Budowa aplikacji wielowarstwowych cd Dostęp do bazy danych w oparciu o technologię ORM (Object Relational Mapping)

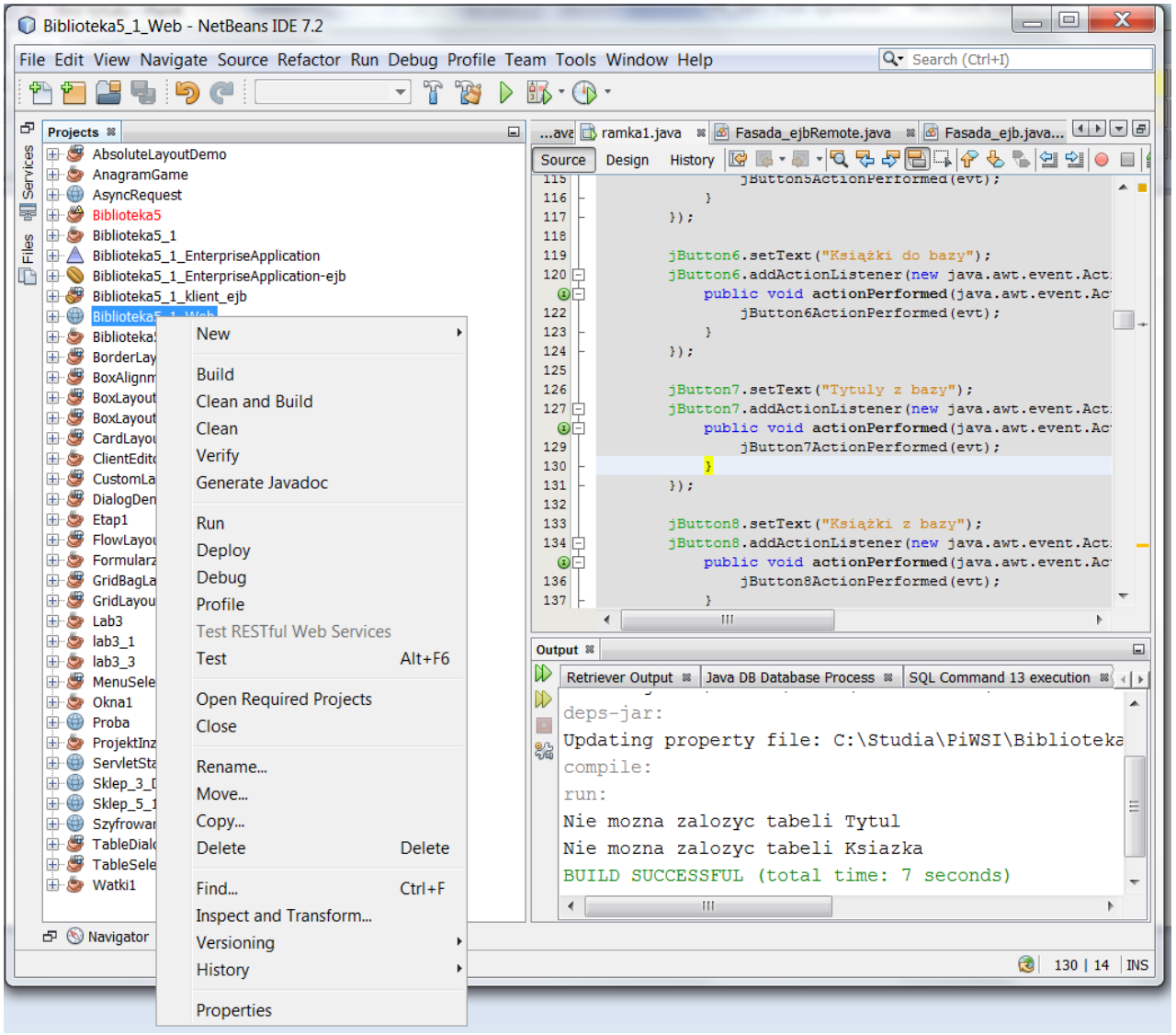
Laboratorium 5

Programowanie komponentowe

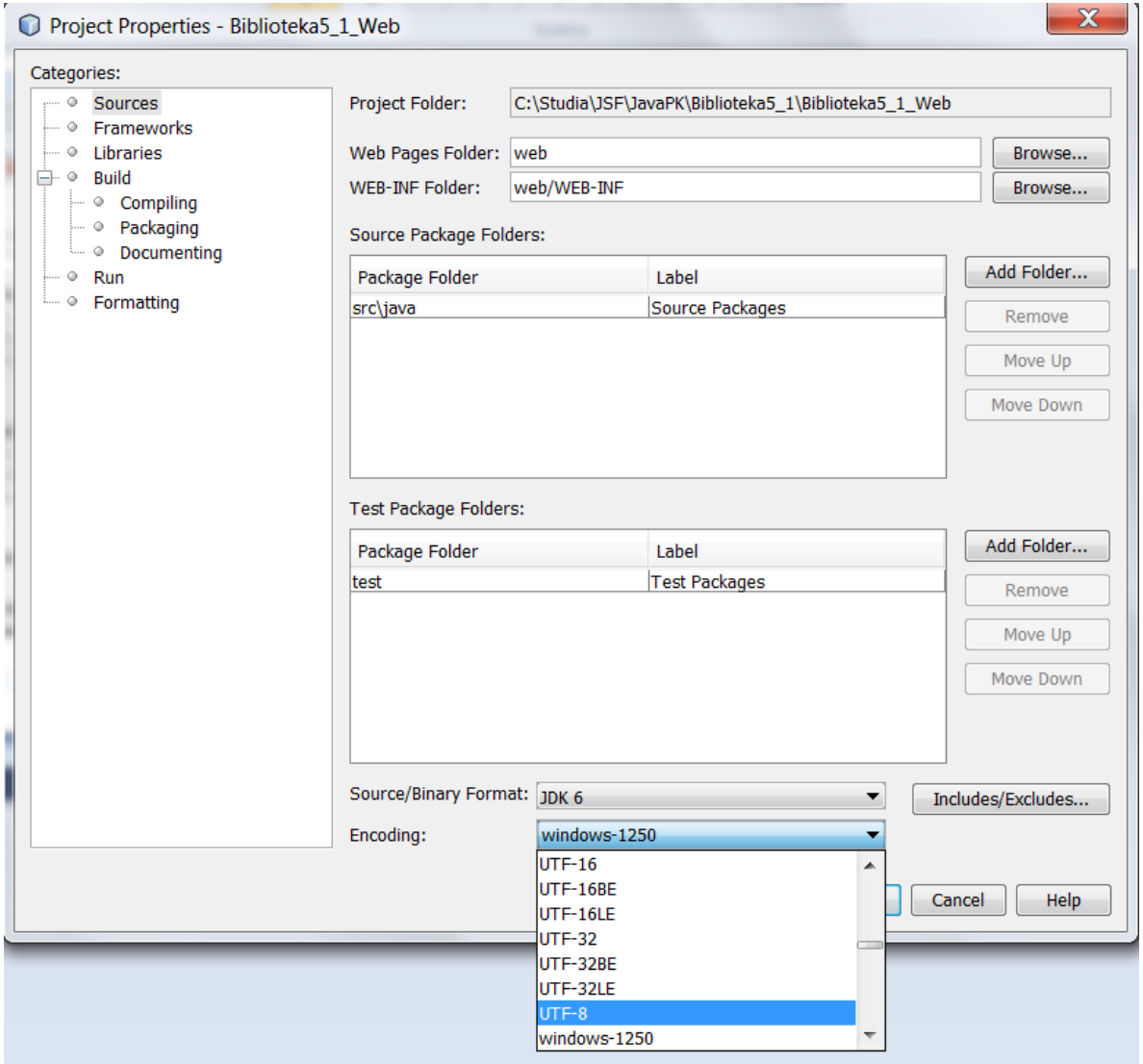
Zofia Kruczkiewicz

Konieczne ustawienie kodowania UTF-8 w projektach tworzonych w ramach laboratorium z Programowania komponentowego w Javie

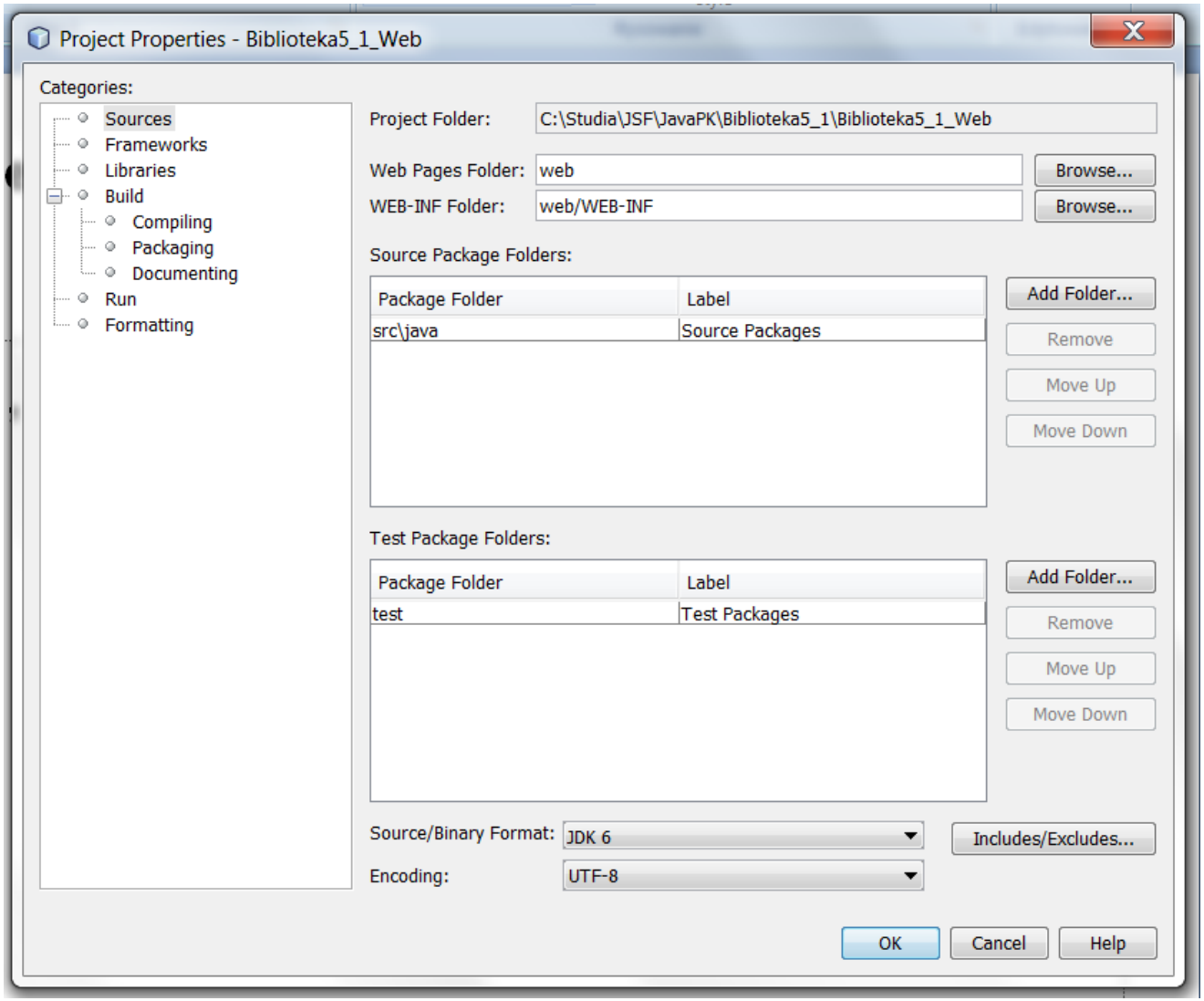
(1)Ustawienie kodowania typu UTF-8 w projektach Java SE oraz Java EE – w zakładce Projects należy prawym klawiszem kliknąć na nazwę projektu i wybrać z listy pozycję Properties



(2)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

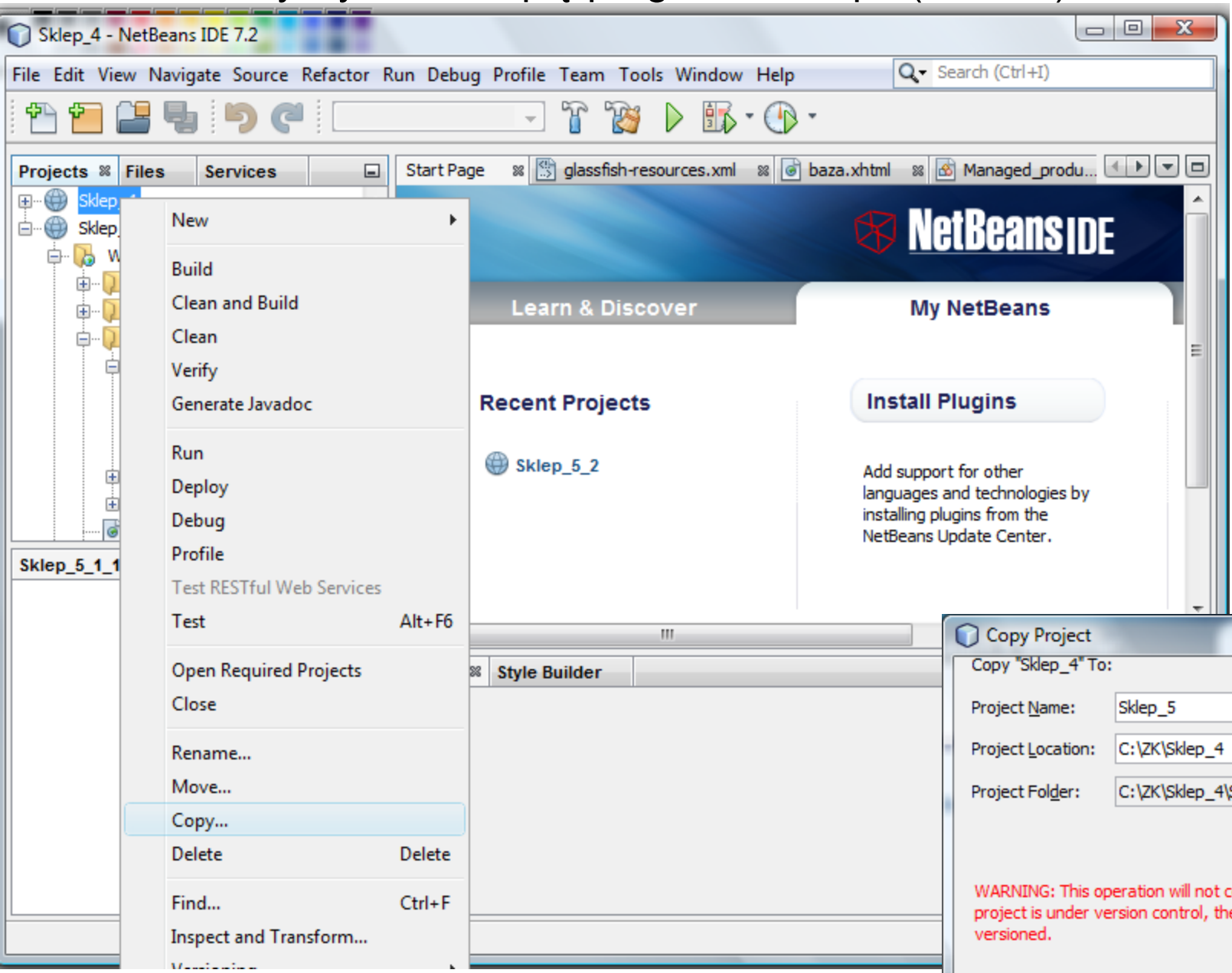


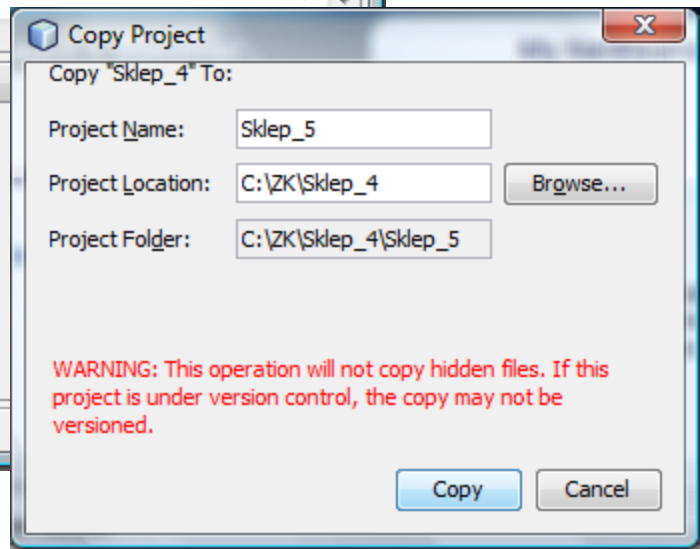
(3)Ustawienie kodowania typu UTF-8 w projektach Java SE oraz Java EE – należy zatwierdzić ustawienia klawiszem OK



1. Zastosowanie komponentu **h:selectOneMenu** (typu Drop Down List) na stronie lista_produktow.xhtml do wyboru promocji

1.1. Należy wykonać kopię programu Sklep4 (z lab. 3) o nazwie Sklep5





1.2. Dodanie do strony **dodaj_produkt2.xhtml** komponentu typu **h:selectOneMenu** – do wprowadzania promocji

```
<h:outputLabel value="#{bundle['jsf.dodaj_produkt2.podaj_promocja']}"
                              for="promocja" />
<h:selectOneMenu id="promocja"
                              value="#{managed_produkt.promocja}"
                              title="#{bundle['jsf.dodaj_produkt2.podaj_promocja']}"
                              required="true"
    requiredMessage="#{bundle['jsf.dodaj_produkt2.podaj_promocja_blad']}">
        <f:selectItems value="#{managed_produkt.itemsAvailableSelectOne}"/>
</h:selectOneMenu>
```

1.3. Dodanie metody w klasie **Fasada_warstwy_biznesowej** do generowania modelu listy rozwijanej, zawierającej wykaz promocji do wyboru z listy

```
public ArrayList<Integer> findAll() {
      ArrayList<Integer> pom = new ArrayList();
      pom.add(new Integer(0));
      pom.add(new Integer(10));
      pom.add(new Integer(20));
      pom.add(new Integer(50));
      return pom;
   }
```

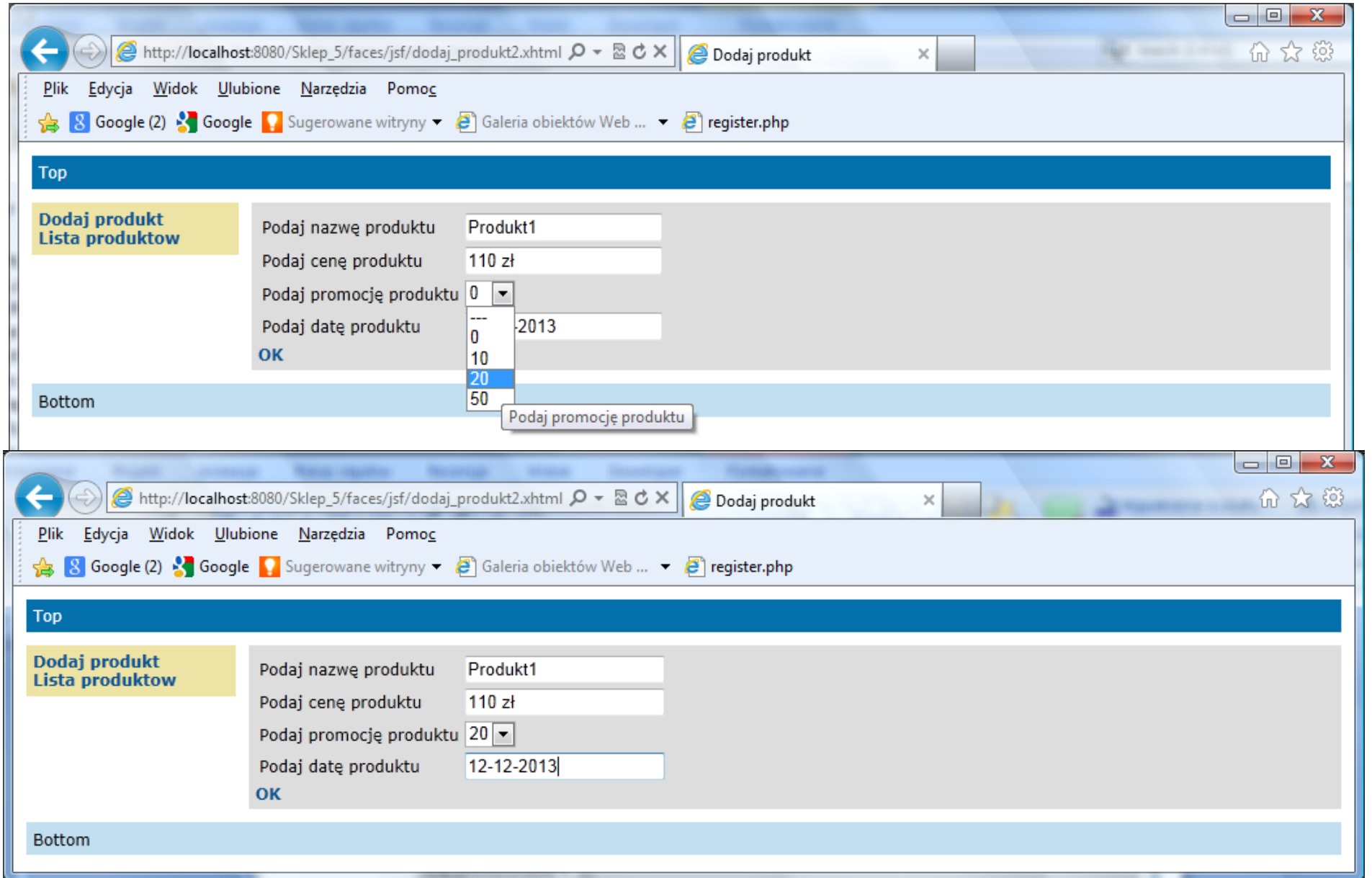
1.4. Dodanie do klasy **Managed_produkt** metody obsługującej wybór z listy – właściwość managed_produkt.itemsAvailableSelectOne dla znacznika f:selectItems value

```
public SelectItem[] getItemsAvailableSelectOne() {
        return JsfUtil.getSelectItems(fasada.findAll(), true);
    }
```

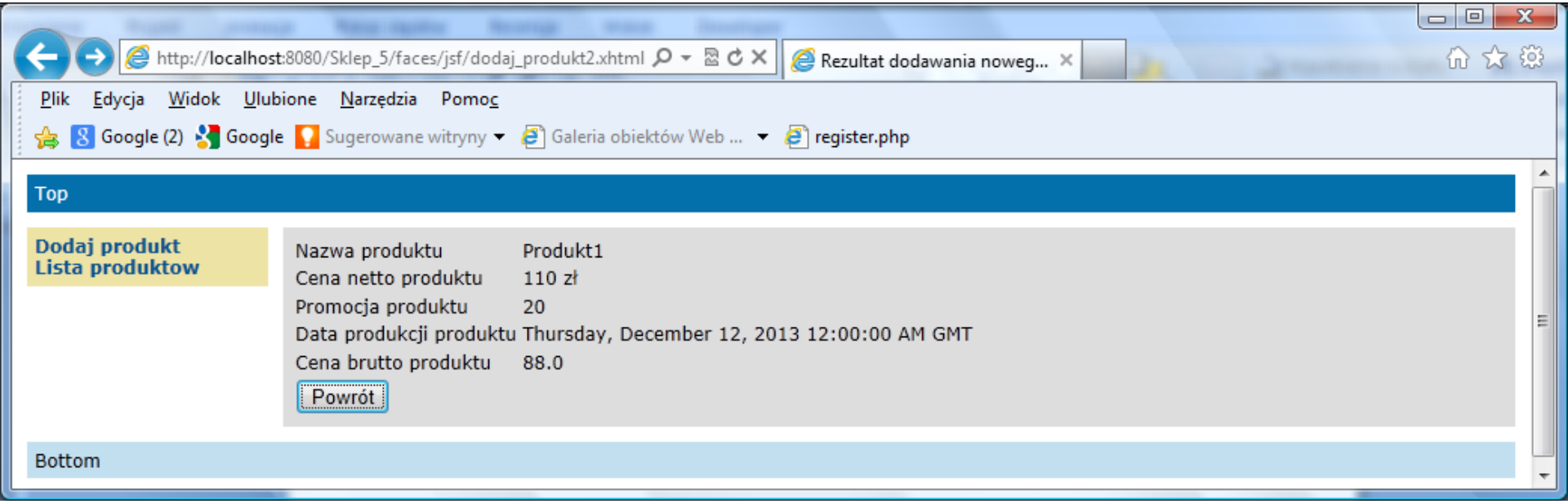
1.5. Należy dodać pakiet **jsf.util, zdefiniować nową klasę JsfUtil** i wstawić następujący kod metody getSelectItems do tej klasy

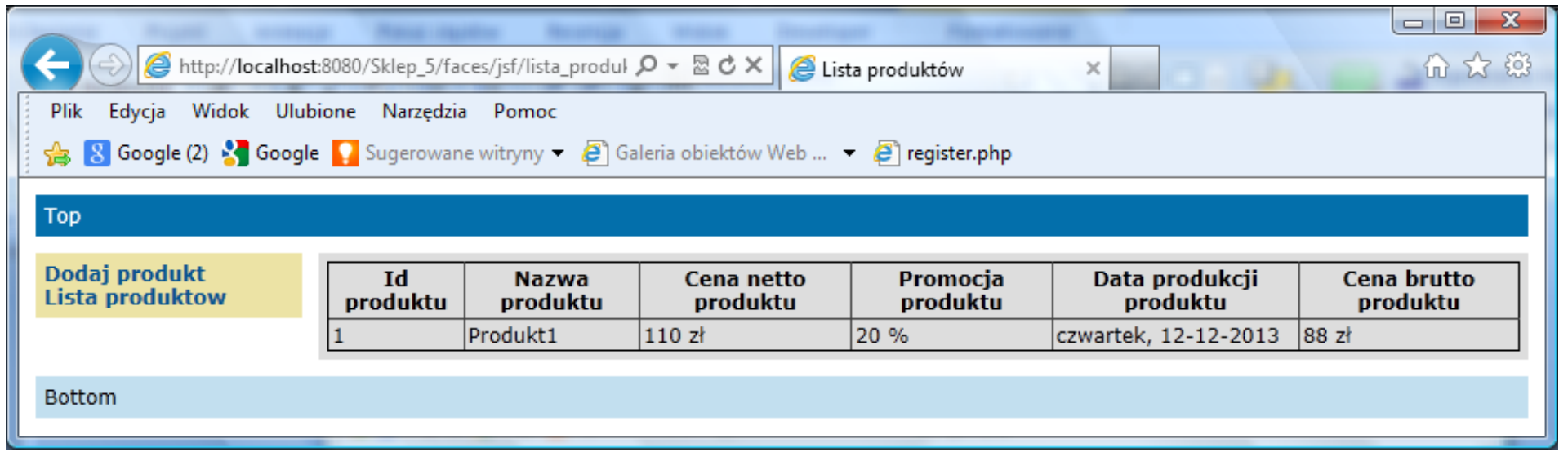
```
package jsf.util;

import java.util.List;
import javax.faces.model.SelectItem;

public class JsfUtil {
    public static SelectItem[] getSelectItems(List<?> entities, boolean selectOne) {
        int size = selectOne ? entities.size() + 1 : entities.size();
        SelectItem[] items = new SelectItem[size];
        int i = 0;
        if (selectOne) {
            items[0] = new SelectItem("", "---");
            i++;  }
        for (Object x : entities) {
            items[i++] = new SelectItem(x, x.toString());
        }
        return items;
    }
}
```

