaczy zdarzeń w komponentach
Znaczniki do obsługi konwersji	Użycie standardowych konwerterów
f:facet	Użycie komponentów typu Data-Bound Table i Laying Out ze znacznikami h:panelGrid oraz h:panelGroup
f:loadBundle	Korzystanie z zasobów np. z plików typu properties
f:metadata	Używanie parametrów widoków do konfiguracji zakładek adresów URL
f:param	Wyświetlanie sformatowane wiadomości ze znacznikiem h:outputFormat
f:selectItem i f:selectItems	Używanie znaczników f:selectItem i f:selectItems do wyboru jednej lub wielu opcji
Znaczniki walidatorów	Używanie standardowych walidatorów
f:ajax	Dotyczy zagdania użycia technologii Ajax z technologią JavaServer Faces