Budowa aplikacji wielowarstwowych. Zastosowanie konwerterów oraz plików typu properties.

Laboratorium 4 – część 1

Technologie internetowe

Zofia Kruczkiewicz

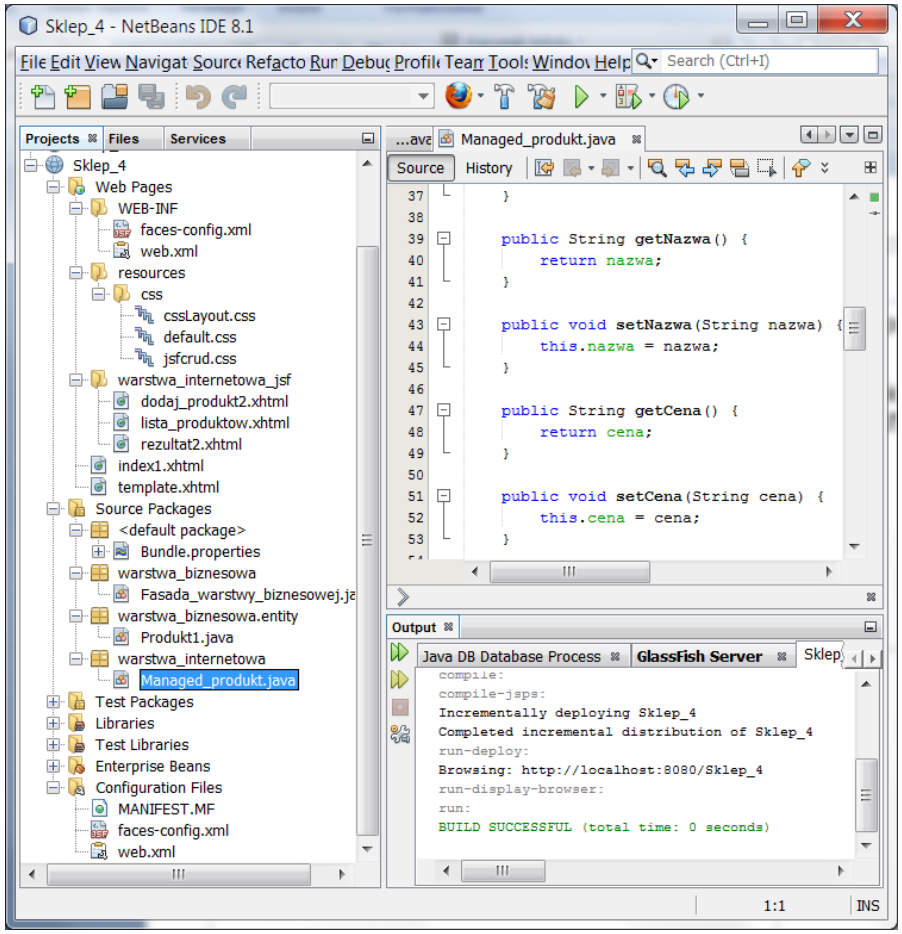
Wykaz pytań dotyczących materiału wykorzystanego w lab4, które należy opracować (m.in. wykład 5).

1. W jaki sposób można umieszczać konwertery liczb całkowitych?
2. W jaki sposób można umieszczać konwertery liczb zmiennoprzecinkowych?
3. W jaki sposób można umieszczać konwertery daty?
4. W jaki sposób ustawia się formaty danych typu data wymagane podczas wprowadzania danych?
5. W jaki sposób ustawia się formaty danych typu liczba całkowita wymagane podczas wprowadzania danych?
6. W jaki sposób ustawia się formaty danych typu liczba rzeczywista wymagane podczas wprowadzania danych?
7. W jaki sposób ustawia się formaty danych typu data wymagane podczas wyświetlania danych?
8. W jaki sposób ustawia się formaty danych typu liczba całkowita wymagane podczas wyświetlania danych?
9. W jaki sposób ustawia się formaty danych typu liczba rzeczywista wymagane podczas wyświetlania danych?
10. Jaka rolę pełni atrybut **locale** w ustalaniu formatu danych wprowadzanych na stronie lub wyświetlanych na stronie JSF?