1.6. Prezentacja działania programu (1)



1.7. Prezentacja programu (2)





2. Dodanie stronicowania zawartości komponentu **dataTable** na stronie **lista_produktow.xhtml.**

Kod dotyczący stronicowania oparty na kodzie wygenerowanym podczas tworzenia stron JSF na podstawie klas typu Entity (pakiet jsf.util) - przykład 3 przedstawiony na wykładzie:
http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/javapk/PK_2.pdf
Wykład jest oparty na materiałach umieszczonych na stronach:
Java EE & Java Web Learning Trail.
Tutorial Java EE 6

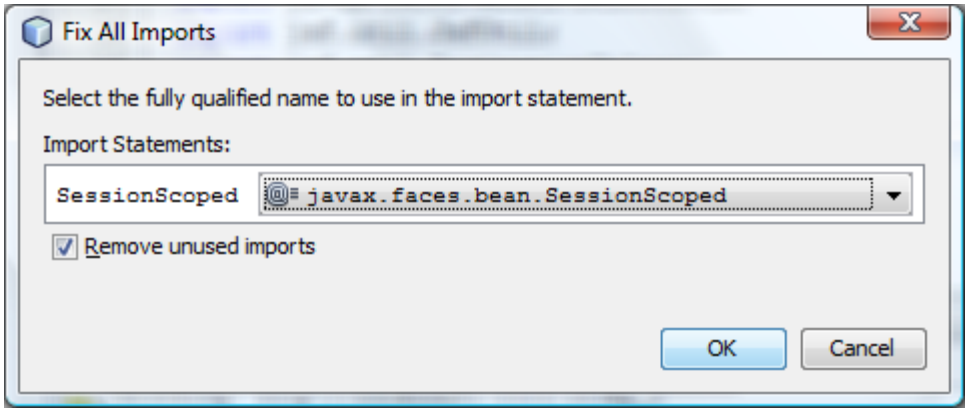
2.1. Dodanie stronicowania stron – fragment pliku lista_produktow.xhtml z dodanym kodem znaczników odpowiedzialnym za stronicowanie (zaznaczony kolorem czerwonym – przed znacznikiem dataTable) – pierwszy znacznik commnadLink służy do stronicowania „wstecz” a drugi do stronicowania „do przodu”

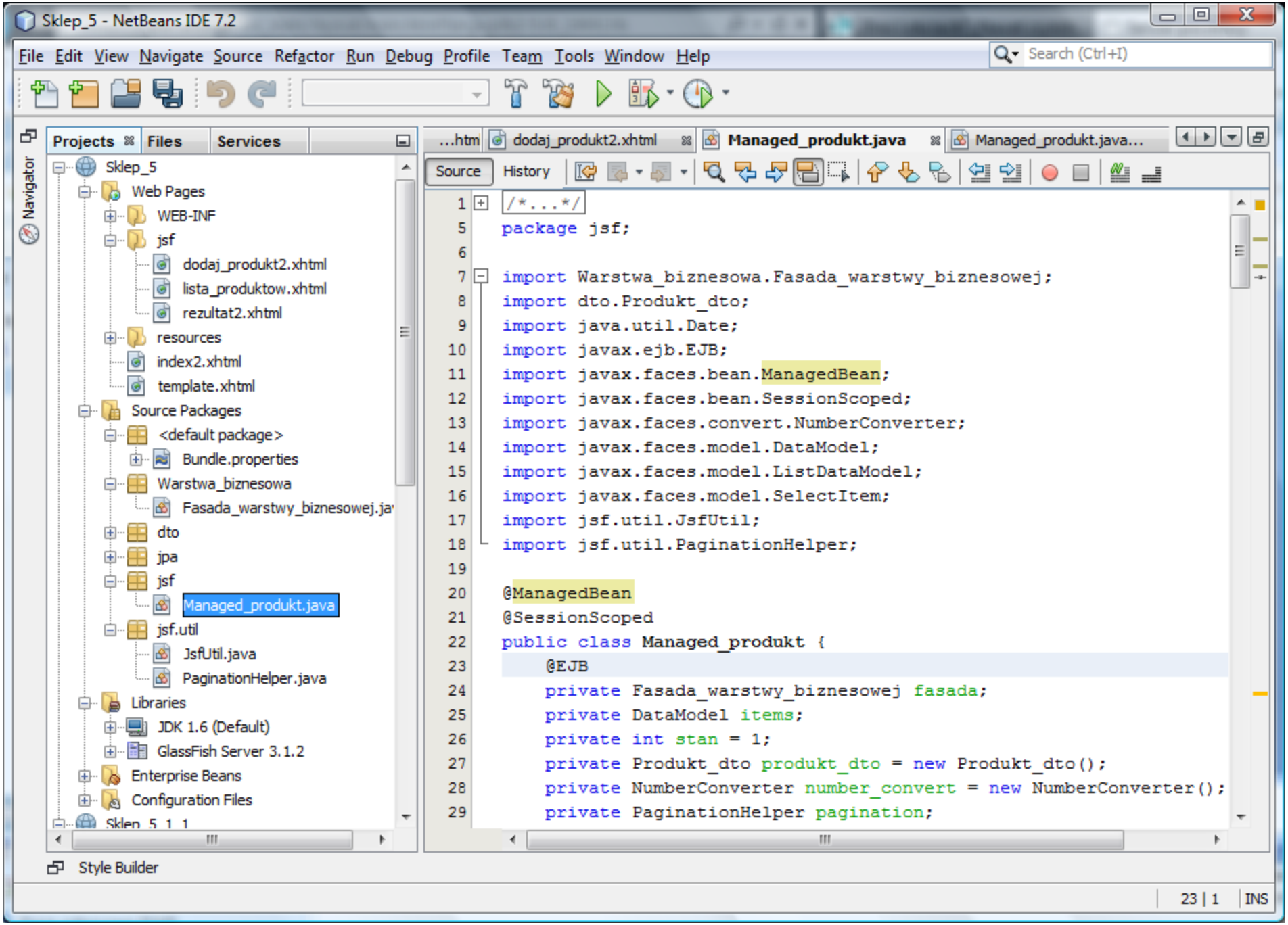
```
<ui:define name="content">
    <h:form styleClass="jsfcrud_list_form">
      <h:panelGroup id="messagePanel" layout="block">
          <h:messages errorStyle="color: red" infoStyle="color: green" layout="table"/>
     </h:panelGroup>
     <h:outputText escape="false" value="#{bundle['jsf.lista_produktow.pusta']}"
                              rendered="#{managed_produkt.pagination.itemsCount == 0}"/>
     <h:panelGroup rendered="#{managed_produkt.pagination.itemsCount > 0}">
        <h:outputText value="#{managed_produkt.pagination.pageFirstItem + 1}
                                  ..#{managed_produkt.pagination.pageLastItem + 1}
                                  /#{managed_produkt.pagination.itemsCount}"/> 
        <h:commandLink
            action="#{managed_produkt.previous}"
            value="#{bundle['jsf.lista_produktow.poprzedni']} #{managed_produkt.pagination.pageSize}"
            rendered="#{managed_produkt.pagination.hasPreviousPage}"/> 
       <h:commandLink
            action="#{managed_produkt.next}"
            value="#{bundle['jsf.lista_produktow.nastepny']} #{managed_produkt.pagination.pageSize}"
            rendered="#{managed_produkt.pagination.hasNextPage}"/> 
```

2.2. Zmiana czasu życia obiektu typu **Managed_produkt** do czasu trwania sesji za pomocą adnotacji **@SessionScoped**

```
package jsf;

import Warstwa_biznesowa.Fasada_warstwy_biznesowej;
import dto.Produkt_dto;
import java.util.Date;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.convert.NumberConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
import jsf.util.JsfUtil;
import jsf.util.PaginationHelper;

@ManagedBean
@SessionScoped
public class Managed_produkt {
```

Podczas wprowadzania importu brakujących klas za pomocą **Fix Import** (wybrana pozycja po kliknięciu prawym klawiszem na powierzchnię okna edytora)




Sklep_5 - NetBeans IDE 7.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
Services
Navigator
Sklep_5
Web Pages
WEB-INF
jsf
dodaj_produkt2.xhtml
lista_produktow.xhtml
rezultat2.xhtml
resources
index2.xhtml
template.xhtml
Source Packages
<default package>
Bundle.properties
Warstwa_biznesowa
Fasada_warstwy_biznesowej.ja
dto
jpa
jsf
Managed_produkt.java
jsf.util
JsfUtil.java
PaginationHelper.java
Libraries
JDK 1.6 (Default)
GlassFish Server 3.1.2
Enterprise Beans
Configuration Files
Sklep_5_1_1
...htm
dodaj_produkt2.xhtml
Managed_produkt.java
Managed_produkt.java...
Source
History
1 /*...*/
5 package jsf;
6
7 import Warstwa_biznesowa.Fasada_warstwy_biznesowej;
8 import dto.Produkt_dto;
9 import java.util.Date;
10 import javax.ejb.EJB;
11 import javax.faces.bean.ManagedBean;
12 import javax.faces.bean.SessionScoped;
13 import javax.faces.convert.NumberConverter;
14 import javax.faces.model.DataModel;
15 import javax.faces.model.ListDataModel;
16 import javax.faces.model.SelectItem;
17 import jsf.util.JsfUtil;
18 import jsf.util.PaginationHelper;
19
20 @ManagedBean
21 @SessionScoped
22 public class Managed_produkt {
23 @EJB
24 private Fasada_warstwy_biznesowej fasada;
25 private DataModel items;
26 private int stan = 1;
27 private Produkt_dto produkt_dto = new Produkt_dto();
28 private NumberConverter number_convert = new NumberConverter();
29 private PaginationHelper pagination;
Style Builder
23 | 1
INS

2.3. Kod, który należy dodać i zmodyfikować w klasie **Managed_produkt**

```
public DataModel getItems() {
        if (items == null) {
            items = getPagination().createPageDataModel();  }
         return items;
    }
 public String dodaj_produkt() {
        fasada.utworz_produkt(produkt_dto);
        dane_produktu();
        recreateModel();
        getPagination().nextPage();
        return "rezultat2";
    }
public void dane_produktu() {
        stan = 1;
        produkt_dto = fasada.dane_produktu();
        if (produkt_dto == null) {
            produkt_dto = new Produkt_dto();
            stan = 0;  }
    }
```

```
public PaginationHelper getPagination() {
    if (pagination == null) {
        pagination = new PaginationHelper(3) {

            @Override
            public int getItemsCount() {
                return getFasada().count();
            }

            @Override
            public DataModel createPageDataModel() {
                int[] range = {getPageFirstItem(), getPageLastItem() +1};
                return new ListDataModel(getFasada().findRange(range));    }
        };
    }
    return pagination; }
```

Ustalenie wartości atrybutu **pageSize**, w obiekcie klasy dziedziczącej po klasie **PaginationHelper** (definicję klasy podano na kolejnych slajdach) z przedefiniowanymi metodami **getItemsCount** oraz **createPageDataModel**

```
private void recreateModel() {
        items = null;
    }

public String next() {
        getPagination().nextPage();
        recreateModel();
        return "lista_produktow";
    }

 public String previous() {
        getPagination().previousPage();
        recreateModel();
        return "lista_produktow";
    }
```

Metoda usuwająca zawartość bieżącej strony – umożliwia metodzie **getItems** utworzenie zawartości wybranej strony

Metoda obsługująca wybór następnej strony

Metoda obsługująca wybór poprzedniej strony

2.4. Do pakietu **jsf.util** należy dodać klasę **PaginationHelper**, zawierającą następujący kod:

```
package jsf.util;
import javax.faces.model.DataModel;

public abstract class PaginationHelper {
    private int pageSize;
    private int page;

    public PaginationHelper(int pageSize) {
        this.pageSize = pageSize;
    }
    public abstract int getItemsCount();
    public abstract DataModel createPageDataModel();

public int getPageFirstItem() {
        return page * pageSize;  }
```

pageSize - rozmiar strony – czyli liczba wierszy komponentu, **dataTable**
page – numer strony, startowa wartość równa 0

pageSize*page – liczba pozycji przypisanych do wszystkich stron, a jednocześnie numer danej wyświetlanej jako pierwsza pozycja na ostatniej stronie

```
public int getPageLastItem() {
    int i = getPageFirstItem() + pageSize - 1;
    int count = getItemsCount() - 1;
    if (i > count) {
        i = count; }
    if (i < 0) {i = 0; }
    return i;
}

public boolean isHasNextPage() {
    return (page + 1) * pageSize + 1 <= getItemsCount();
}

public void nextPage() {
    if (isHasNextPage()) {
        page++;
    }
}
```

Indeks ostatniej pozycji, czyli numer ostatniej danej, wyświetlany na ostatniej stronie

Sprawdzenie, czyli liczba danych wymaga utworzenie następnej strony

Numer ostatniej strony

```
public boolean isHasPreviousPage() {
        return page > 0;
    }

    public void previousPage() {
        if (isHasPreviousPage()) {
            page--;
        }
    }

    public int getPageSize() {
        return pageSize;
    }

public void setPage() {
        this.page = getItemsCount()/pageSize;
    }     //aktualizacja w dniu 26.05.2013
}
```

Sprawdzenie, czy istnieje poprzednia strona (pierwsza strona ma numer 0)

Pobranie numeru poprzedniej strony

Pobranie rozmiaru strony

Aktualizacja liczby stron

2.5. Kod, który należy dodać do klasy **Fasada_warstwy_biznesowej**

```
public int count() {
        return produkty.size();
    }
public ArrayList<Produkt_dto> findRange(int[] range) {
        ArrayList<Produkt_dto> pom = new ArrayList();
        for (int i = range[0]; i < range[1]; i++) {
            pom.add(produkt_transfer(getProdukty().get(i)));
        }
        return pom;
    }
```

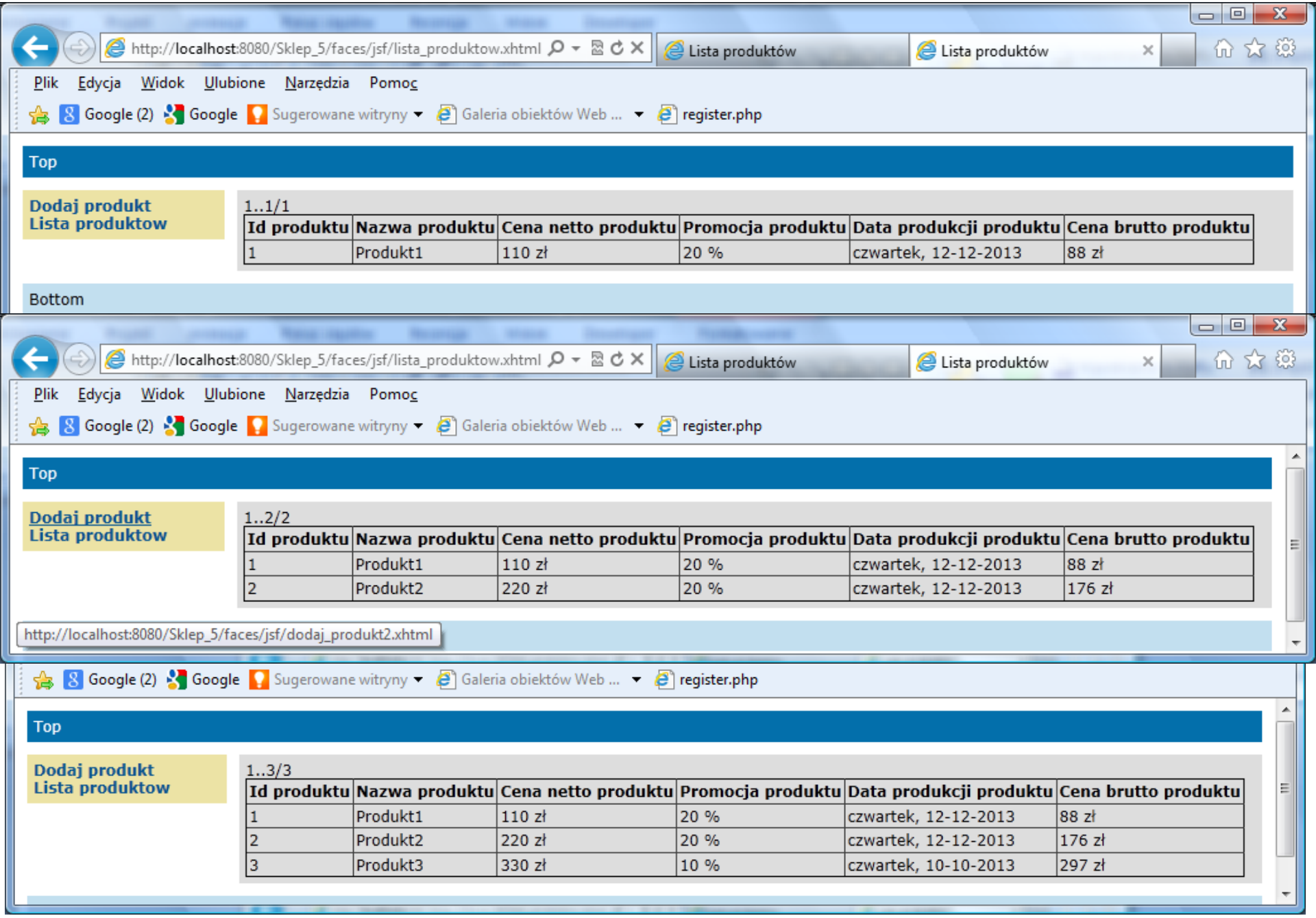
Pobranie liczby danych

Pobranie podzbioru danych potrzebnych do wyświetlenia na stronie za pomocą metody **findRange**. Tablica **range** zawiera dwa elementy: pierwszy zawiera numer pierwszego elementu, drugi element zawiera numer ostatniego elementu z kolekcji **produkty**, które wyznaczają podzbiór danych pobieranych do wyświetlenia na stronie. Pobrane elementy z kolekcji produktu są przekształcone na obiekty transferowe typu Produkt_dto.

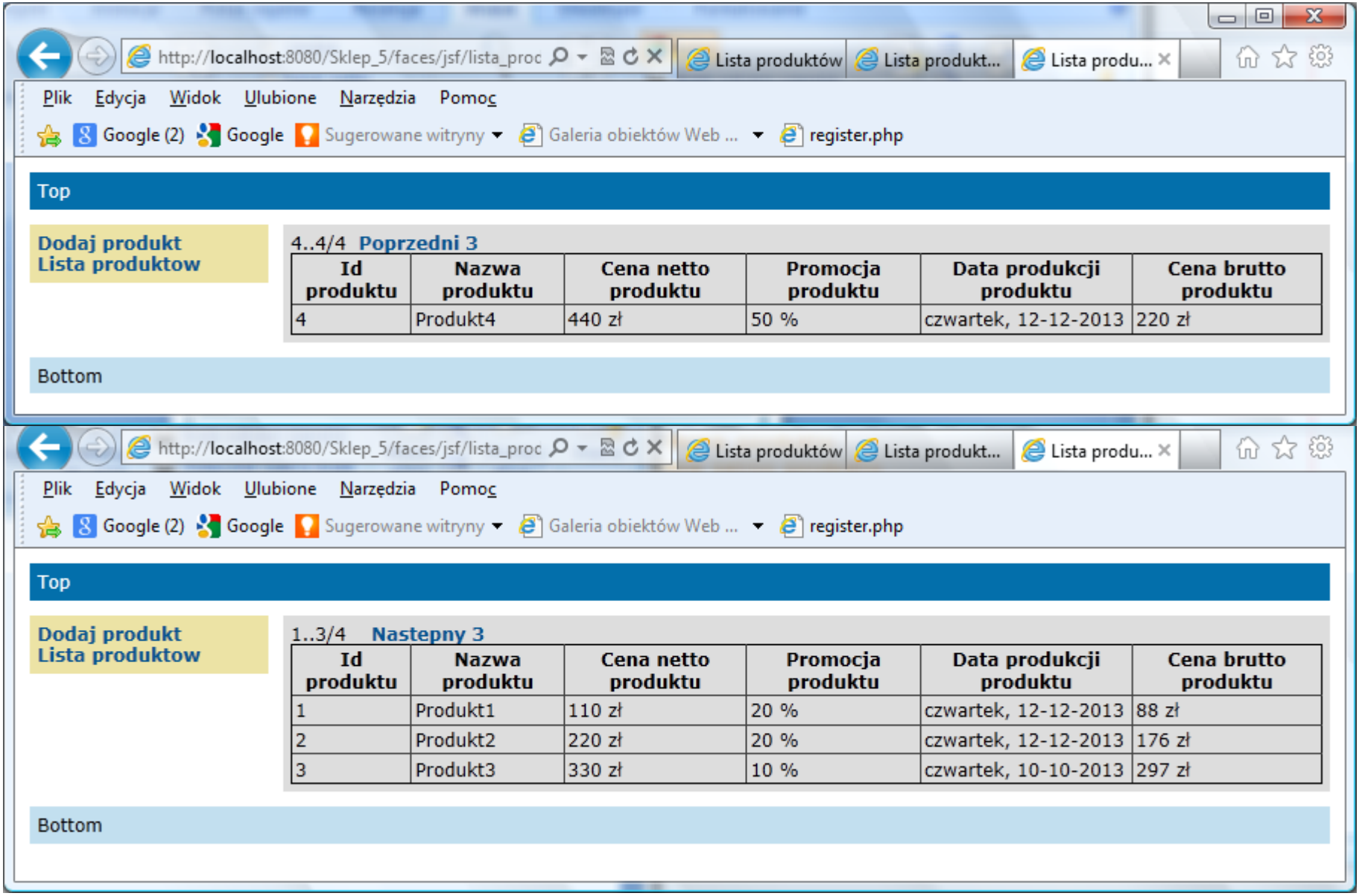
2.6. Uzupełnienie zawartości pliku **Bundle.properties**

```
jsf.lista_produktow.poprzedni=Poprzedni
jsf.lista_produktow.nastepny=Nastepny
```

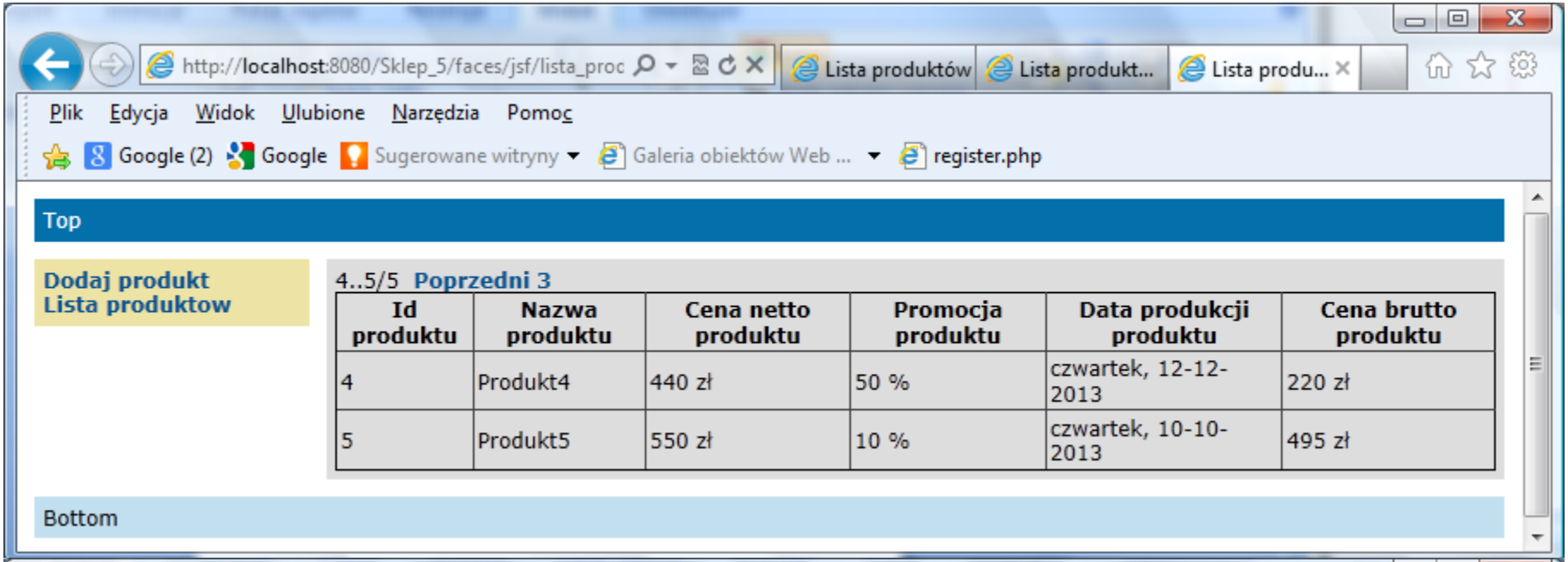
2.7. Prezentacja stronicowania (1)

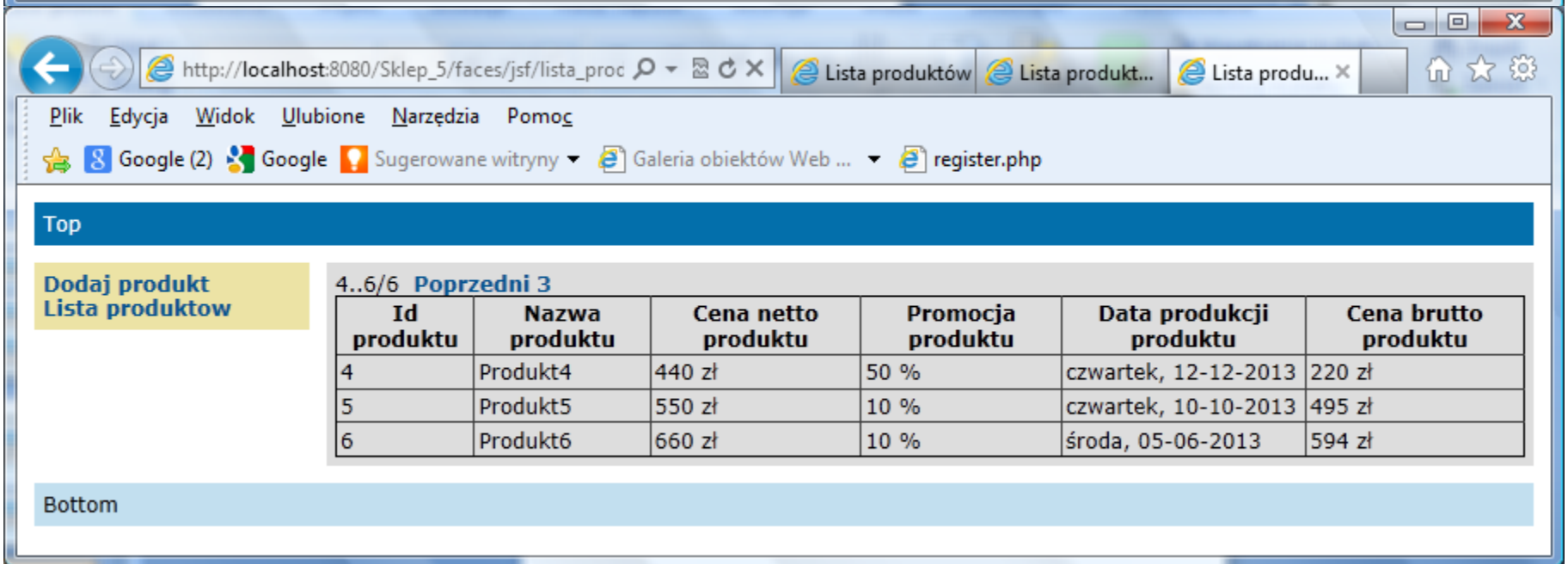


2.8. Prezentacja stronicowania (2)

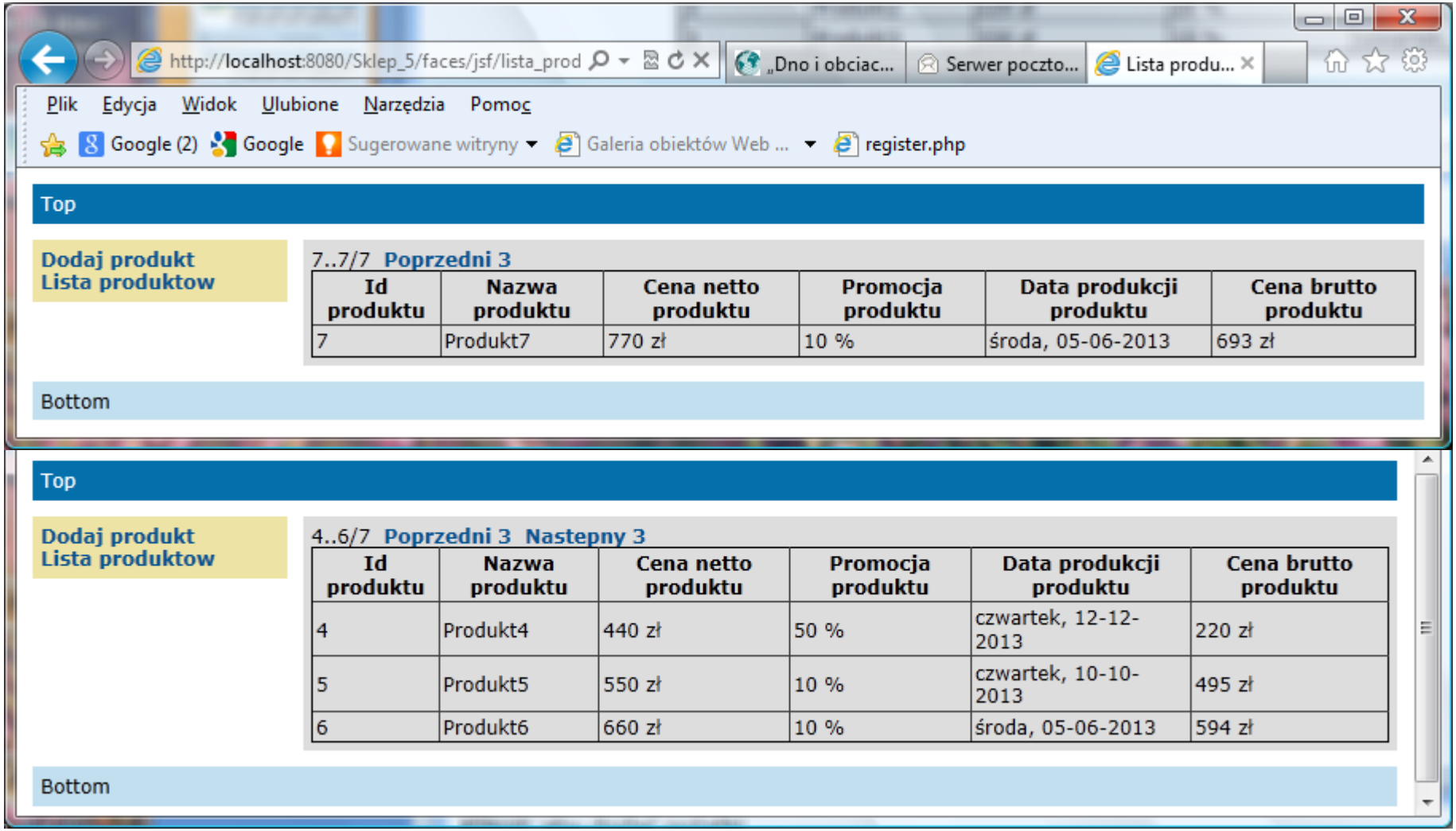


2.9. Prezentacja stronicowania (3)

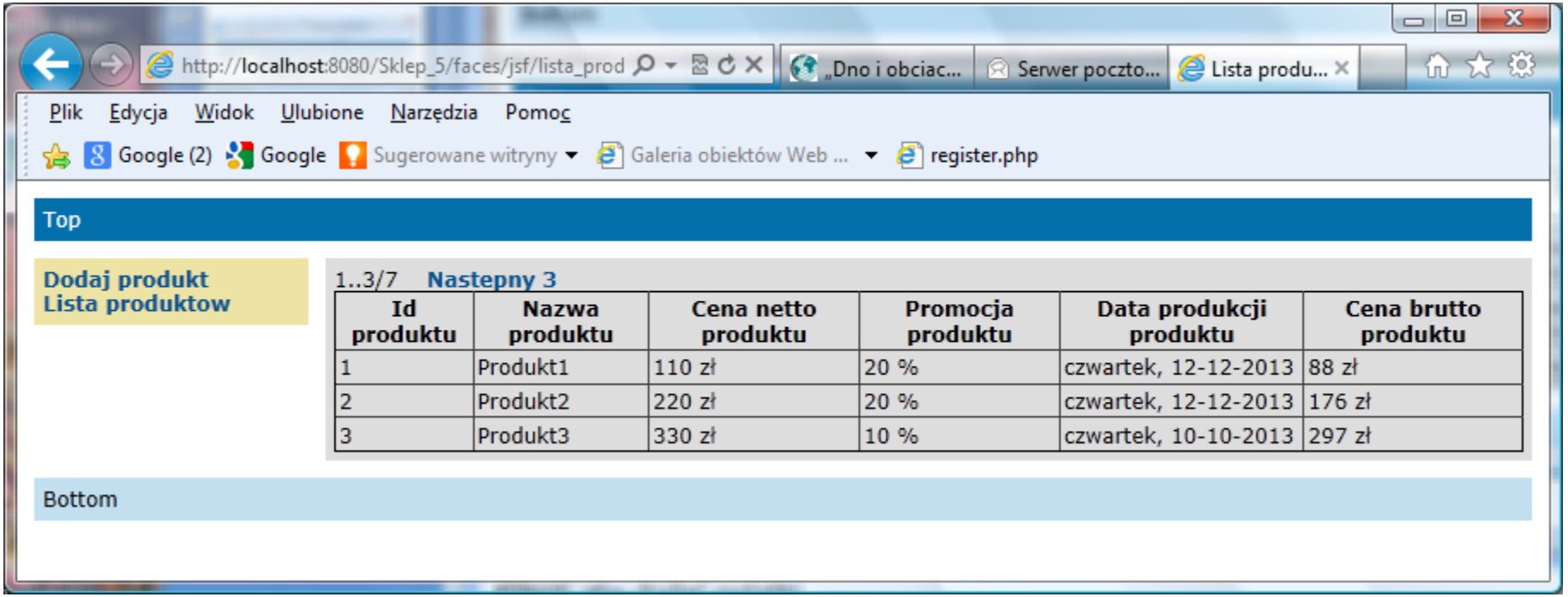




2.10. Prezentacja stronicowania (4)



2.11. Prezentacja stronicowania (5)



3. Wprowadzenie warunkowego renderowania strony rezultat2.xhtml w celu zapewnienia wielożywalności tej strony

3.1. Dodanie do komponentu **dataTable** na stronie **lista_produktow.xhtml** kolumny z przyciskiem obsługującym operację **Rezultat**

```
<h:column>
<f:facet name="header">
    <h:outputText value=" "/>
 </f:facet>
 <h:commandLink action="#{managed_produkt.prepareView}"
                value="#{bundle['jsf.lista_produktow.rezultat']}"/>
 </h:column>
```

Podany znacznik **h:column** należy umieścić jako ostatni w komponencie **dataTable**

3.2. Wprowadzenie renderowanych przycisków na stronie **rezultat2.xhtml** – w przypadku wywołania tej strony ze strony **dodaj_produkt2.xhtml** zmienna **powrot** jest równa 1 i następuje powrót do strony **index2.xhtml**, a w przypadku wywołania ze strony **lista_produktow.xhtml** zmienna **powrot** jest równa 0 – wtedy ze strony powraca się z powrotem do strony **lista_produkt.xhtml**

```
<h:commandButton id="powrot1"
                 value="#{bundle['jsf.rezultat2.akcja']}"
                 action="/faces/index2"
                 rendered="#{managed_produkt.powrot!=0}"/>
 <h:commandButton id="powrot2"
                  value="#{bundle['jsf.rezultat2.akcja']}"
                  action="#{managed_produkt.powrot}"
                  rendered="#{managed_produkt.powrot==0}"/>
```

3.3. Dodanie do klasy **Managed_produkt** obsługi przycisku umożliwiającego przejście do strony **rezulat2.xhtml** ze strony **lista_produktow.xhtml** (metoda **prepareView**) i powrót ze strony **rezultat2.xhtml** do strony **lista_produktow.xhtml** (metoda **powrot**) oraz właściwości **powrot**

```
private int powrot = 1;
 public int getPowrot() {
        return powrot;
    }

public String prepareView() {
        produkt_dto = (Produkt_dto) getItems().getRowData();
        powrot = 0;
        stan=1;
        return "rezultat2";
    }
public String powrot() {
        powrot = 1;
        produkt_dto = new Produkt_dto();
        return "lista_produktow";
    }
```

Właściwość **powrot** pozwala renderować przyciski umożliwiające powrót ze strony **rezultat2.xhtml** do strony index2.xhtml (**powrot** = 1) lub **lista_produktow.xhtml** (**powrot** = 0).

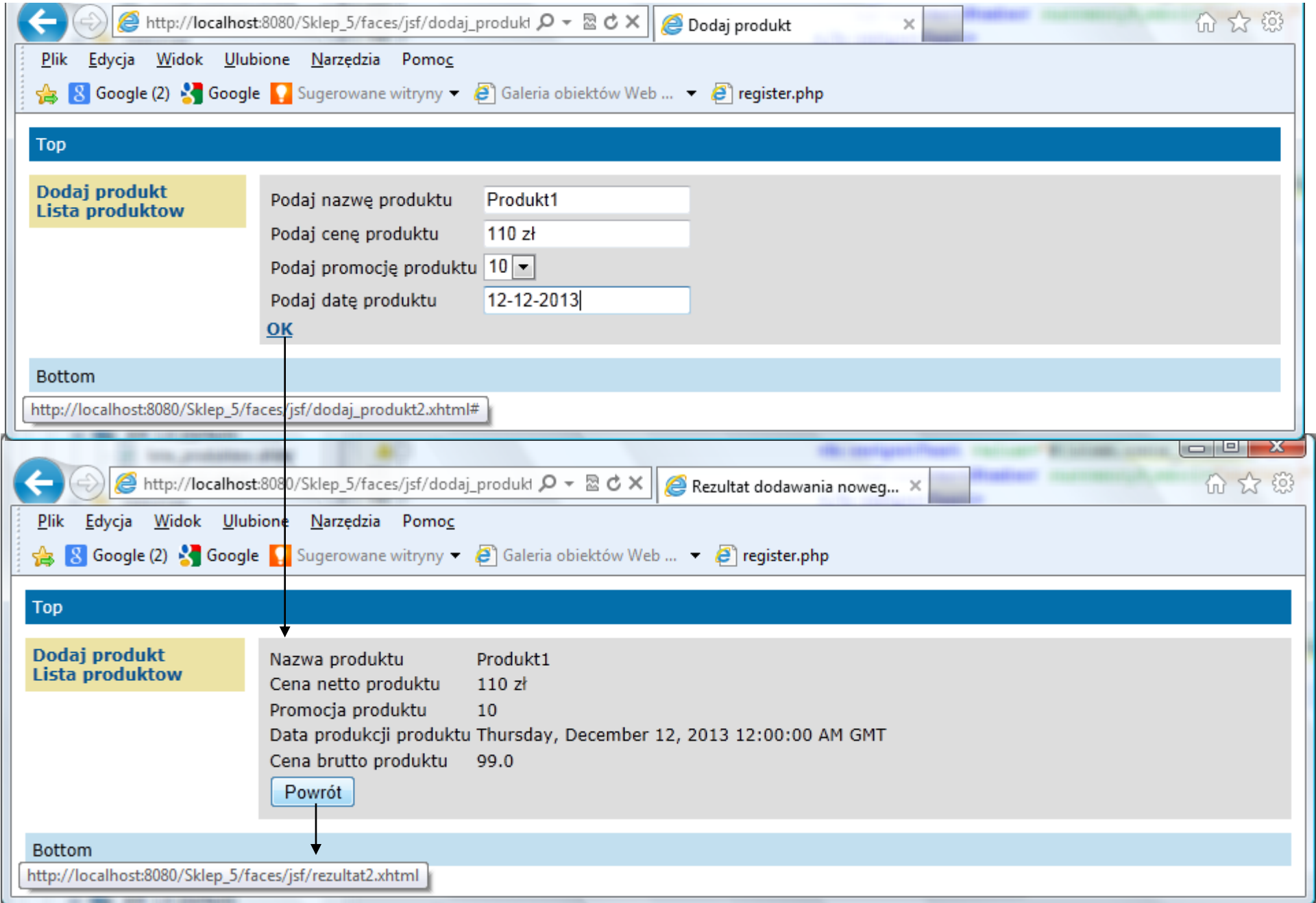
```
public String dodaj_produkt() {
        fasada.utworz_produkt(produkt_dto);
        powrot = 1;
        dane_produktu();
        recreateModel();
        getPagination().nextPage();
        return "rezultat2";
    }

public DataModel getItems() {
        if (items == null) {
            items = getPagination().createPageDataModel();
        }
        powrot = 1;
        return items;
    }
```

Zmienna **powrot** ma nadaną wartość 1 w przypadku opuszczenia strony **rezultat2.xhtml** za pomocą bloku „left” szablonu np. i przejście do strony **dodaj_produkt2.xhtml** lub **lista_produktow.xhtml** z renderowaniem właściwych przycisków.
.

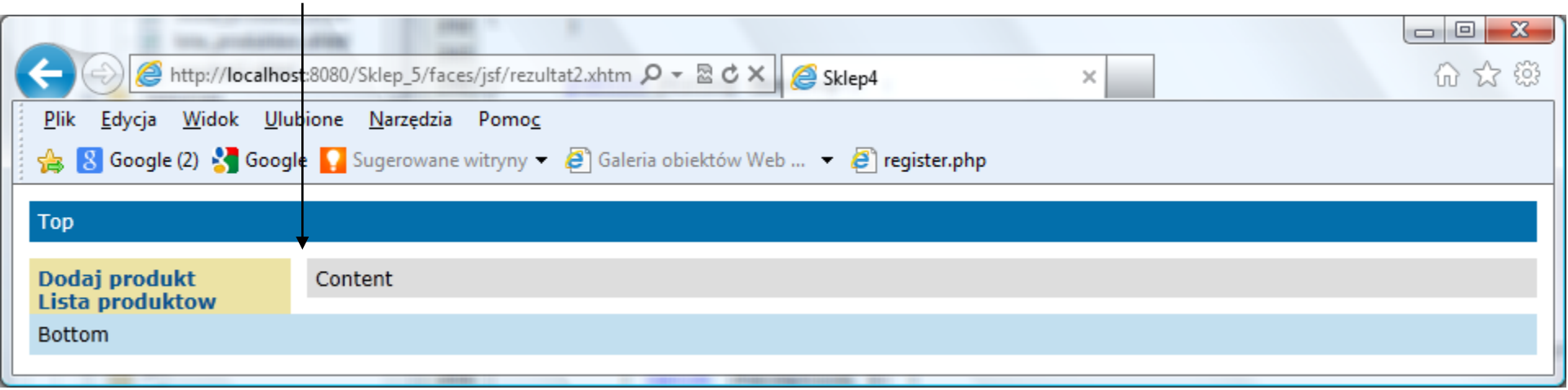
3.4. Uzupełnienie zawartości pliku **Bundle.properties**

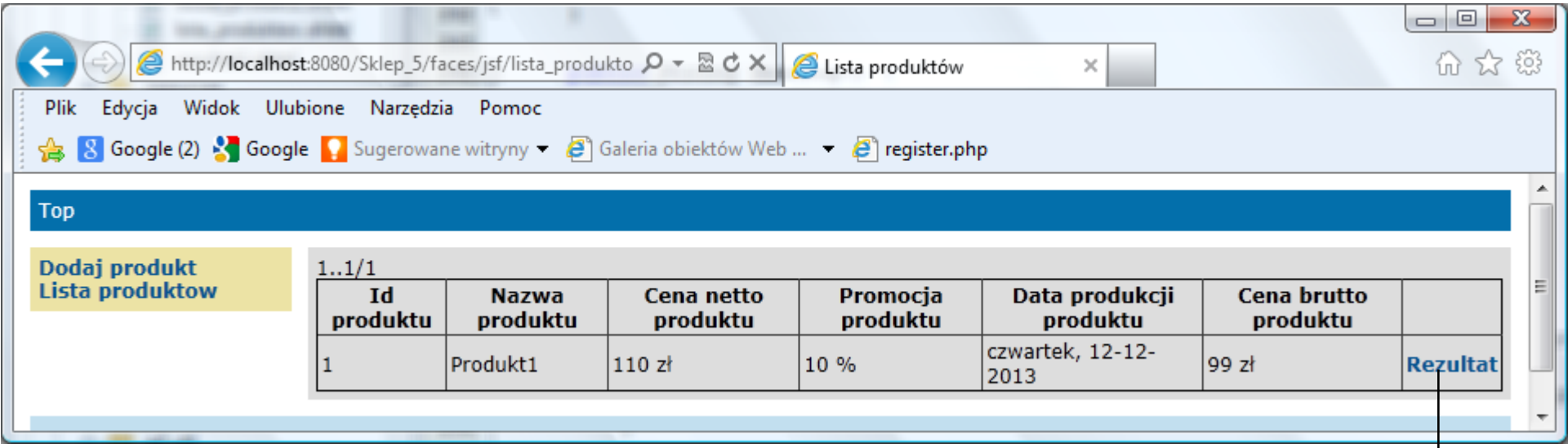
```
jsf.lista_produktow.rezultat=Rezultat
```

3.5. Prezentacja wielożywalności strony **rezulta2.xhtml** (1)

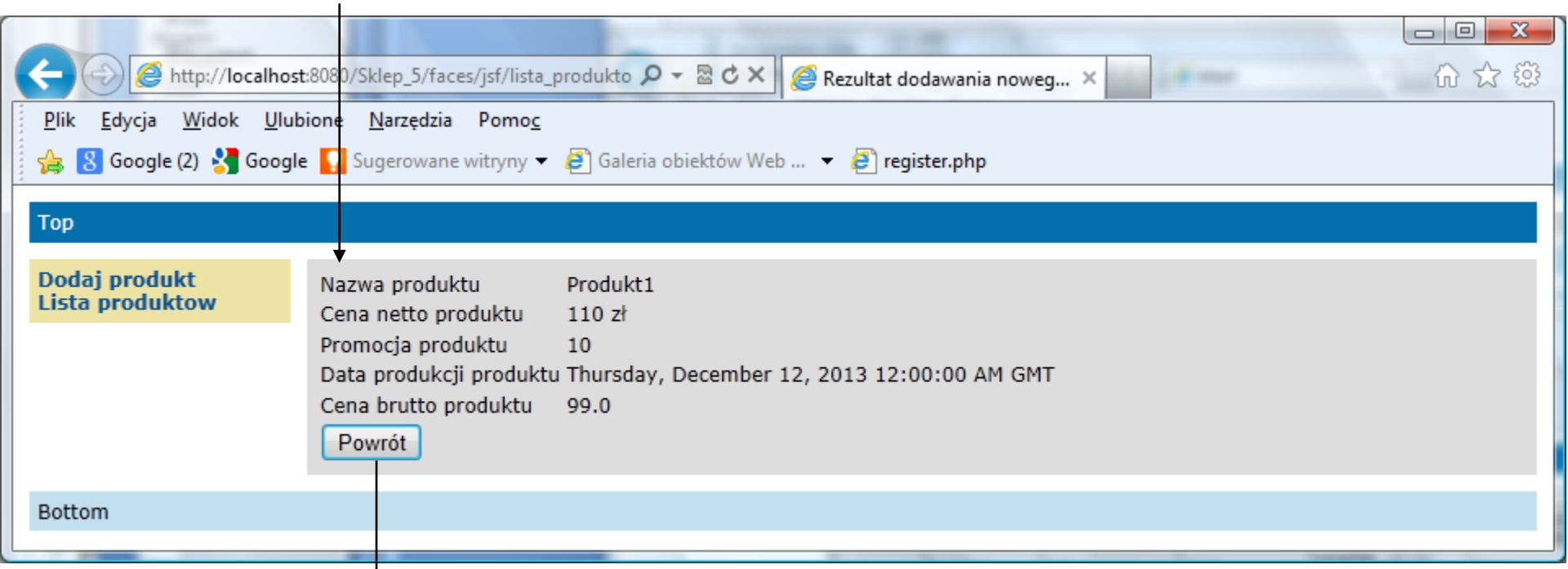


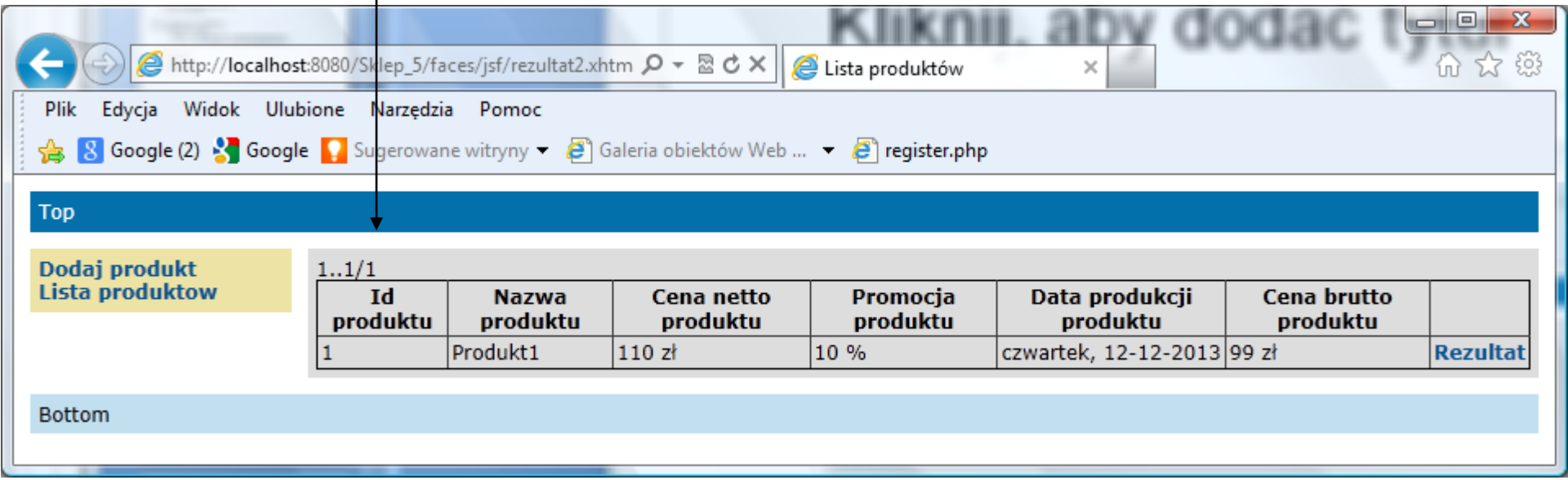
3.6. Prezentacja wielużywalności strony **rezultat2.xhtml** (2)





3.7. Prezentacja wielożywalności strony **rezultat2.xhtml** (3)





4. Wprowadzenie warunkowego renderowania strony dodaj_produkt2.xhtml w celu zapewnienia wielożywalności tej strony

4.1. Dodanie przycisku wywołującego stronę **dodaj_produkt2.xhtml** w celu modyfikacji danych produktu z wybranego wiersza tabeli typu **dataTable** na stronie **lista_produktow.xhtml.**

```
<h:column>
<f:facet name="header">
      <h:outputText value=" "/>
 </f:facet>
 <h:commandLink action="#{managed_produkt.prepareView}"
                      value="#{bundle['jsf.lista_produktow.rezultat']}"/>
<h:outputText value=" "/>
 <h:commandLink action="#{managed_produkt.prepareEdit}"
                       value="#{bundle['jsf.lista_produktow.edycja']}"/>
 <h:outputText value=" "/>
</h:column>
```

4.2. Dodanie nowych komunikatów do pliku Bundle.properties

Produkt_zmieniony=Zmiana danych produktu
Blad_modyfikacji=Nie dokonano zmiany danych produktu

jsf.lista_produktow.edycja=Edycja

4.3. Dodanie zmiennej **zmiana** do klasy **Managed_produkt** w celu renderowania strony **dodaj_produkt2.xhtml**

```
@ManagedBean
@SessionScoped
public class Managed_produkt {

    @EJB
    private Fasada_warstwy_biznesowej fasada;
    private DataModel items;
    private int stan = 1;
    private Produkt_dto produkt_dto = new Produkt_dto();
    private NumberConverter number_convert = new NumberConverter();
    private PaginationHelper pagination;
    private int powrot = 1;
    private int zmiana = 1;

public int getZmiana() {
        return zmiana;
    }
```

```
public String prepareEdit() {
        produkt_dto = (Produkt_dto) getItems().getRowData();
        zmiana = 0;
        return "dodaj_produkt2";
    }

    public String update() {
        try {
            getFasada().edit(produkt_dto);
            produkt_dto=new Produkt_dto();
            zmiana = 1;
            recreateModel();     //dodano w dniu 26.05.2013
            JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("Produkt_zmieniony"));
            return "lista_produktow";
        } catch (Exception e) {
            JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("Blad_modyfikacji"));
            return null;
        }
    }
```

Metoda **prepareEdit**, wywołana po kliknięciu na link **Edycja** na stronie **lista_produktow.xhtml** nadaje wartość 0 zmiennej zmiana, co powoduje przejście ze strony **lista_produktow.xhtml** do strony **dodaj_produkt2.xhtml** i powrót po edycji do strony **lista_produktow.xhtml**

```
public DataModel getItems() {
        if (items == null) {
            items = getPagination().createPageDataModel();    }
        zmiana = 1;
        powrot = 1;
        return items;
    }

 public String dodaj_produkt() {
        fasada.utworz_produkt(produkt_dto);
        powrot = 1;
        zmiana=1;
        dane_produktu();
        recreateModel();
        getPagination().nextPage();
        return "rezultat2";
    }
```

4.4. Zmiany na stronie **dodaj_produkt_2.xhtml**

```
<h:inputText
    id="nazwa"
    title="#{bundle['jsf.dodaj_produkt2.podaj_nazwa']}"
    value="#{managed_produkt.nazwa}"
    required="true„
    requiredMessage="#{bundle['jsf.dodaj_produkt2.podaj_nazwa_blad']}"
                    disabled="#{managed_produkt.zmiana==0}"  >
</h:inputText>
```

Dodanie atrybutu **disabled** pozwala pokazać widok komponentu na stronie jako nieaktywny – nie można wprowadzić nazwy produktu

---

```
<h:commandLink  action="#{managed_produkt.dodaj_produkt}"
                value="#{bundle['jsf.dodaj_produkt2.akcja']}"
                rendered="#{managed_produkt.zmiana==1}"/>
<h:commandLink  action="#{managed_produkt.update}"
                value="#{bundle['jsf.dodaj_produkt2.akcja']}"
                rendered="#{managed_produkt.zmiana==0}"/>
```