Przykład 4
Wprowadzenie domyślnych konwerterów oraz konwertera typu convertDateTime

1. Czynności początkowe

- Należy wykonać kopię programu **Sklep_3,** wykonanego podczas lab3 jako **Sklep_4.**

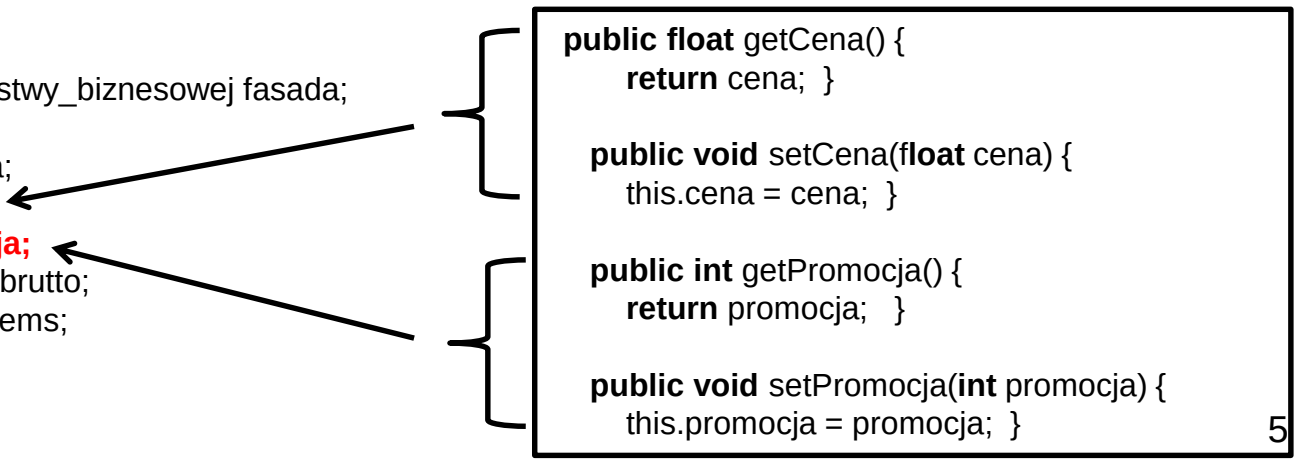


- Ustawić kodowanie UTF-8; po zaznaczeniu nazwy projektu w oknie **Projects** prawym klawiszem myszy wybrać pozycję **Properties/Sources/Encoding/UTF-8**

2. Dodanie konwerterów domyślnych

Należy zmodyfikować typy metod dostępu do danych ceny i promocji w klasie **Managed_produkt** oraz należy wstawić dodatkowo datę zakupu jako kolejny atrybut klasy **Produkt1,** a więc kolejną pozycję wprowadzaną i prezentowaną na stronach JSF. Należy zmodyfikować metody get i set nowych pól: **cena, promocja** za pomocą **Insert Code….** - służą one do obsługi pobierania danych i wyświetlania danych z wykorzystaniem domyślnnych konwerterów liczbowych na stronach JSF: **dodaj_produkt2.xhtml** oraz **rezultat.xhtml**

```
package warstwa_internetowa;

import javax.ejb.EJB;
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import warstwa_biznesowa.Fasada_warstwy_biznesowej;

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

    @EJB
    private Fasada_warstwy_biznesowej fasada;

    private String nazwa;
    private float cena;
    private int promocja;
    private String cena_brutto;
    private DataModel items;
    private int stan = 1;
```



2. Dodanie konwerterów domyślnych (cd)

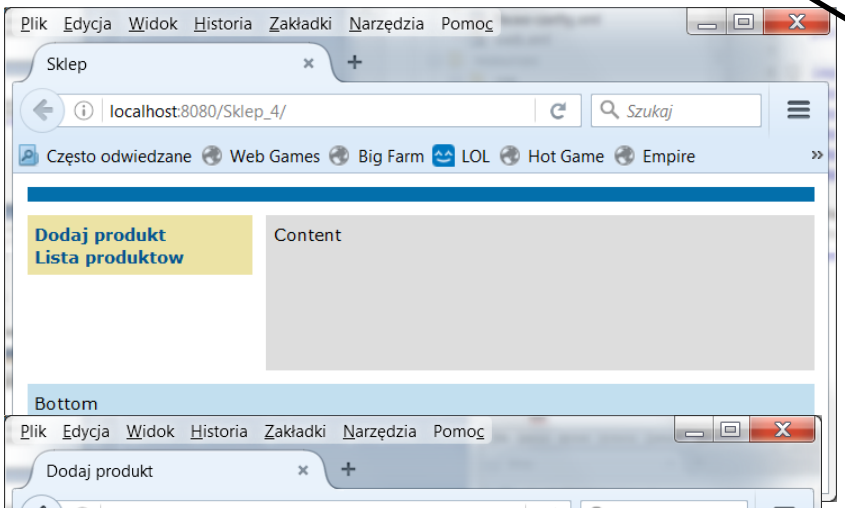
Należy zmodyfikować kod metod **dodaj_produkt()** i **dane_produktu()**