Dodawanie nowego produktu

Modyfikacja danych produktu

4.5. Metoda **edit** w klasie **Fasada_warstwy_biznesowej** wywołana podczas obsługi zdarzenia kliknięcia na przycisk OK. na stronie **dodaj_produkt2.xhtml,** gdy renderowany jest przycisk dla wartości zmiennej **zmiana=0** i wywołana jest metoda **update** z klasy **Managed_produkt**

```
public void edit(Produkt_dto o) {
        for (Produkt p : getProdukty()) {
            if (p.equals_(new Long(o.getId())) != null) {
                p.setCena(o.getCena());
                p.setData_produkcji(o.getData_produkcji());
                p.setPromocja(o.getPromocja());
                break;
            }
        }
    }
```

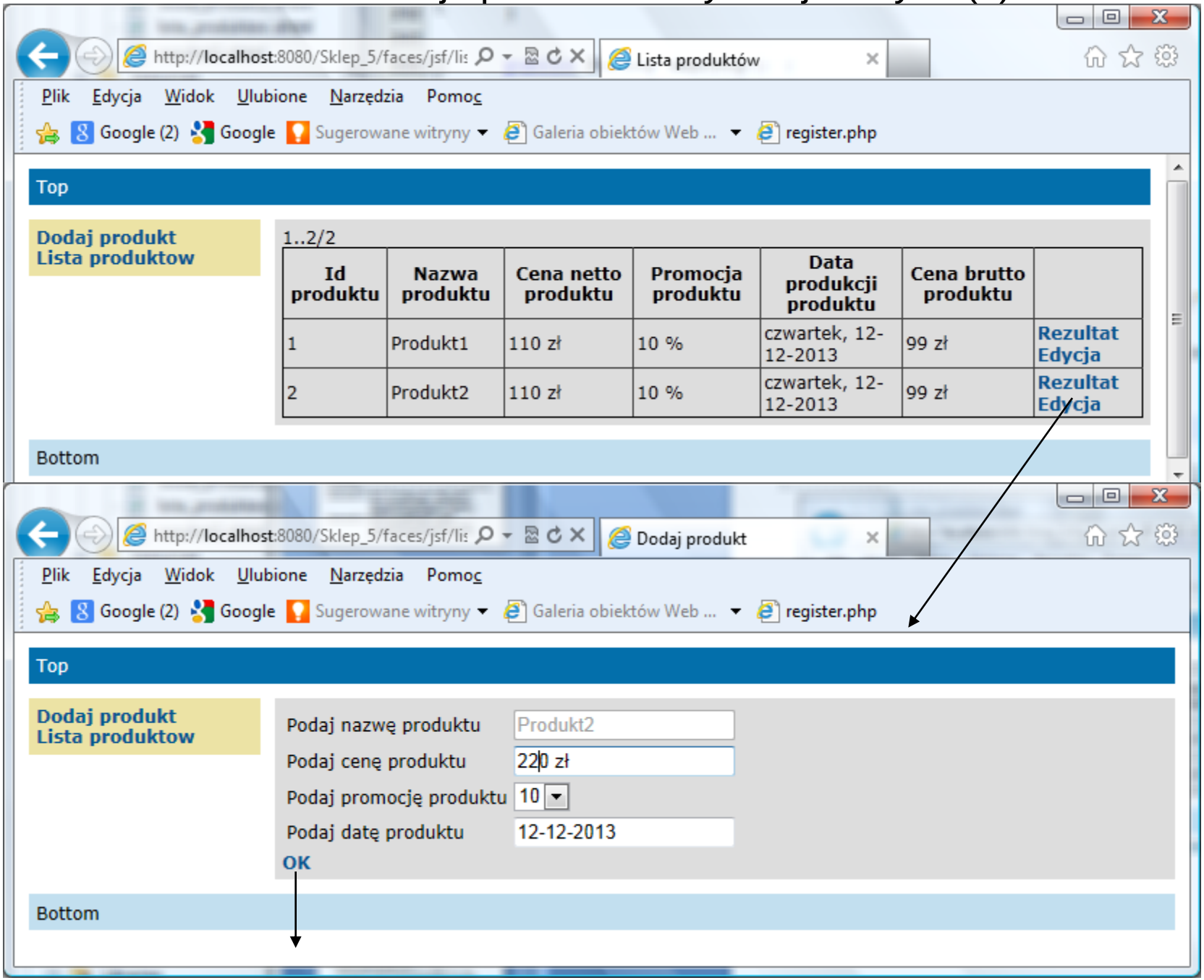
4.6. Metoda **equals_**, dodana do klasy **Produkt** w celu wyszukania obiektu typu Produkt do edycji – wywołana z metody edit z klasy **Fasada_warstwy_biznesowej**

```
public Produkt equals_(Object o)
    {
        if(id.equals(o))
            return this;
        else return null;
    }
```

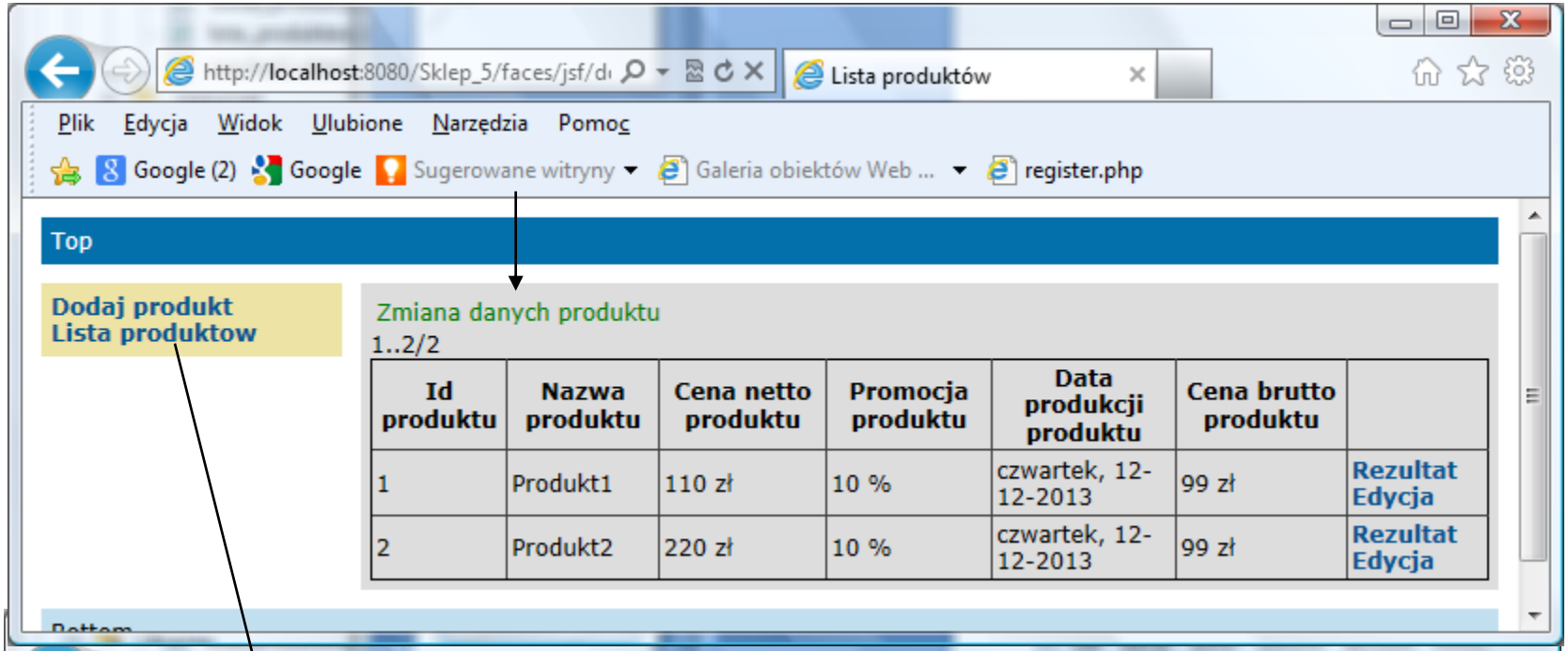
4.7. Dodane metody do klasy **JsfUtil** z pakietu **jsf.util** do obsługi błędów np. w metody edit klasy **Fasada_warstwy_biznesowej**

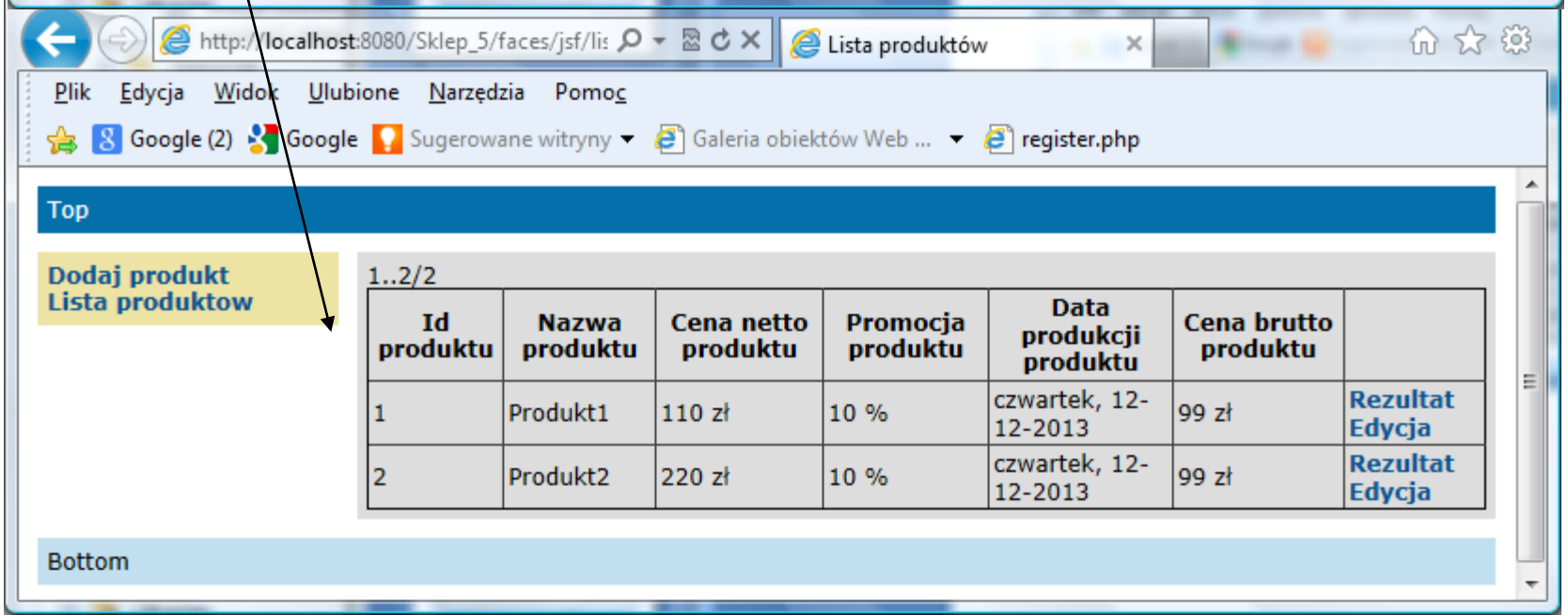
```
public static void addErrorMessage(Exception ex, String defaultMsg) {
        String msg = ex.getLocalizedMessage();
        if (msg != null && msg.length() > 0) {
            addErrorMessage(msg);
        } else {
            addErrorMessage(defaultMsg);        }
    }
 public static void addErrorMessage(String msg) {
     FacesMessage facesMsg =
                new FacesMessage(FacesMessage.SEVERITY_ERROR, msg, msg);
     FacesContext.getCurrentInstance().addMessage(null, facesMsg);
    }
public static void addSuccessMessage(String msg) {
     FacesMessage facesMsg =
                   new FacesMessage(FacesMessage.SEVERITY_INFO, msg, msg);
     FacesContext.getCurrentInstance().addMessage("successInfo", facesMsg);
    }
```

4.8. Prezentacja procesu modyfikacji danych (1)



4.9. Prezentacja procesu modyfikacji danych (1)





5. Dodanie na stronie **lista_produktow.xhtml** przycisku do usuwania danych

5.1. Dodanie przycisku do usuwania produktu wybranego w wierszu tabeli typu **dataTable** na stronie **lista_produktow.xhtml**

```
<h:column>
<f:facet name="header">
      <h:outputText value=" "/>
 </f:facet>
 <h:commandLink action="#{managed_produkt.prepareView}"
                      value="#{bundle['jsf.lista_produktow.rezultat']}"/>
<h:outputText value=" "/>
 <h:commandLink action="#{managed_produkt.prepareEdit}"
                       value="#{bundle['jsf.lista_produktow.edycja']}"/>
 <h:outputText value=" "/>
 <h:commandLink action="#{managed_produkt.destroy}"
                       value="#{bundle['jsf.lista_produktow.usun']}"/>
 </h:column>
```

5.2. Uzupełnienie zawartości pliku **Bundle.properties**

```
Usunieto_produkt=Produkt został usuniety
Blad_usuwania=Produkt nie został usuniety

jsf.lista_produktow.usun=Usun
```

5.3. Dodanie metod do obsługi usuwania danych w klasie **Managed_produkt**

```
public String destroy() {
        produkt_dto = (Produkt_dto) getItems().getRowData();
        performDestroy();
        recreateModel();
        return "lista_produktow";
    }

    private void performDestroy() {
        try {
            getFasada().remove(produkt_dto);
            JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("Usunieto_produkt"));
        } catch (Exception e) {
             JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("Blad_usuwania"));
        }
    }
```

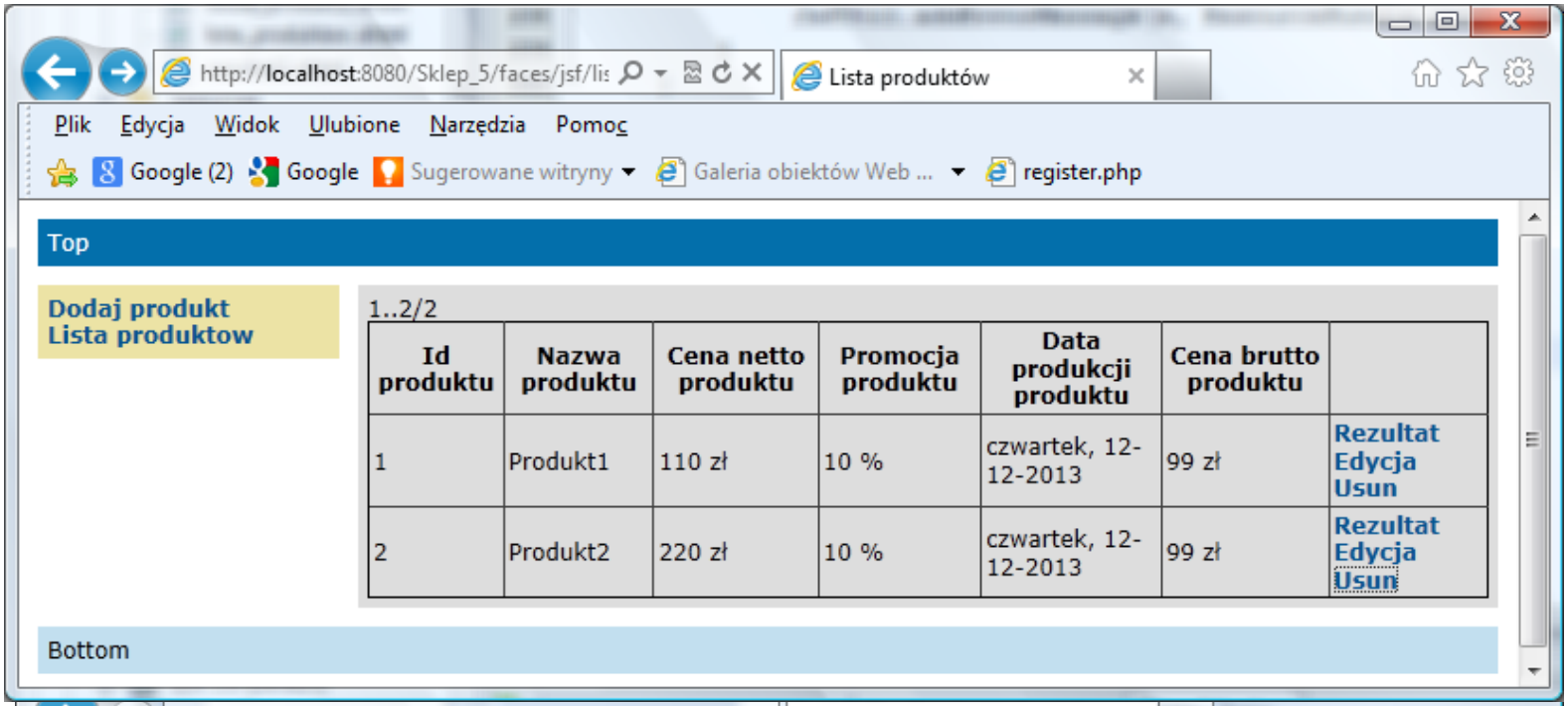
5.4. Dodanie metod **remove** do klasy **Fasada_warstwy_biznesowej** –modyfikacja kodu metody utworz_produkt. Metoda **max_klucz** wyznacza wartość unikatowego klucza

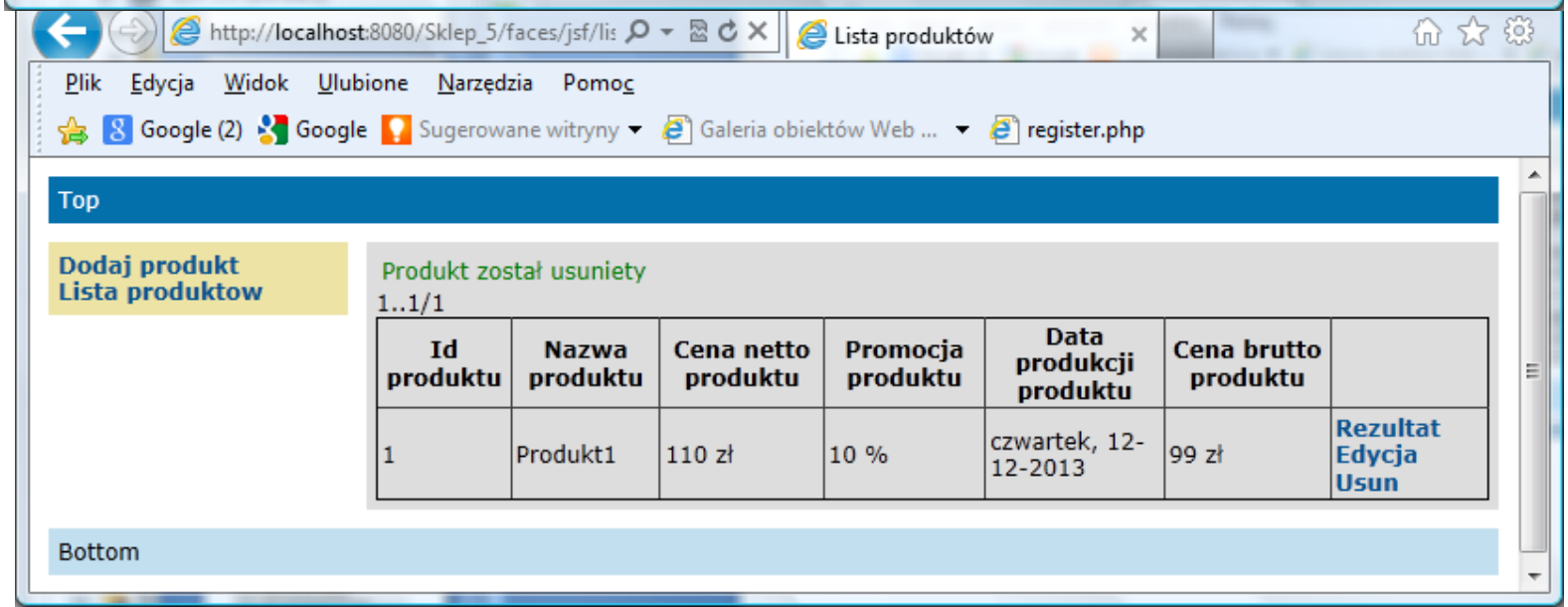
```
public void utworz_produkt(Produkt_dto produkt_dto) {
        Produkt produkt = wykonaj_produkt(produkt_dto);
        dodaj_produkt(produkt);
    }
  Produkt wykonaj_produkt(Produkt_dto produkt_dto) {
        Produkt produkt = new Produkt();
        max_klucz(); //dodano w dniu 26.05.2013
        produkt.setId(new Long(klucz));
        produkt.setNazwa(produkt_dto.getNazwa());
        produkt.setCena(produkt_dto.getCena());
        produkt.setPromocja(produkt_dto.getPromocja());
        produkt.setData_produkcji(produkt_dto.getData_produkcji());
        return produkt;
    }
public void remove(Produkt_dto p) {
        Produkt produkt = wykonaj_produkt(p);
        getProdukty().remove(produkt);
    }
 void max_klucz() { //dodano w dniu 26.05.2013
        long max = 0;
        for (Produkt p : produkty)
            if (p.getId() > max)   max = p.getId();
        klucz = max + 1;   }
```

Refaktoryzacja metody **utworz_produkt** przez dodanie metody **wykonaj_produkt,** która również została zastosowana w kodzie nowej metody **remove**

Metoda **max_klucz** wyznacza wartość kolejnego największego klucza zapewniając jego unikatowość

5.5. Prezentacja usuwania tytułu ksiazki (1)





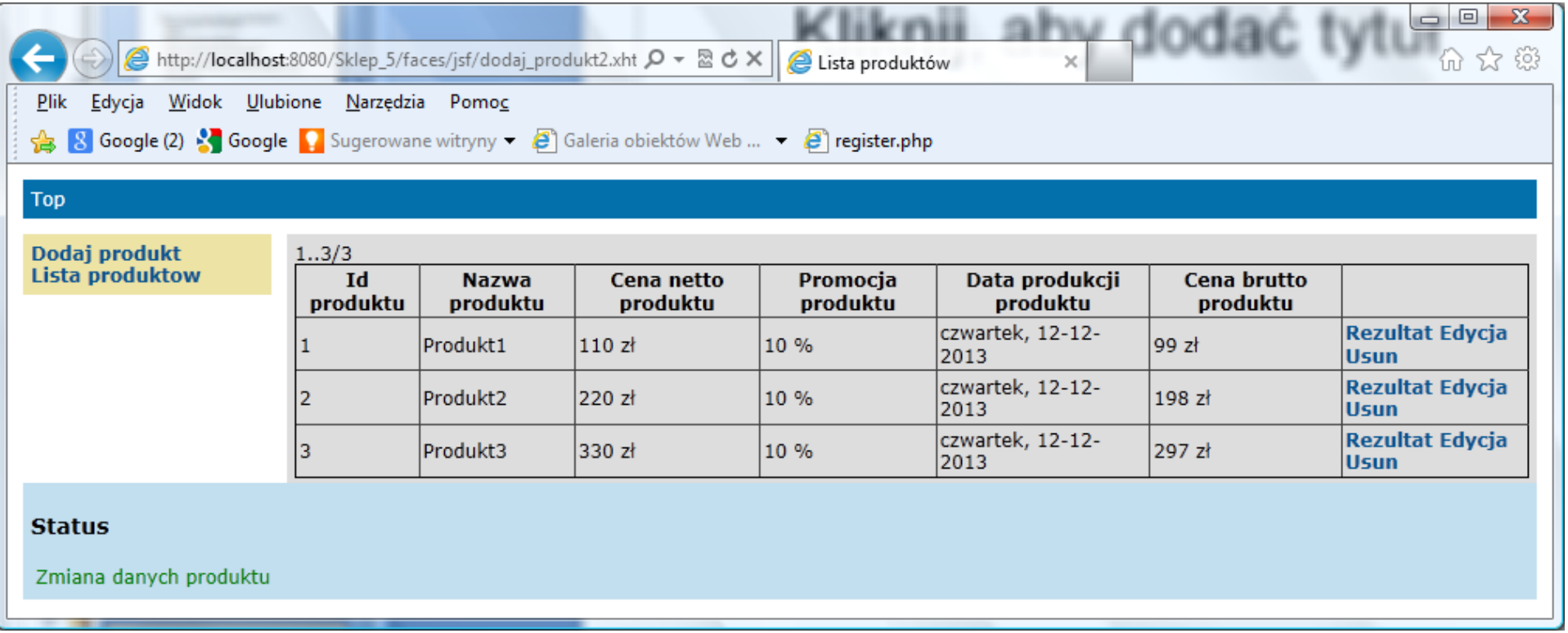
5.6. Przeniesienie panelu komunikatów **<h:messages** do części Bottom szablonu (plik **template.xhtml**)

```
<div style="height:150px">
   <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>
<div id="bottom" class="left_content">
      <h3>Status</h3>
      <h:panelGroup id="messagePanel" layout="block">
        <h:messages errorStyle="color: red" infoStyle="color: green"
                        layout="table"/>
      </h:panelGroup>
</div>
```

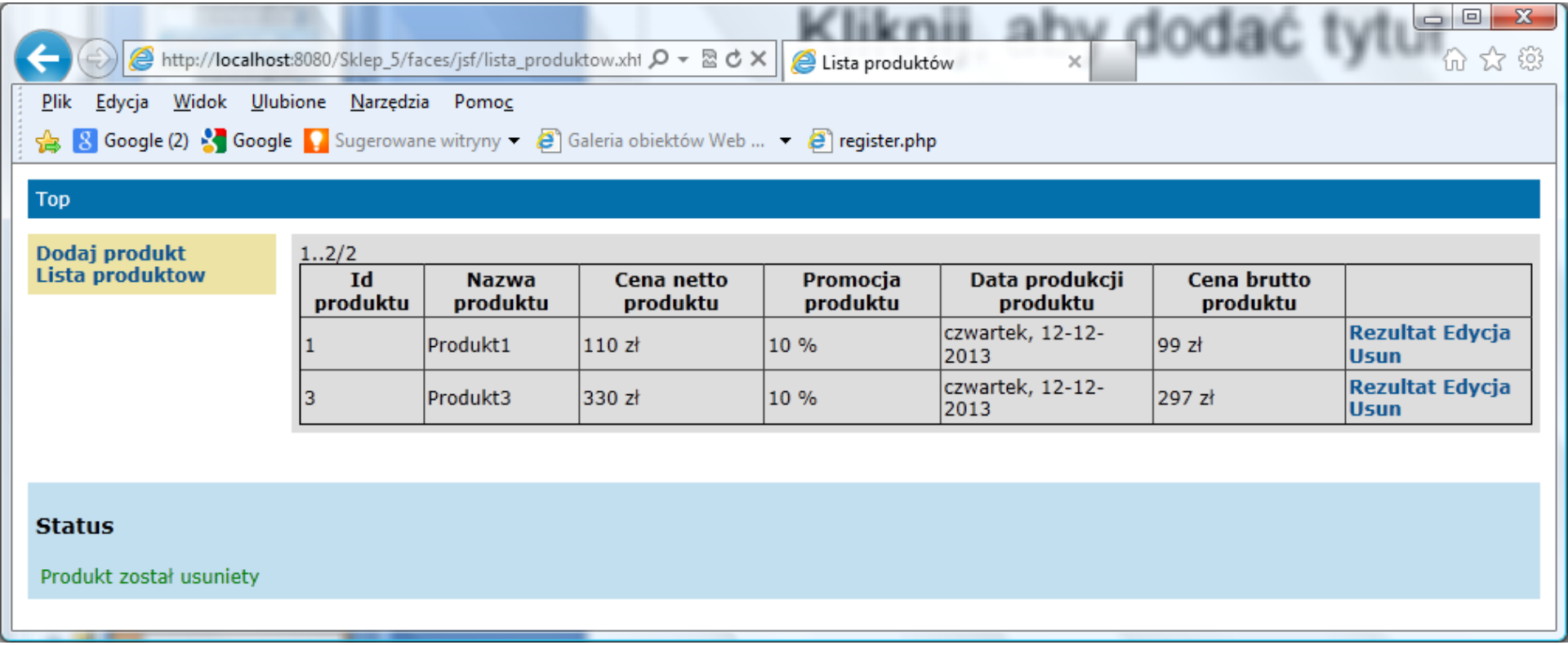
5.7. Wyeliminowanie panelu wiadomości **<h:messages** ze strony **lista_produktow.xhtml** – teraz komunikaty wyświetlane są w części **Bottom** o nazwie **Status** (poprzedni slajd)

```
<ui:define name="content">
    <h:form styleClass="jsfcrud_list_form">
        <h:panelGroup id="messagePanel" layout="block">
            <h:messages errorStyle="color: red" infoStyle="color: green"
                        layout="table"/>
        </h:panelGroup>
```

5.8. Prezentacja usuwania produktu (1)

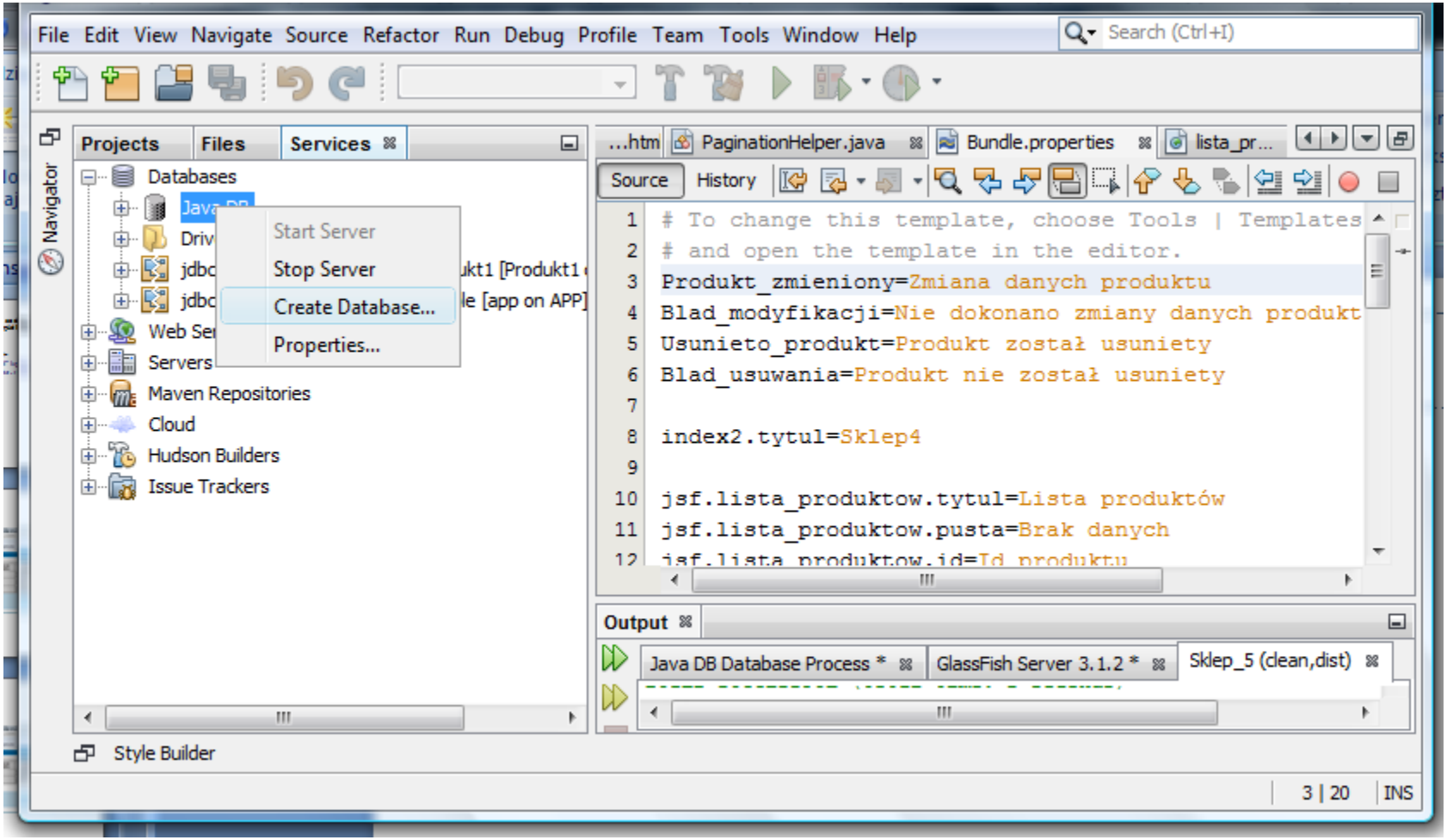


5.9. Prezentacja usuwania tytułu ksiazki (2)

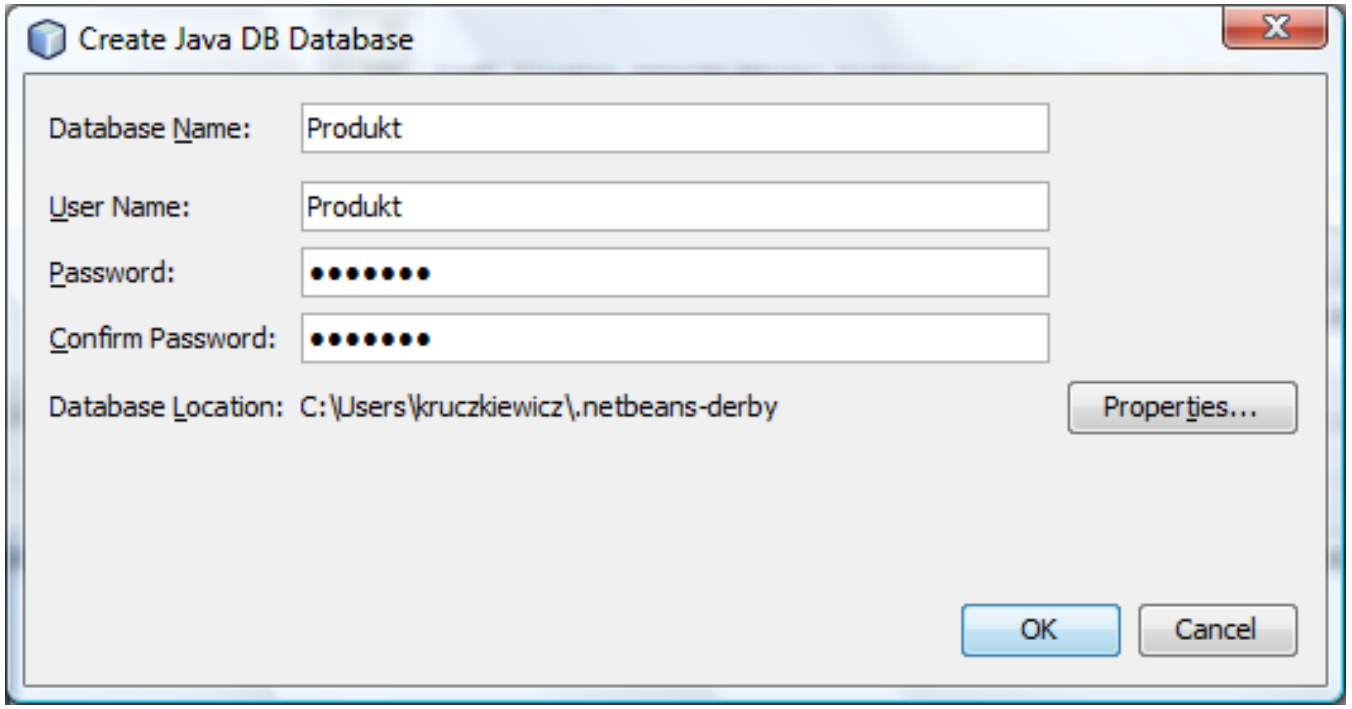


6. Zapis produktów do bazy danych metodą ORM (Object Relational Mapping)

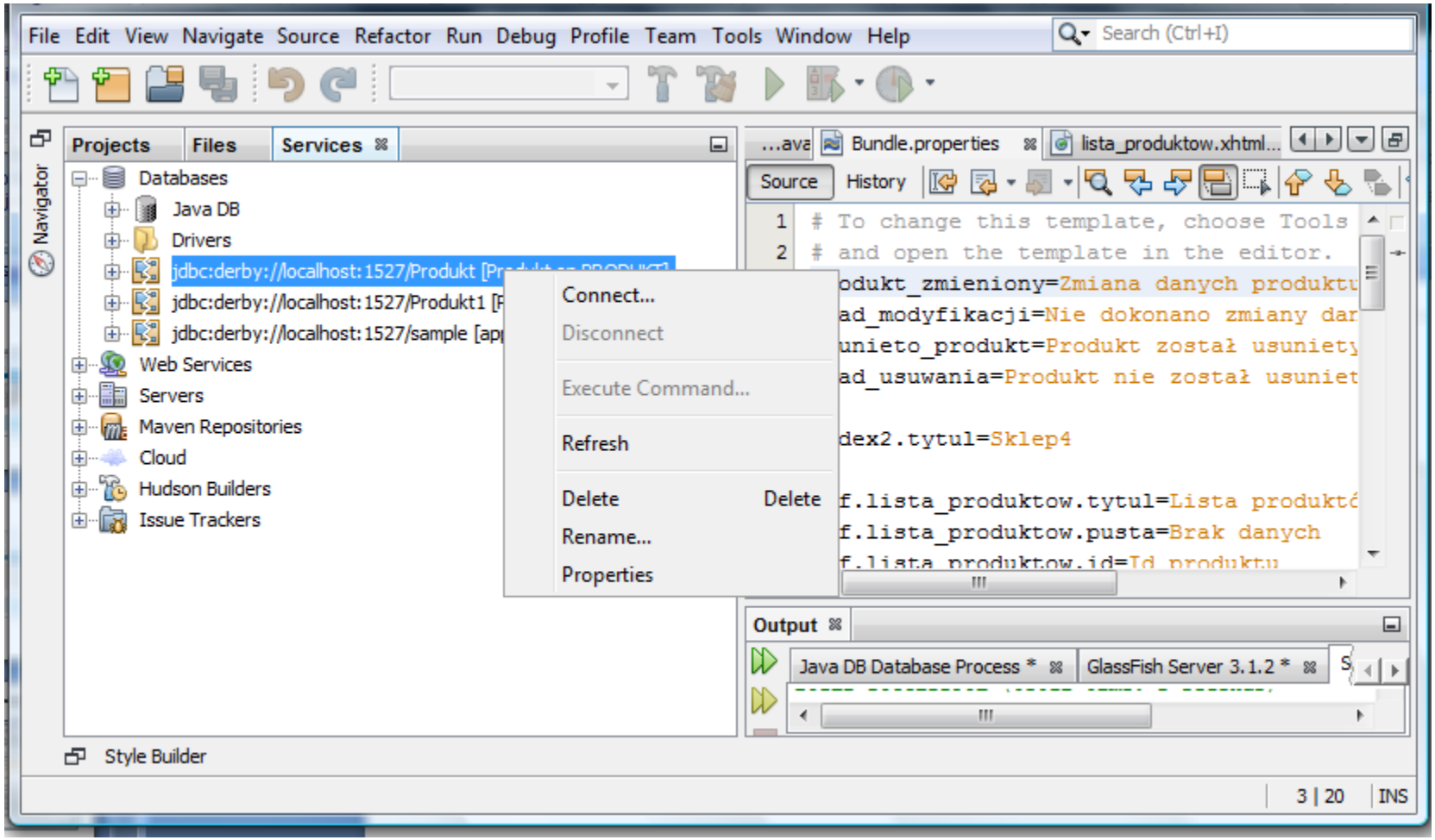
6.1 . Zapis do bazy danych – należy utworzyć pustą bazę danych w zakładce **Services** (prawy przycisk myszy na pozycji JavaDB po rozwinięciu pozycji Database i wybór **Create Database**)



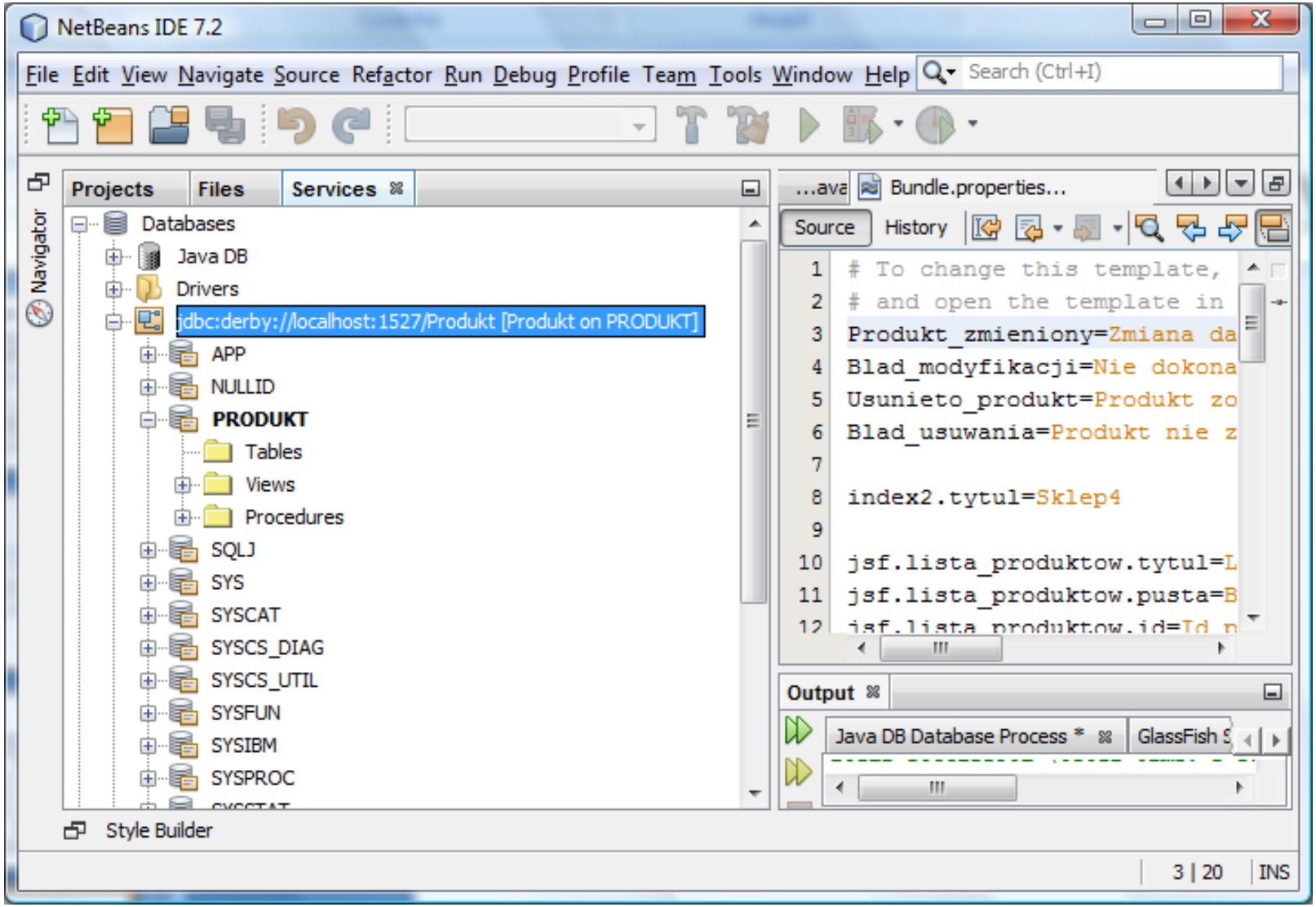
6.2. Wpisanie nazwy bazy danych (Database Name), loginu (User Name) i hasła (Password)



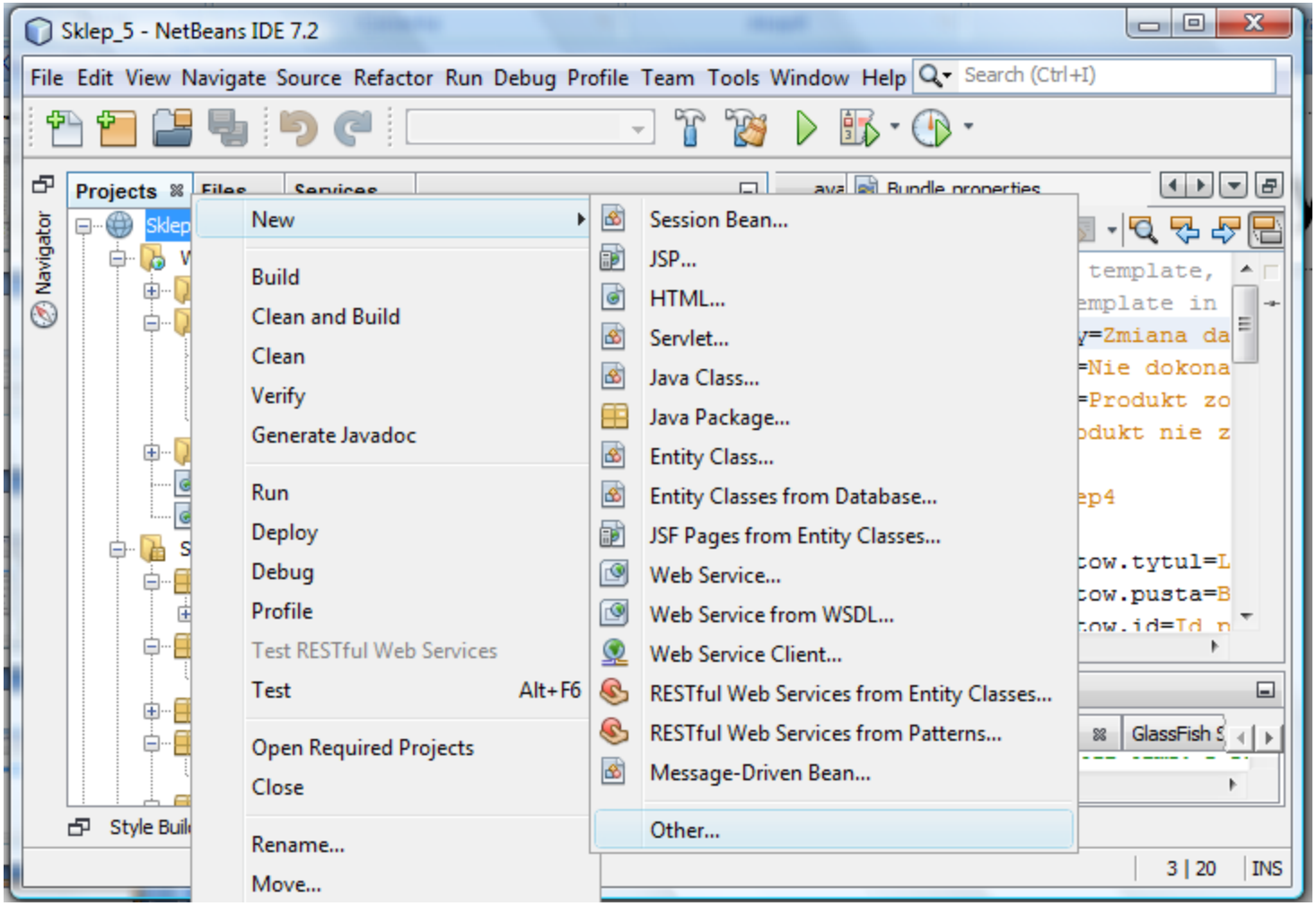
6.3. Połączenie z pustą bazą danych (wybór pozycji Connect po kliknięciu prawym klawiszem myszy na połączeniu)



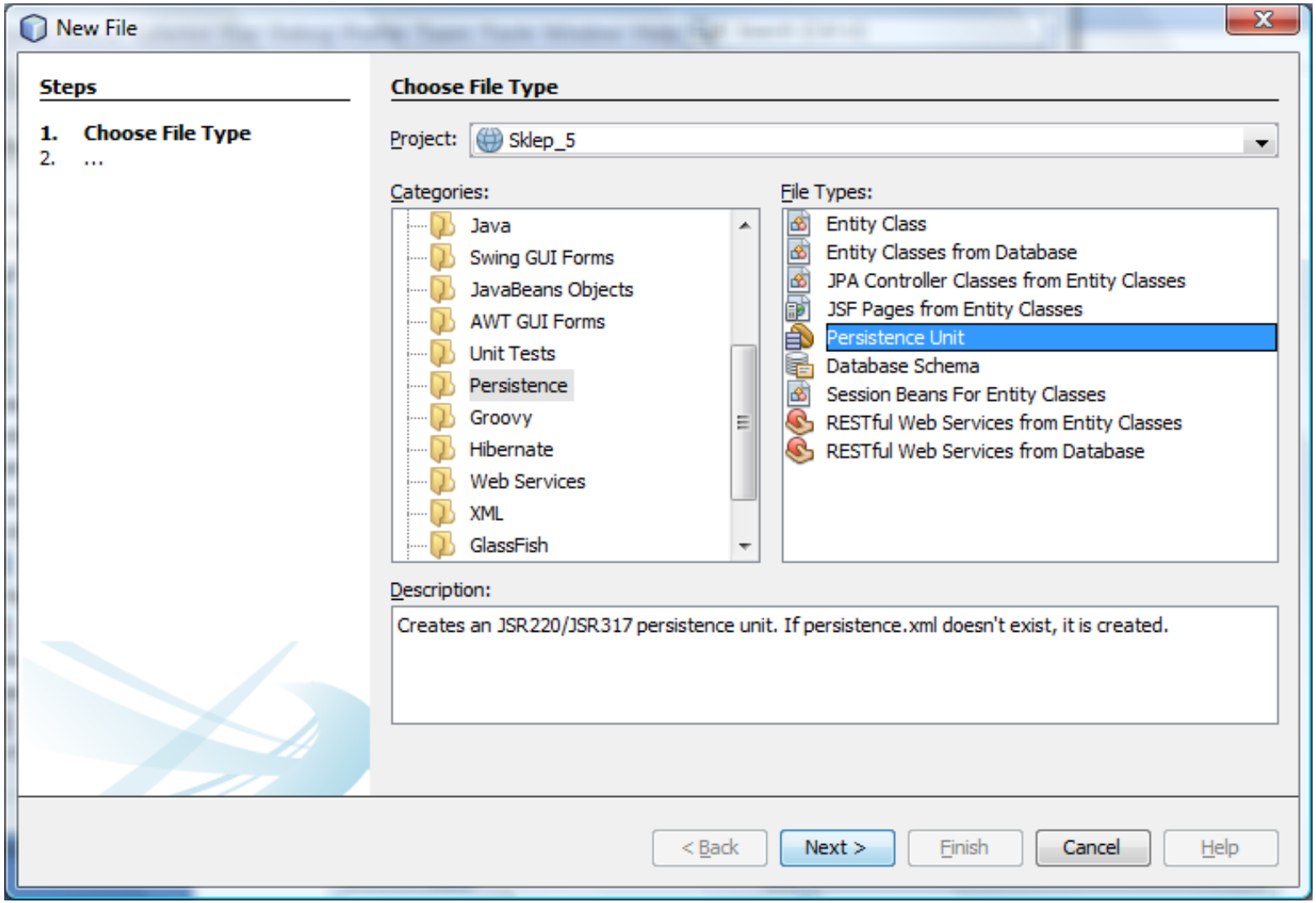
6.4. Widok utworzonej pustej bazy danych **Produkt**



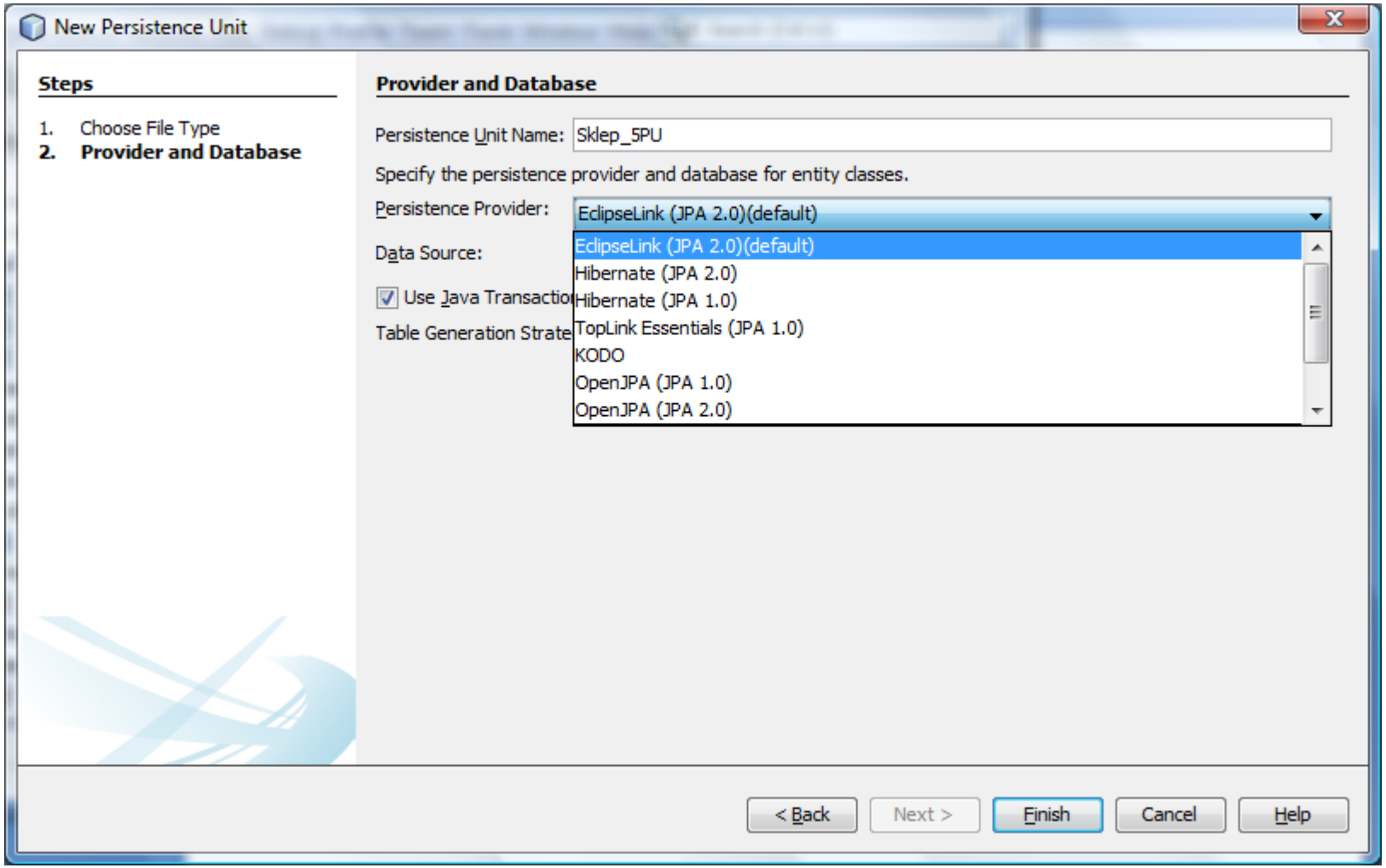
6.5. Dodanie pliku **persistence.xml** do projektu, potrzebnego w procesie utrwalania danych metodą ORM - **New/Other**

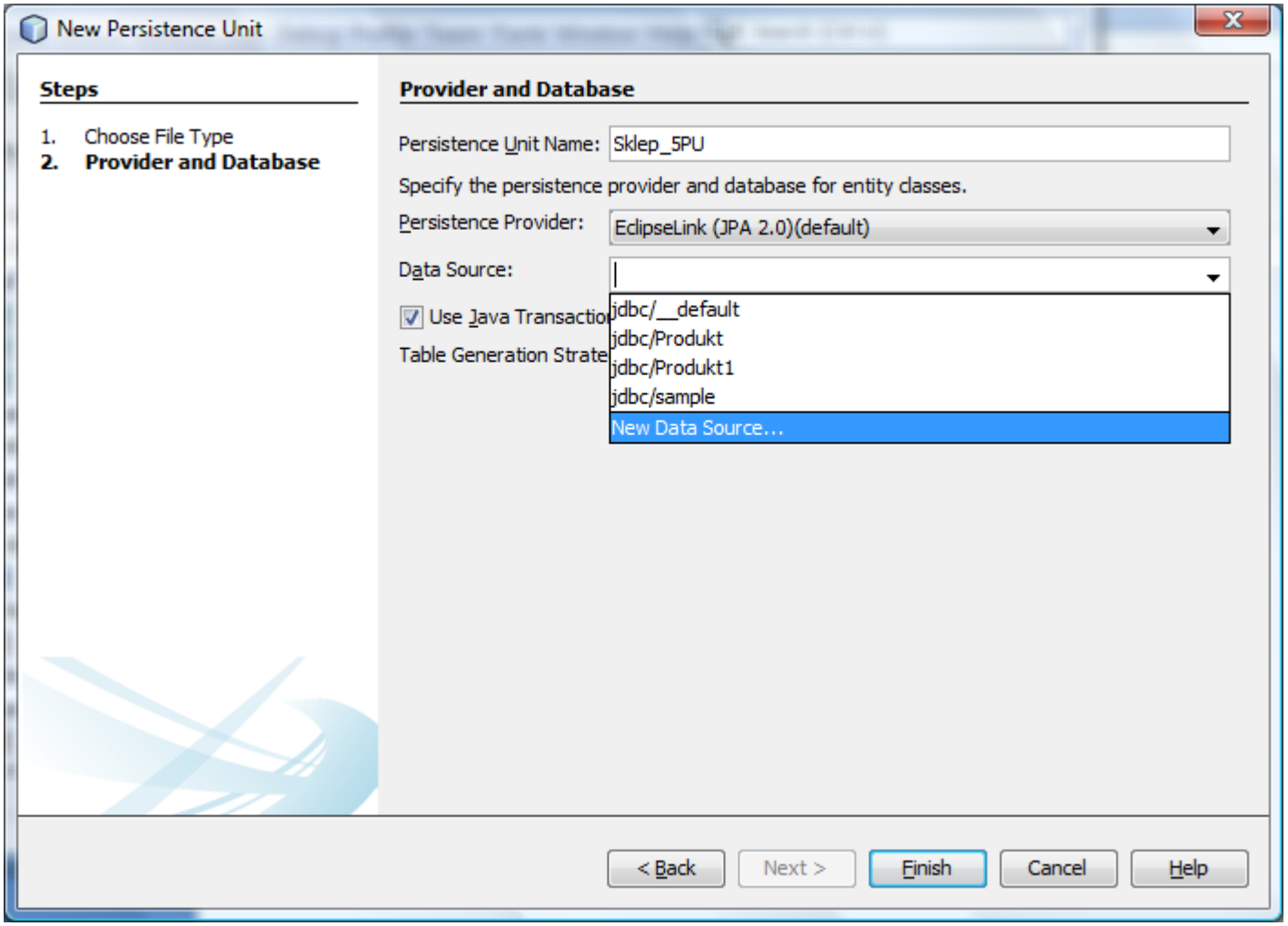


6.6. Wybór **Persistence/Persistence Unit**



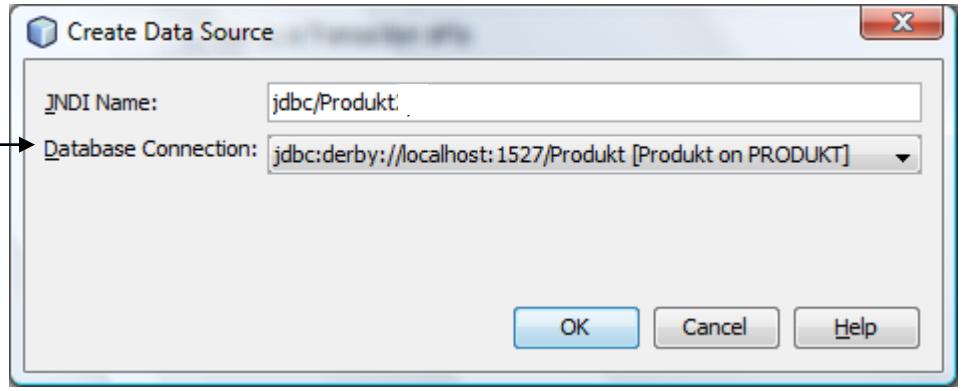
6.7. Wybór technologii EclipseLink do obsługi bazy danych metodą ORM



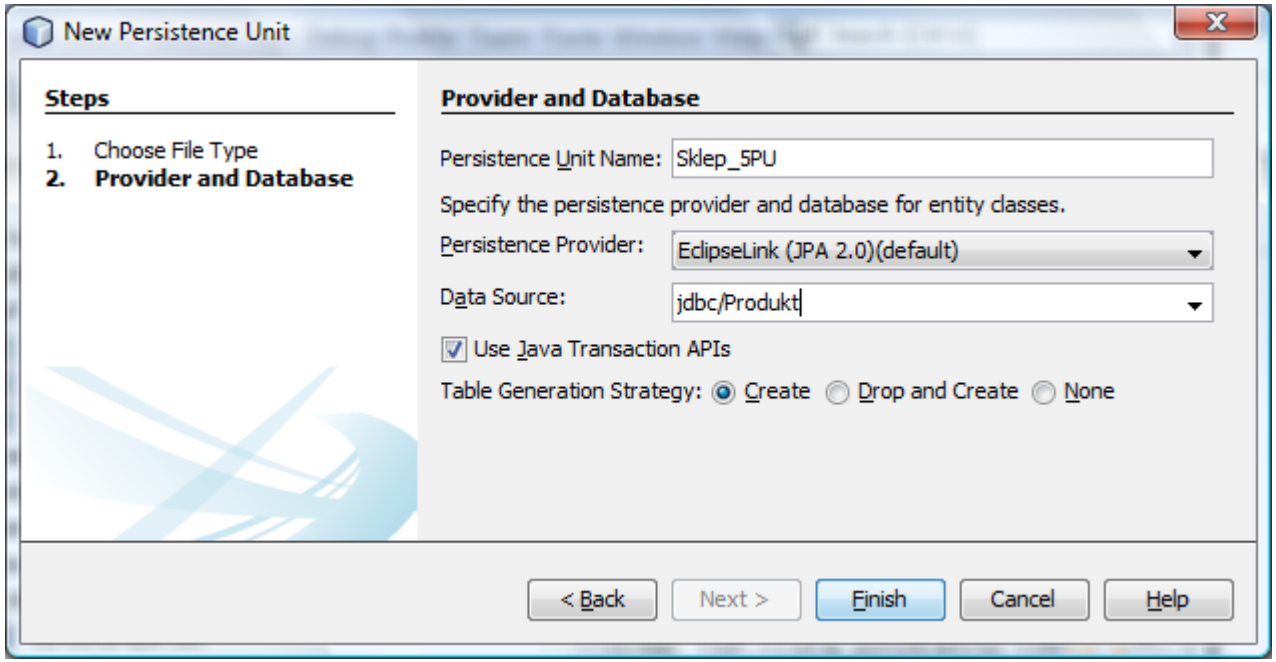


6.8. Utworzenie połączenia opartego na JNDI – nadanie nazwy **jdbc/Produkt**

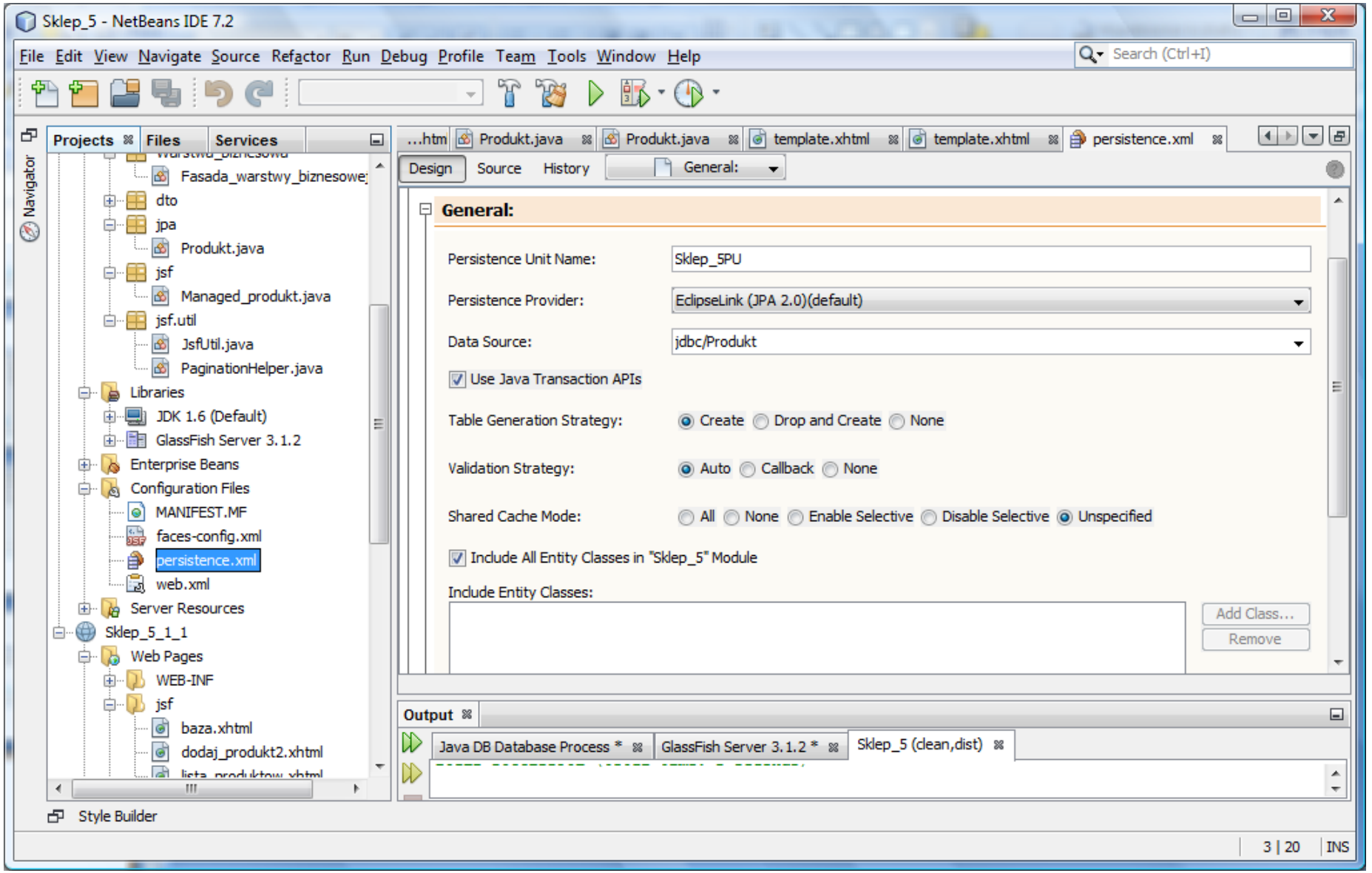
Wybór połączenia z bazą danych



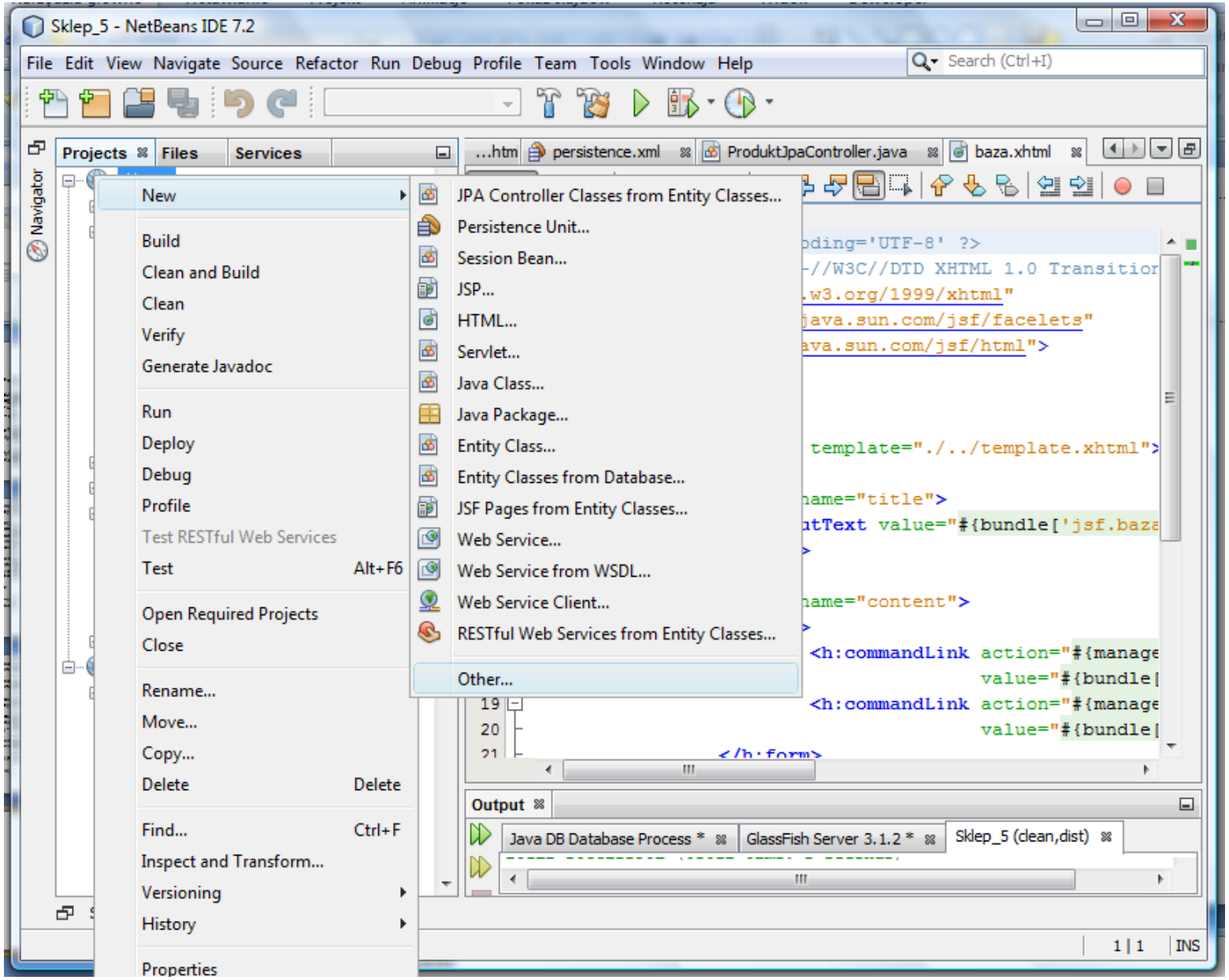
6.9. Gotowe połączenie z bazą danych - **Finish**



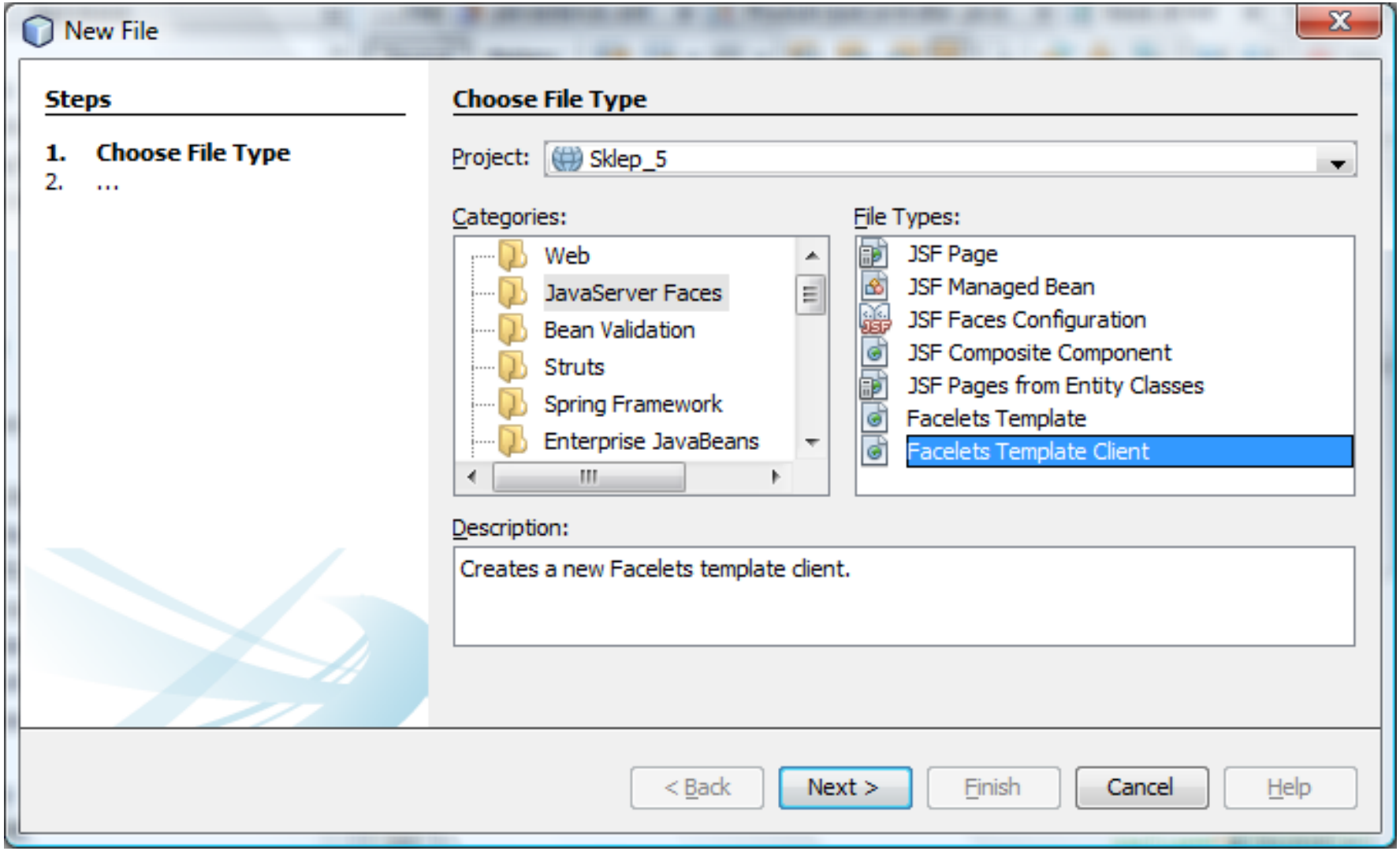
6.10. Pełna zawartość pliku **Persistence.xml**, pozwalająca na korzystanie z technologii ORM dla klas z adnotacją **@Entity** (tutaj klasa **Produkt**)



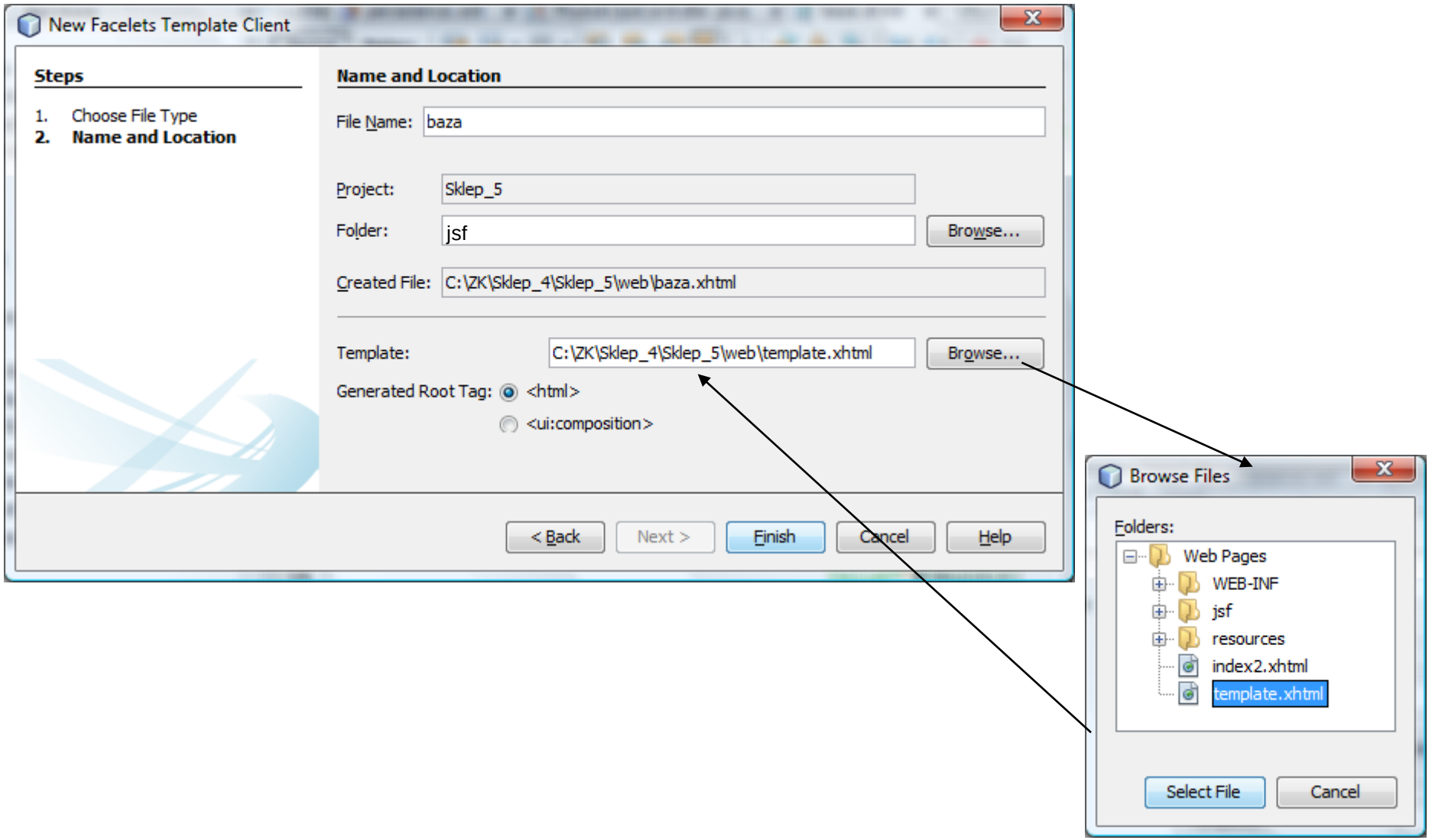
6.11. Dodanie nowej strony do obsługi bazy danych – **New/Other**



6.12. Dodanie strony opartej na szablonie **template.xml** (1)



6.13. Dodanie strony **baza.xhtml** opartej na szablonie **template.xml** (2)



6.14. Zawartość wygenerowanego pliku **baza.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://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">
    <body>
        <ui:composition template="./../template.xhtml">
            <ui:define name="title">
                <h:outputText value="#{bundle['jsf.baza.tytul']}"></h:outputText>
            </ui:define>
```

6.15. Modyfikacja zawartości strony **baza.xhtml** opartej na szablonie **template.xml**

```
<ui:define name="content">
        <h:form>
            <h:commandLink action="#{managed_produkt.zapisz}"
                value="#{bundle['jsf.baza.zapisz']}"/><br/>
            <h:commandLink action="#{managed_produkt.pobierz}"
                value="#{bundle['jsf.baza.pobierz']}"/><br/>
        </h:form>
    </ui:define>
  </ui:composition>
</body>
</html>
```

Pierwszy comandLink obsługuje **zapis** do bazy danych, a drugi **odczyt** z bazy danych

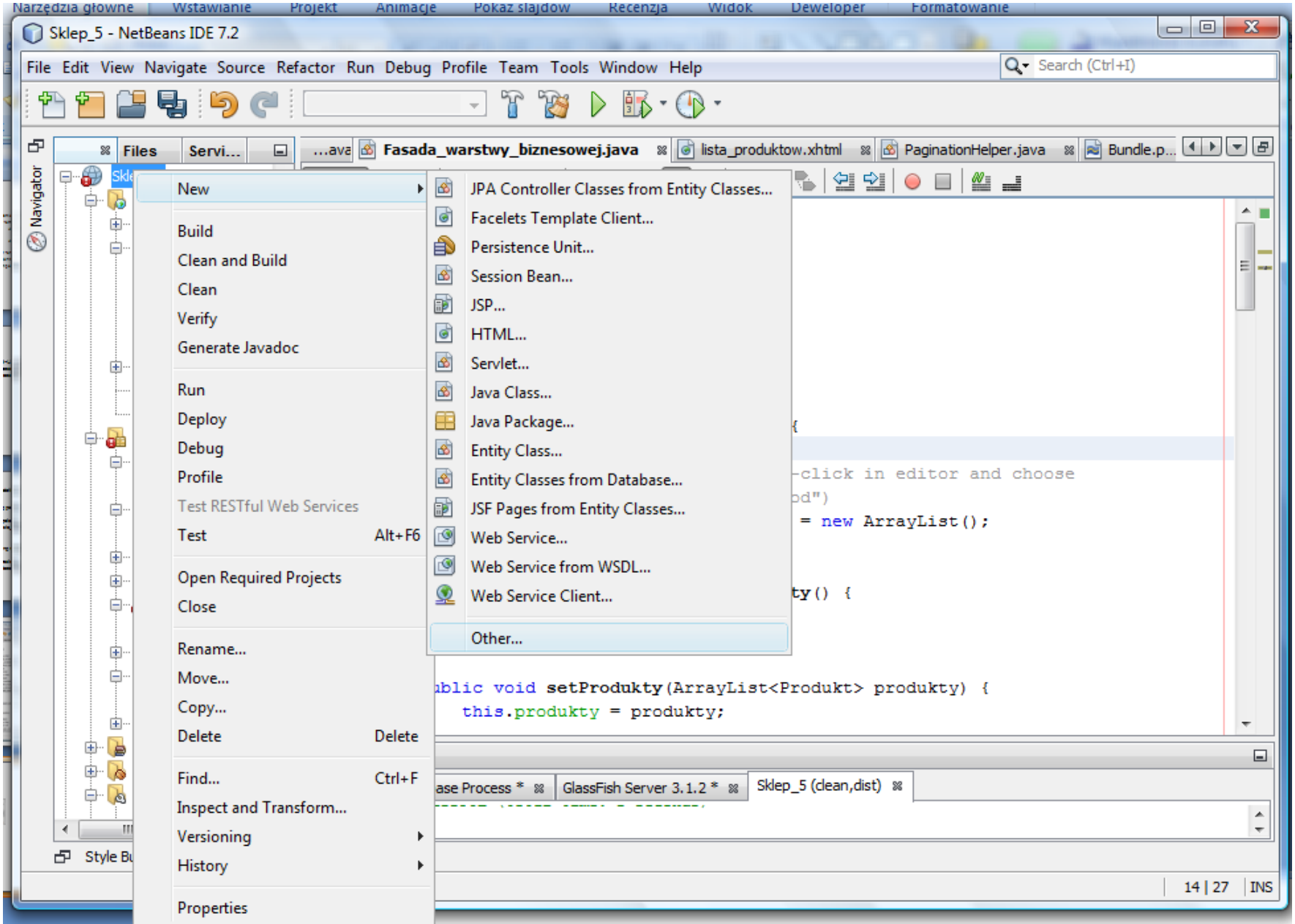
6.16. Uzupełnienie pliku **Bundle.properties**

```
jsf.baza.tytul=Utrwalanie danych
jsf.baza.zapisz=Zapisz do bazy danych
jsf.baza.pobierz=Pobierz z baza danych
```

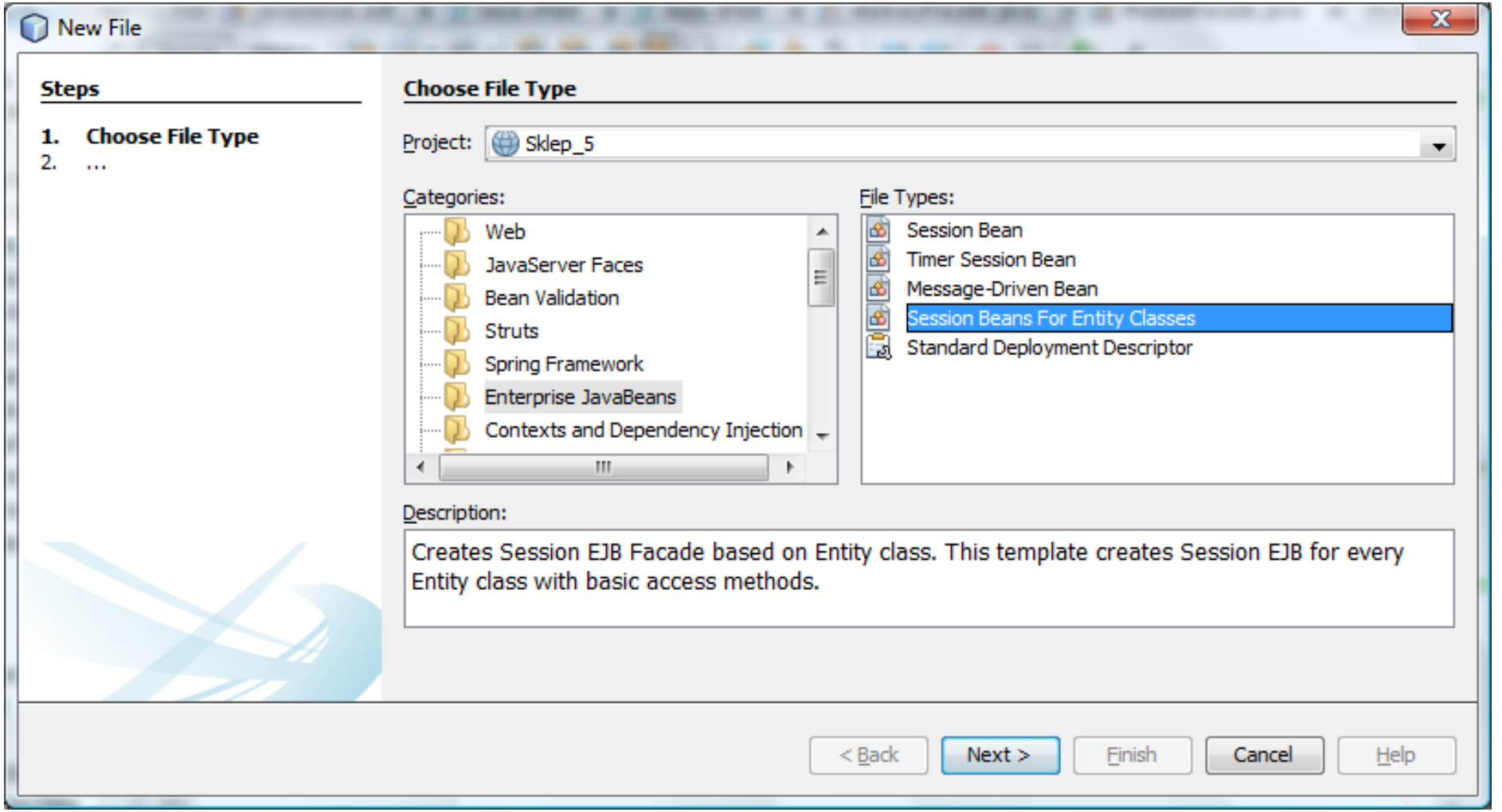
6.17. Uzupełnienie kodu pliku **template.xhtml** – bloku ”left”, czyli dodanie znacznika **h:link** do wywoływania strony **baza.xhtml**

```
<div id="left">
            <h:link  outcome="/faces/jsf/dodaj_produkt2"
                  value="Dodaj produkt"/> <br/>
            <h:link outcome="/faces/jsf/lista_produktow"
                  value="Lista produktow"/><br/>
             <h:link outcome="/faces/jsf/baza"
                        value="#{bundle['jsf.baza.tytul']}"/>
</div>
```

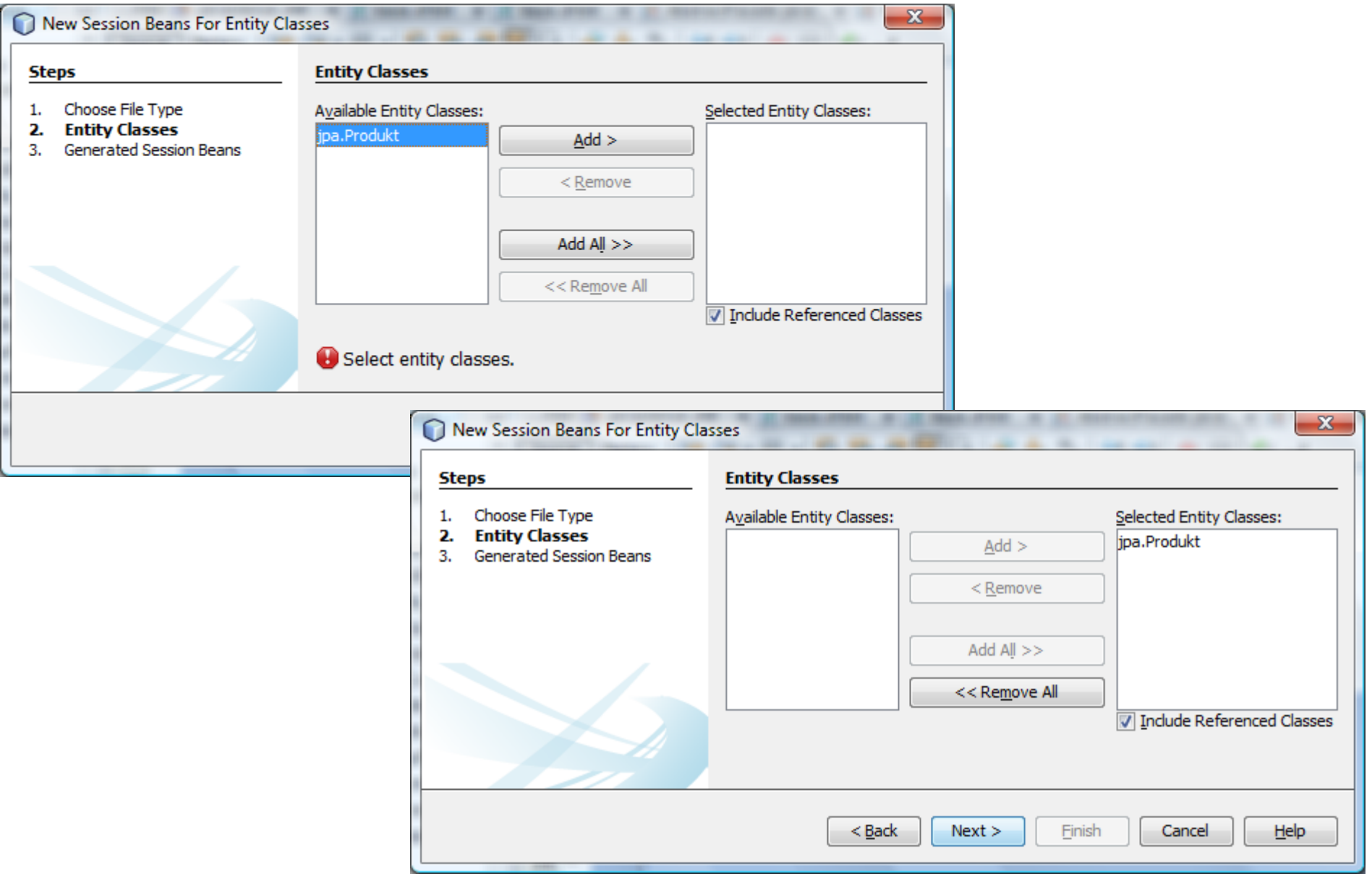
6.18. Dodanie klasy do utrwalania obiektów typu Produkt – **New/Other**



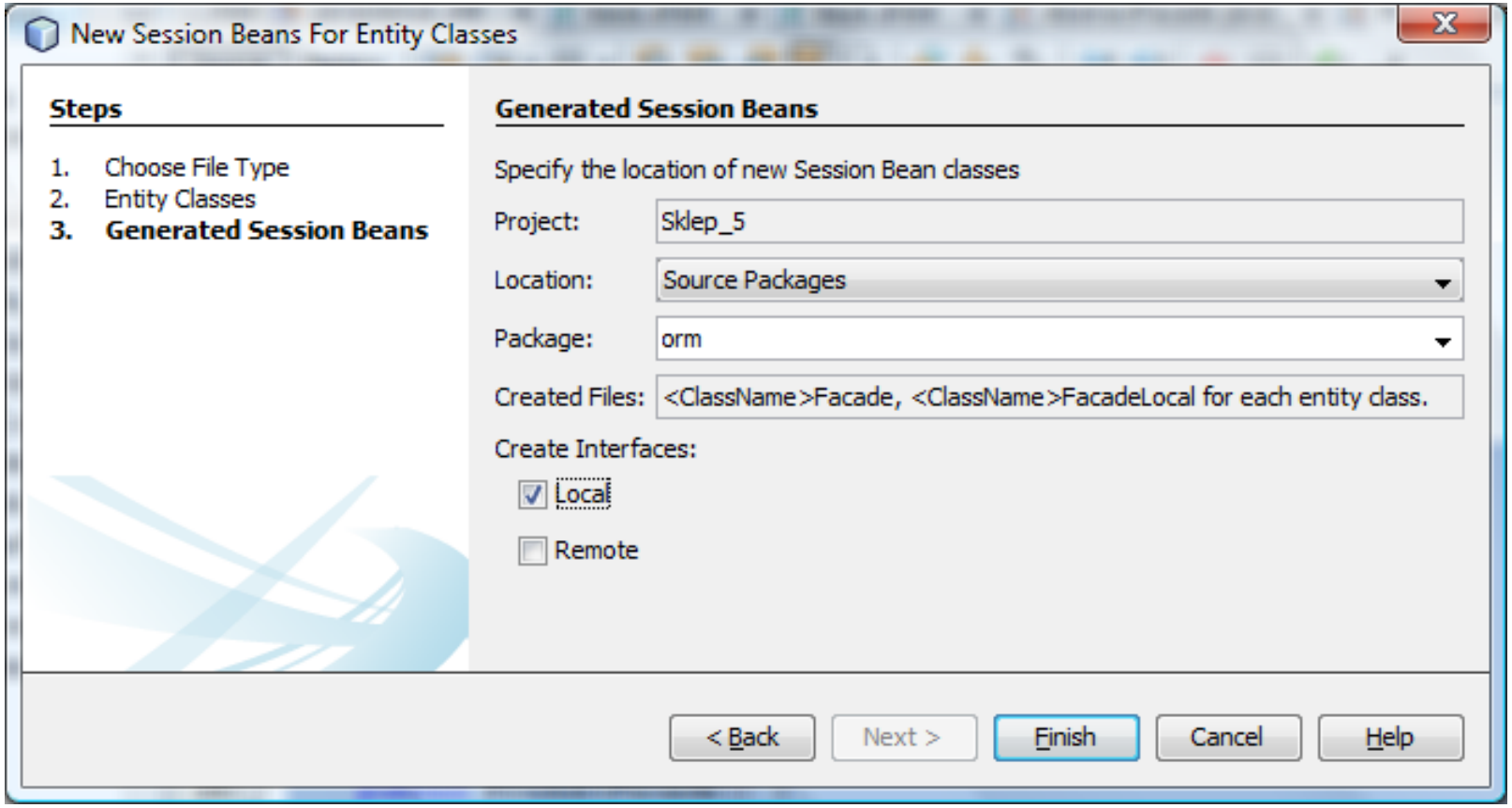
6.19. Utworzenie pliku wspierającego ORM klasy Produkt – **Enterprise JavaBeans/SessionBeans for Entity Classes**



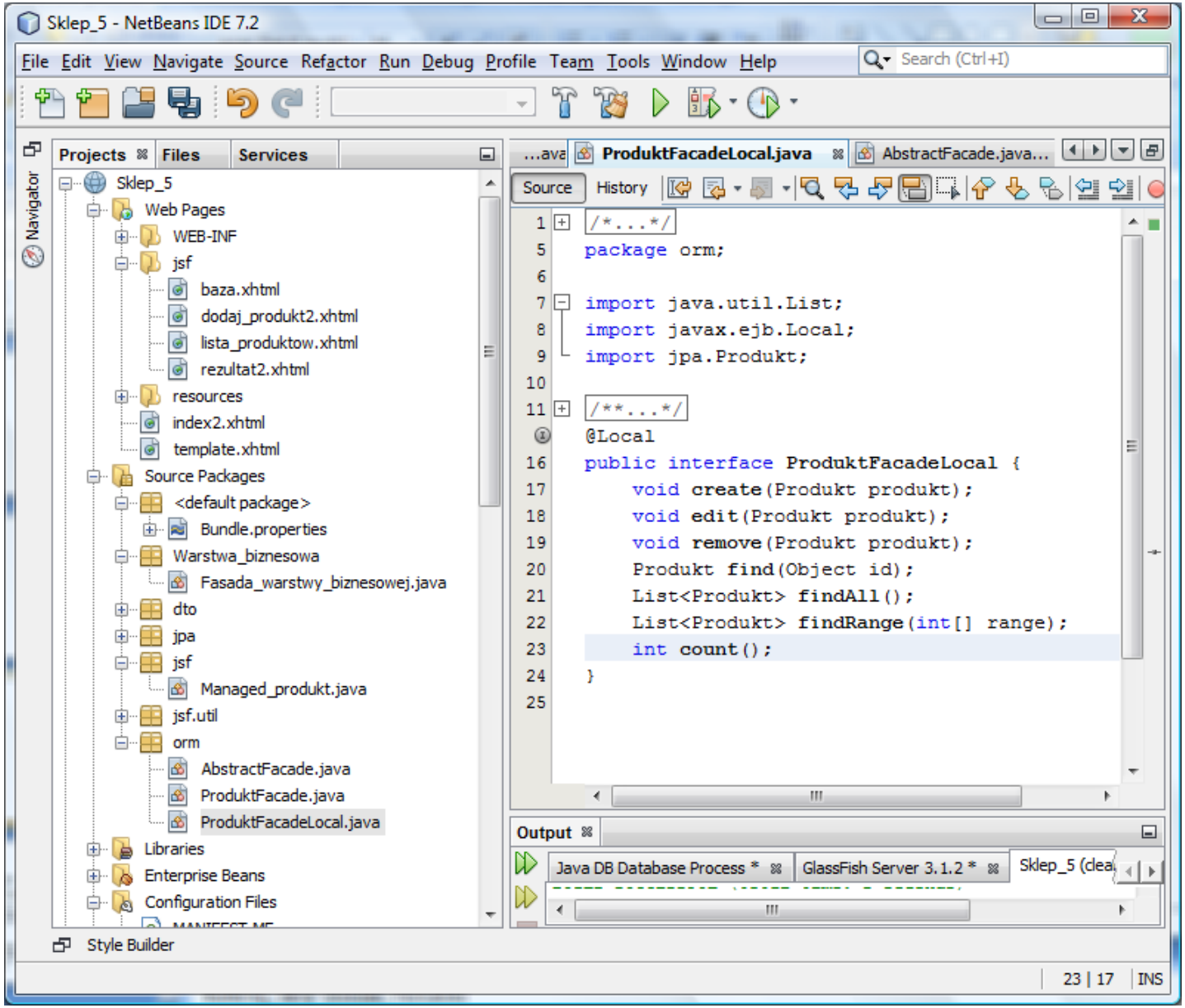
6.20. Wskazanie klasy Produkt typu **Entity**



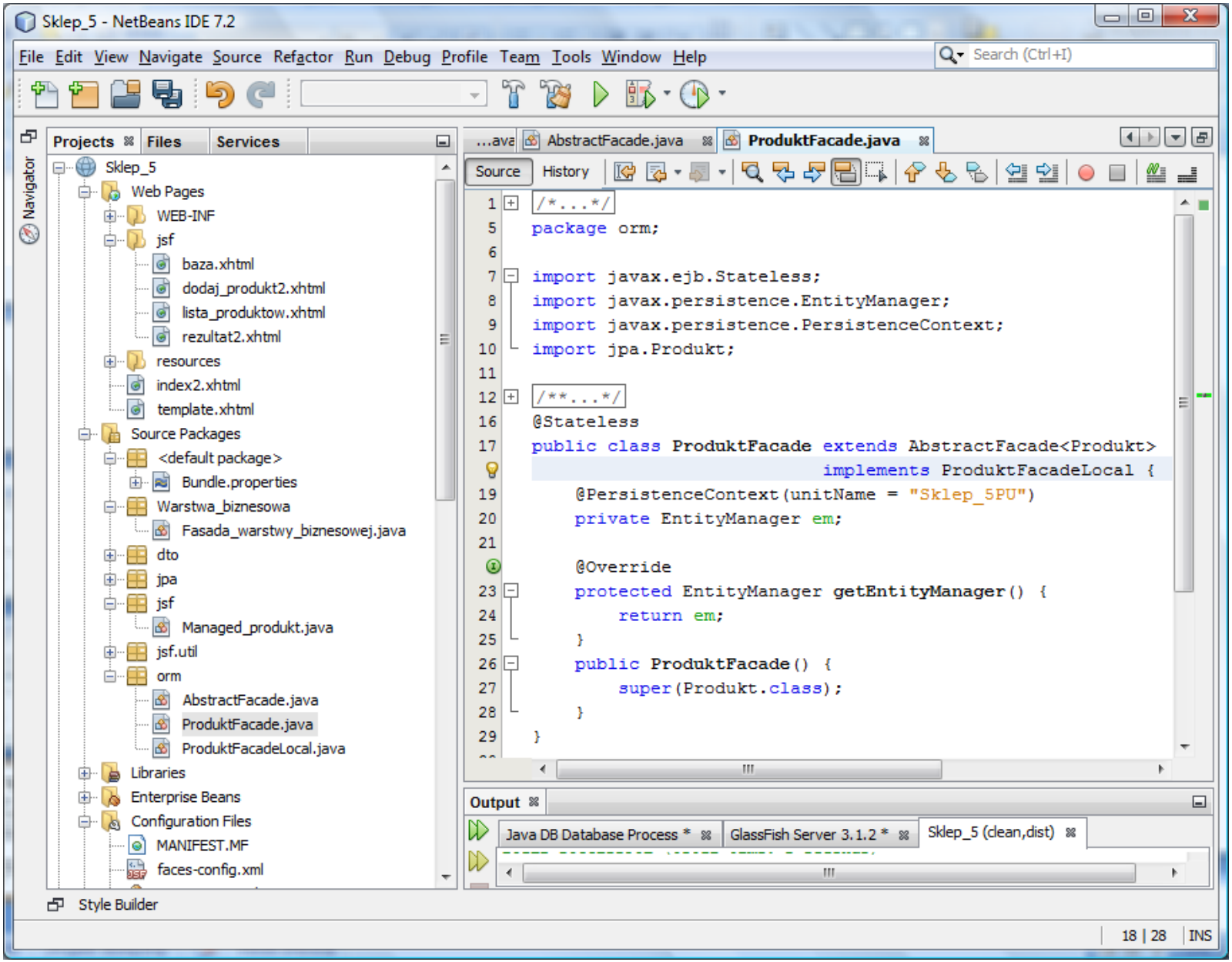
6.21. Wybór **Local** jako zasięg dla klasy typu EJB oraz wpisanie nazwy orm w polu Package



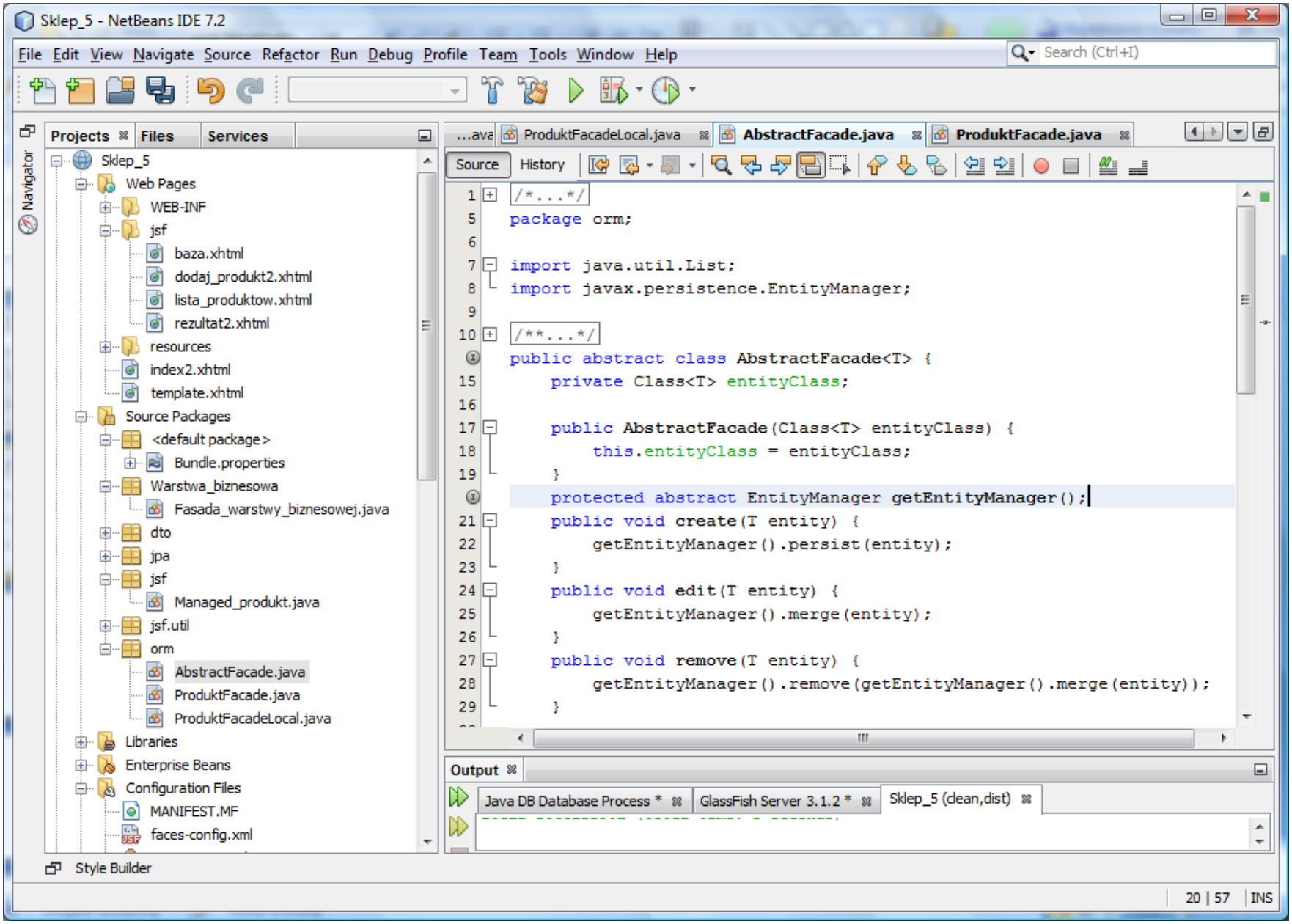
6.22. Widok wygenerowanego interfejsu **ProduktFacadeLocal**



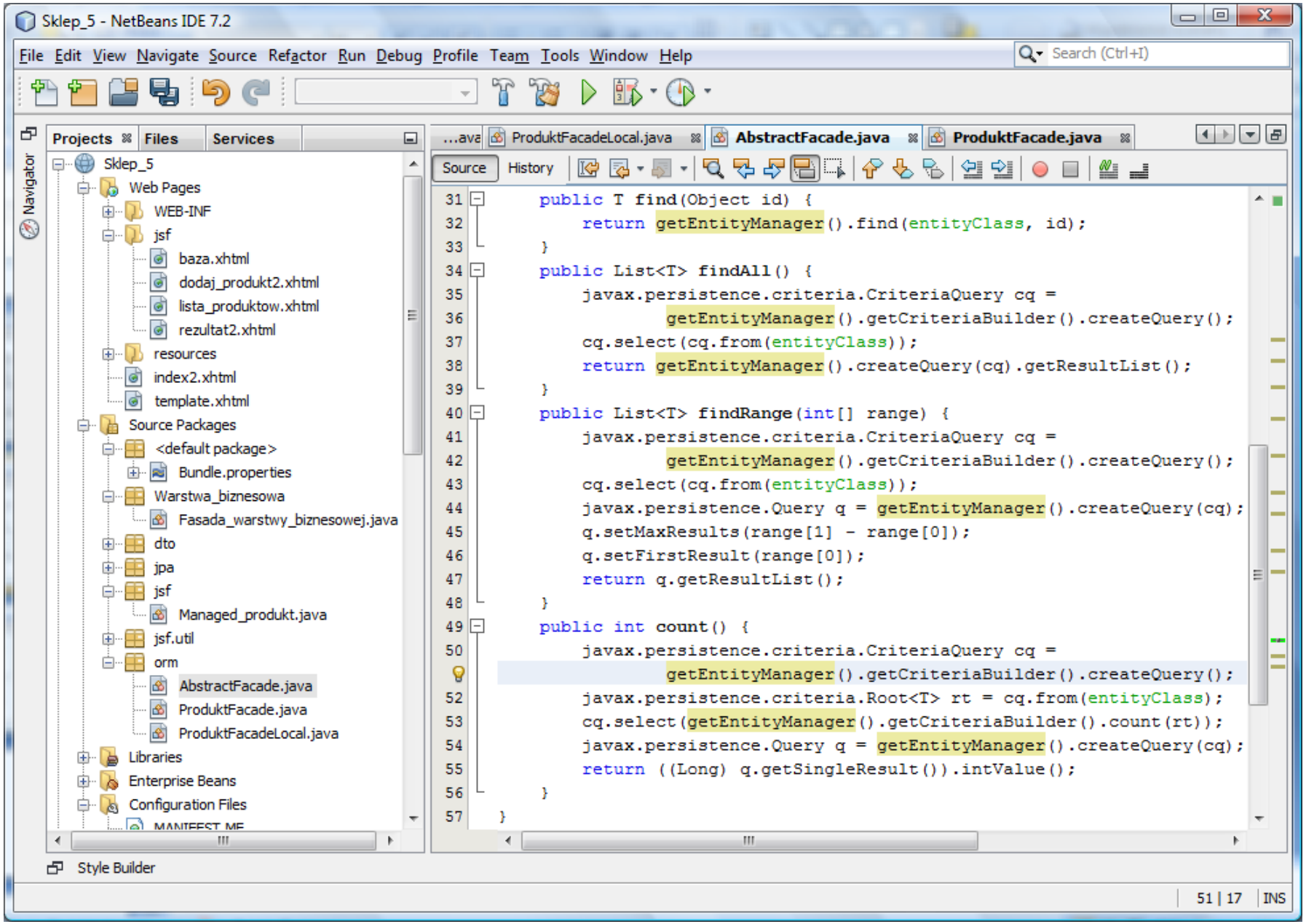
6.23. Widok kontrolera ORM **ProduktFacade**, który implementuje interfejs **ProduktFacadeLocal** i dziedziczy po klasie abstrakcyjnej **AbstractFacade**



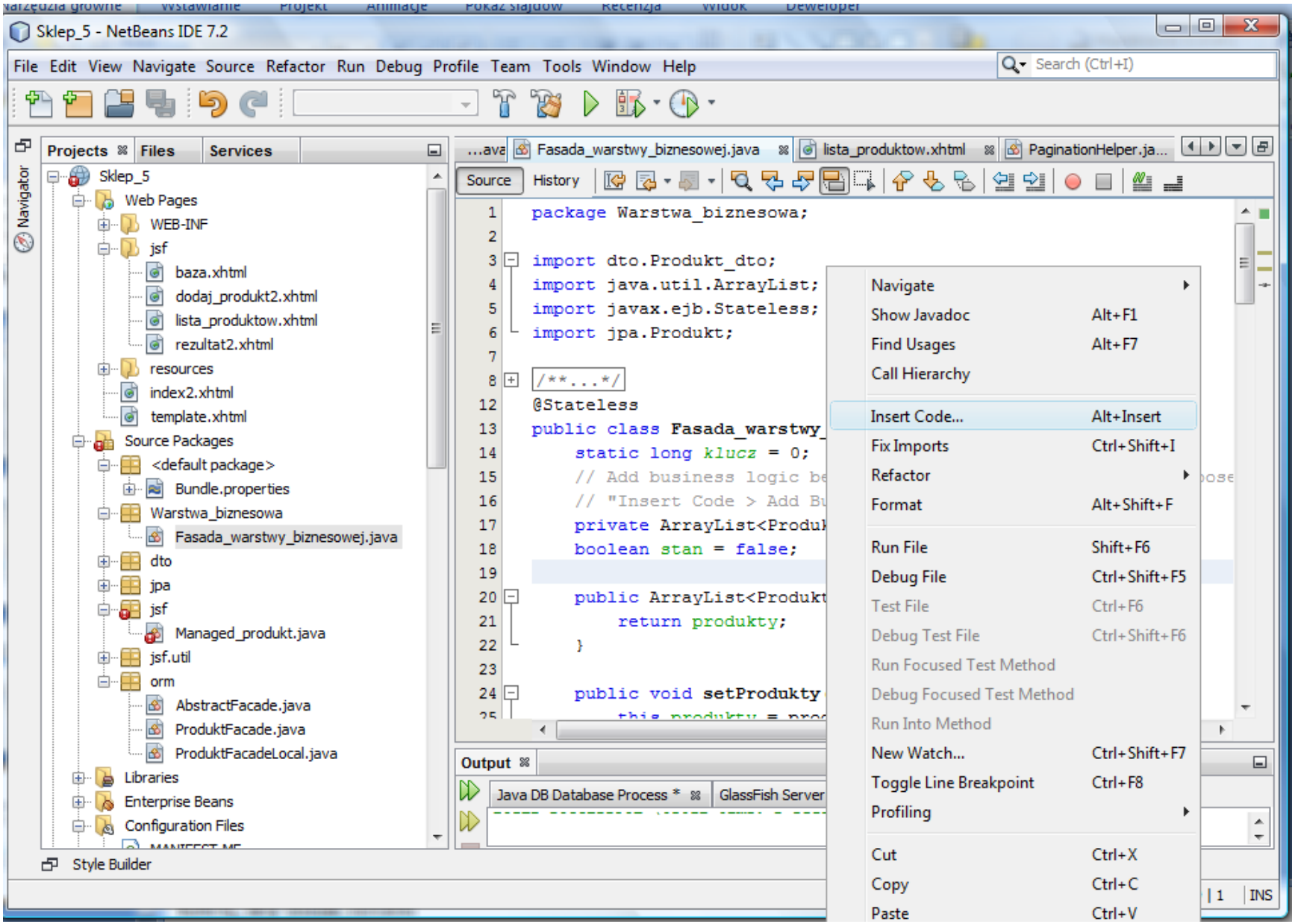
6.24. Wygenerowana klasa abstrakcyjna **AbstractFacade** (1)



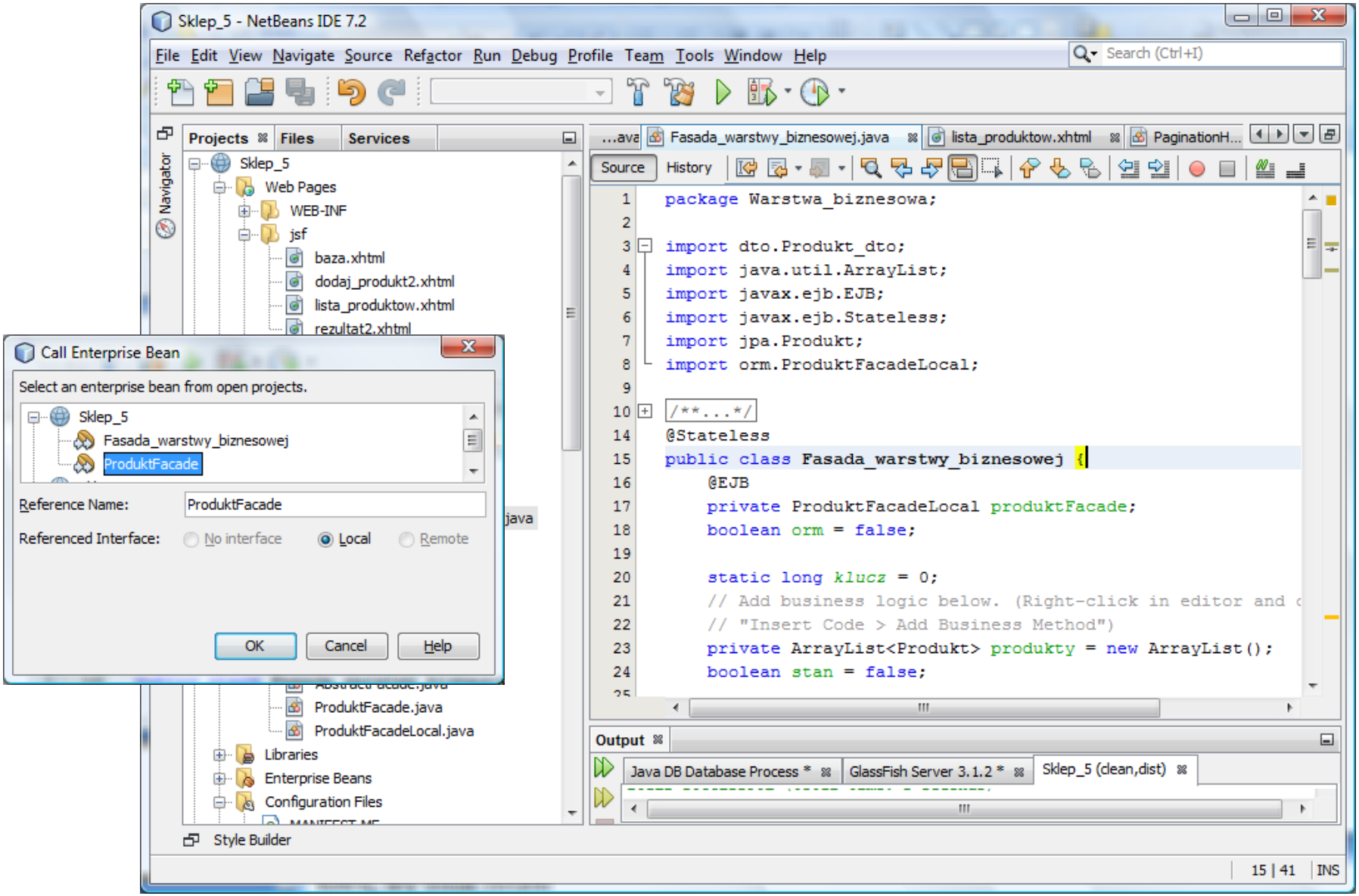
6.25. Wygenerowana klasa abstrakcyjna **AbstractFacade** (2)



6.26. Utworzenie referencji do ziarna **ProduktFacade** w klasie **Fasada_warstwy_biznesowej** (1)



6.27. Utworzenie referencji do ziarna **ProduktFacade** w klasie **Fasada_warstwy_biznesowej** (2) – **Insert Code/Call Enterprise Bean** i wybór z listy ziarna typu **ProduktFacade –** pojawiła się referencja typu **ProduktFacadeLocal**



6.28. Metody w klasie **Fasada_wartwy_biznesowej** do utrwalania danych i do pobierania danych z danych Produkt.

```
public void zapisz() {
    for (Produkt p:produkty)
        { if(produktFacade.find(p.getId())==null)
                produktFacade.create(p);
        }
}

 public void pobierz() {  //bazy usunieto w dniu 26.05.2013
        List<Produkt> pom = produktFacade.findAll();
        produkty.clear();
        for(Produkt p:pom)
            produkty.add(p);
    }
```

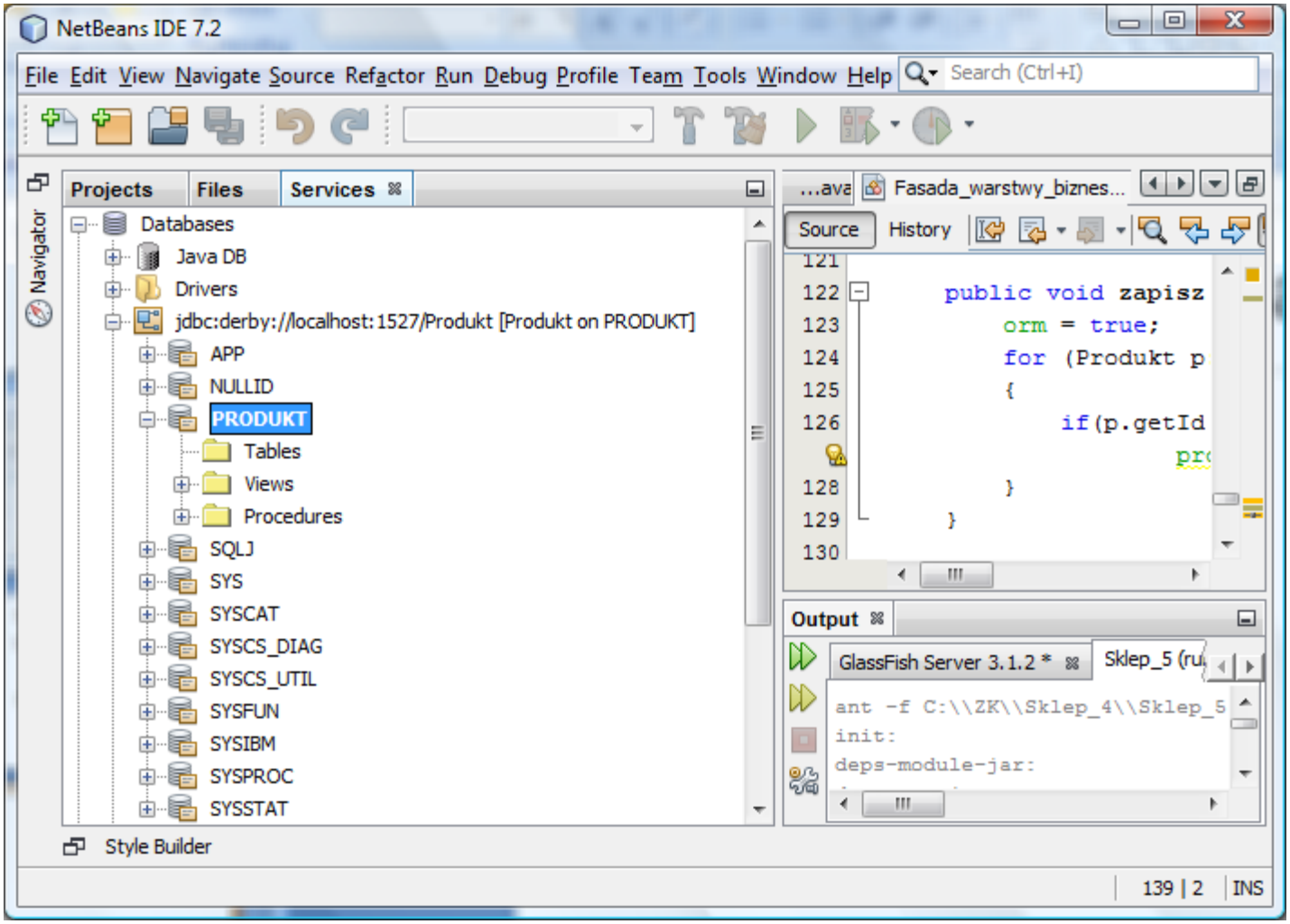
6.29. Metody w klasie **Managed_produkt**

```
public String zapisz() {
        fasada.zapisz();
        return "/faces/index2";
    }

    public String pobierz() {
        fasada.pobierz();
        getPagination().setPage(); //dodano w dniu 26.05.2013
        recreateModel();           //dodano w dniu 26.05.2013
        return "/faces/index2";
    }
```

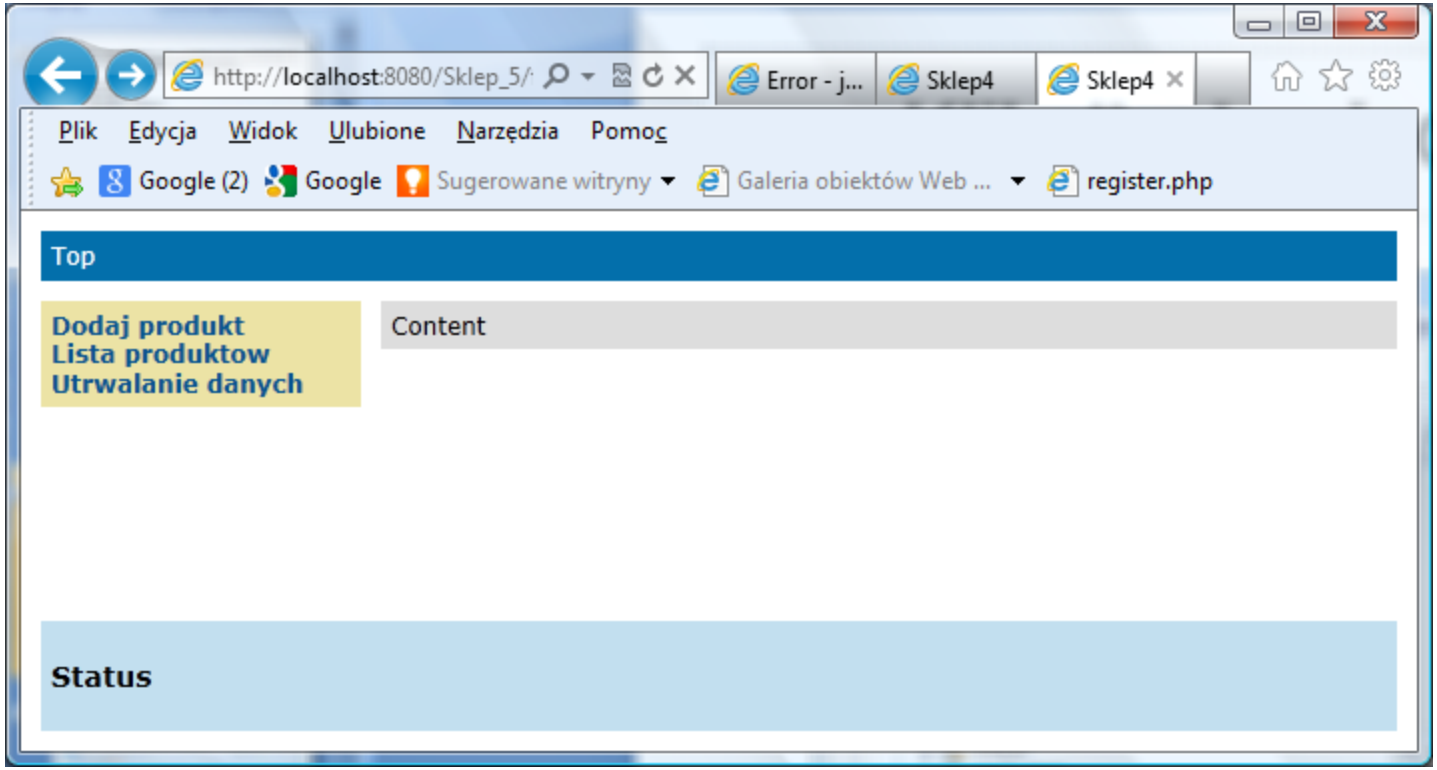
Aktualzacja liczby stron po odczytaniu danych z bazy danych

Aktualzacja modelu tabeli

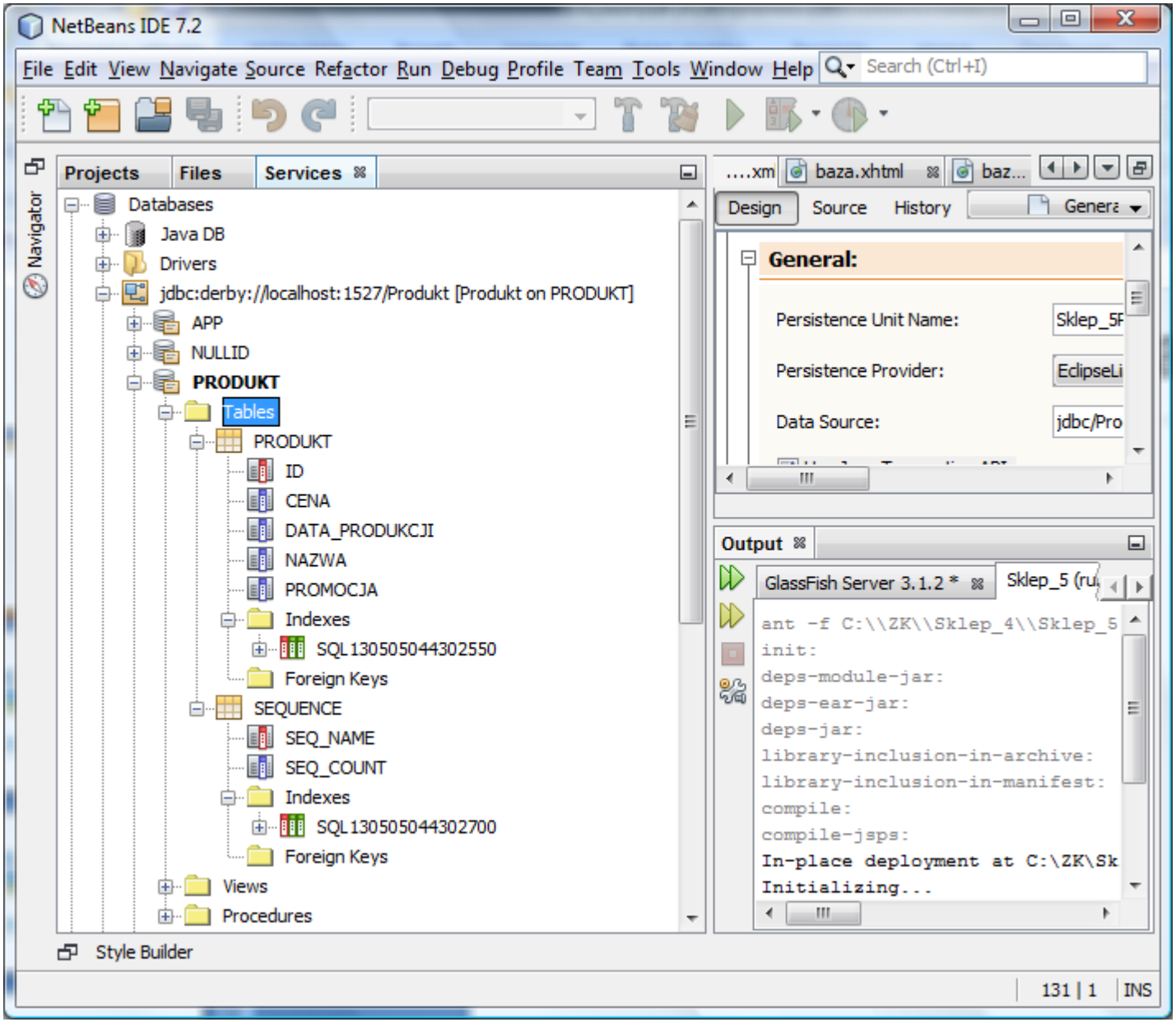
6.30 . Baza danych jest pusta przed uruchomieniem aplikacji



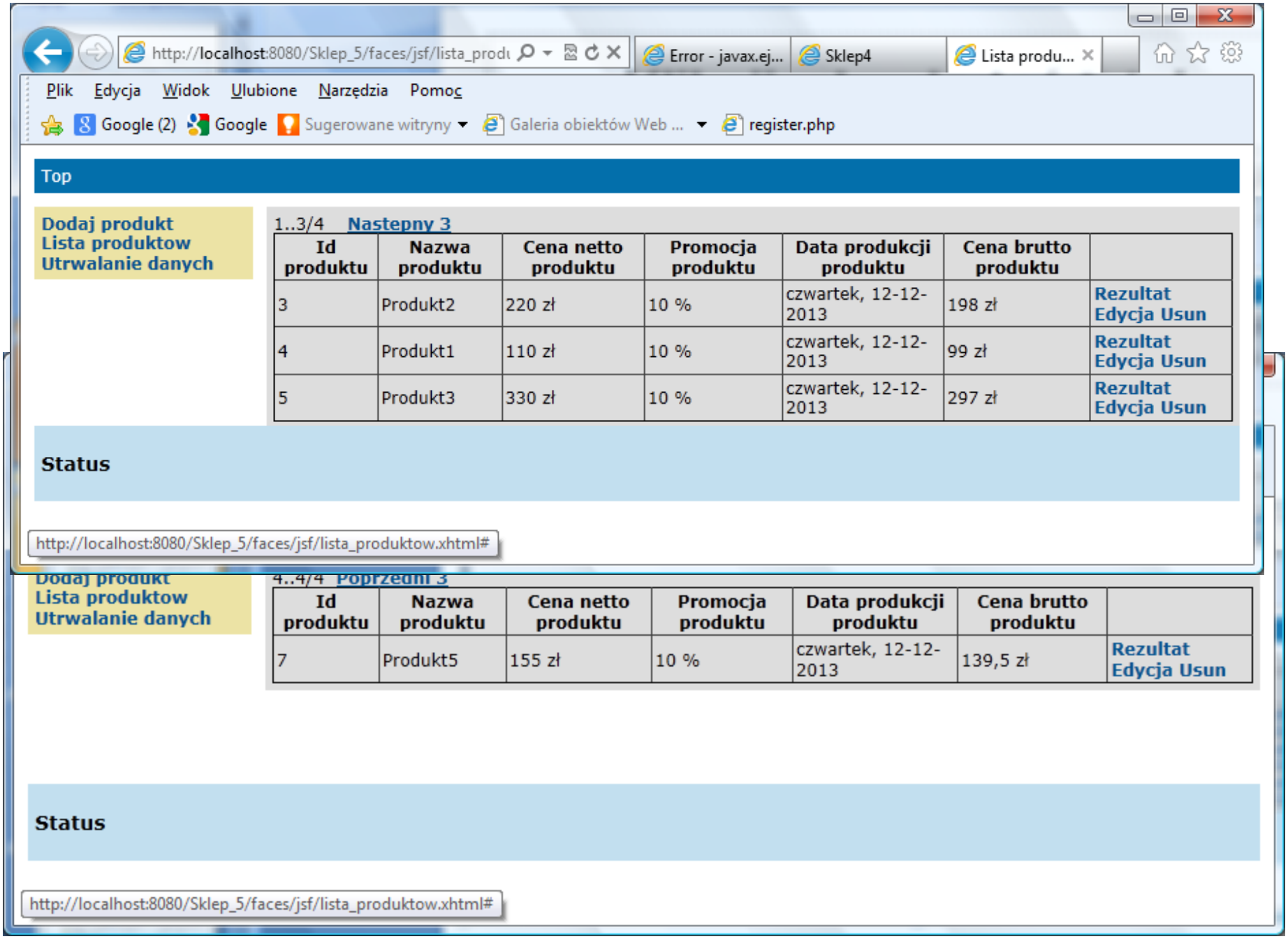
6.31. Uruchomienie aplikacji



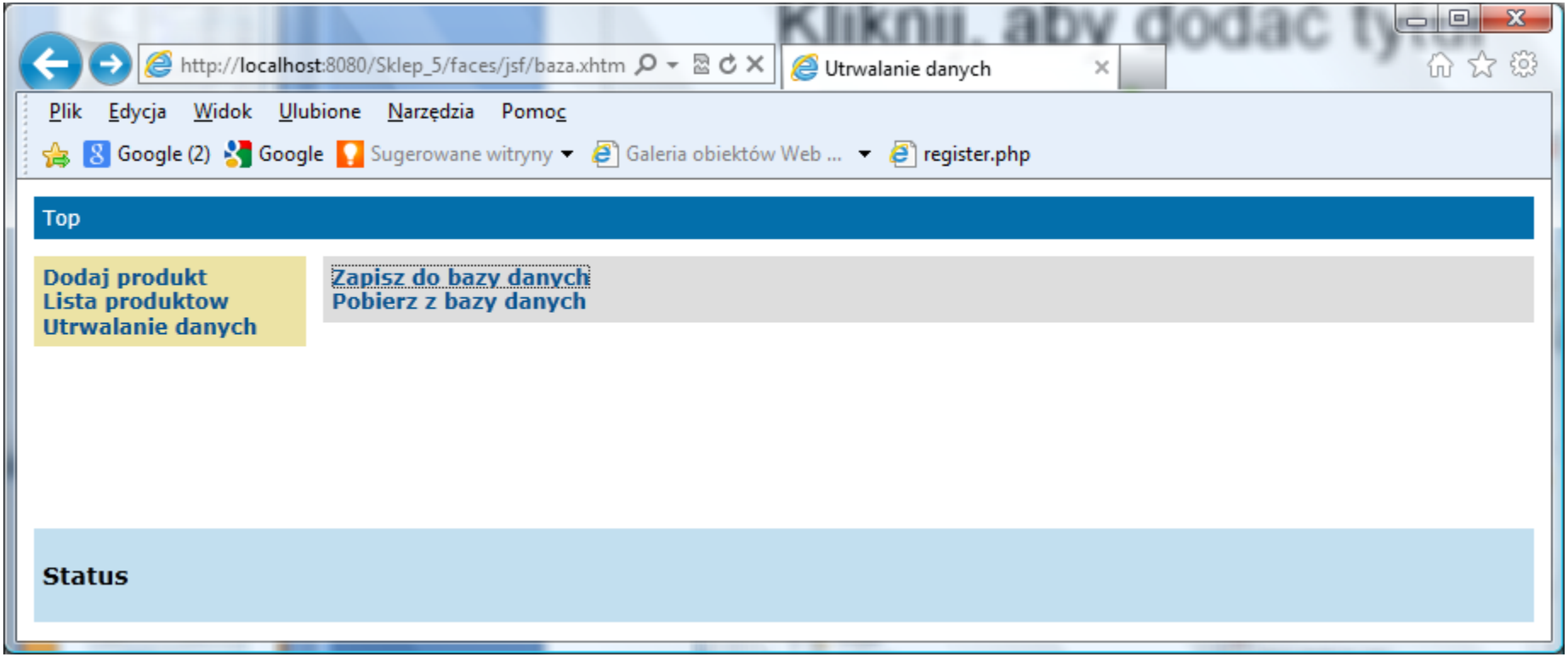
6.32. Po uruchomieniu aplikacji tworzy się schemat bazy danych.



6.33. Prezentacja działania aplikacji



6.34. Zapis do bazy danych



6.35. Prezentacja bazy danych – należy kliknąć prawym klawiszem myszy na tabelę Produkt i wybrać View data.