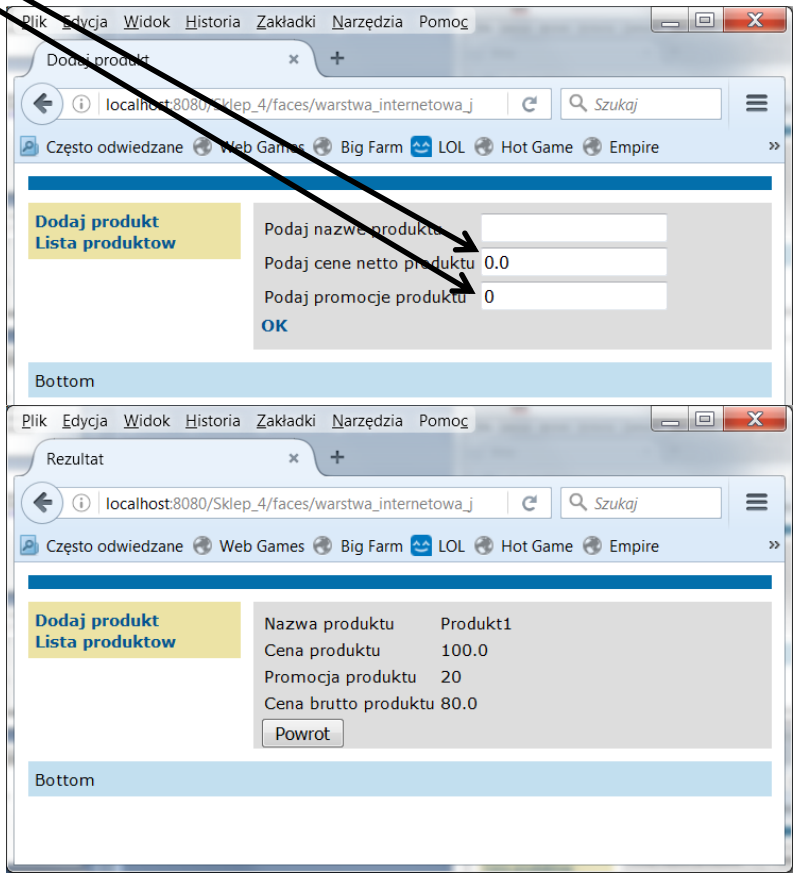
```
public String dodaj_produkt() {
        String[] dane = {nazwa, ""+cena, ""+promocja};
        fasada.utworz_produkt(dane);
        dane_produktu();
        return "rezultat2";
    }

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

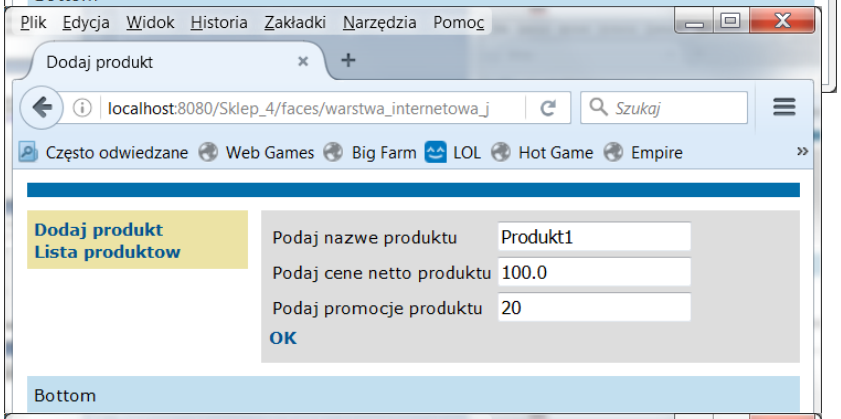
Konwersja typu **float** i **int** do typu **String**

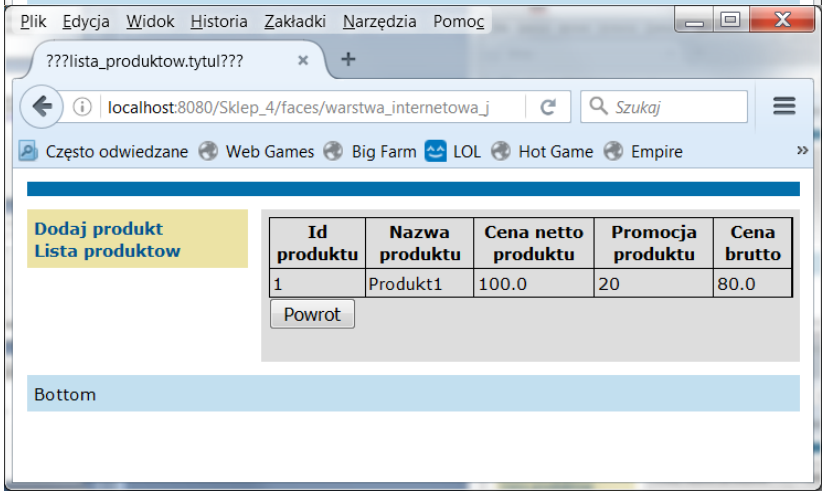
Konwersja typu **String** do typu **float** i **int**

3. Dodanie konwerterów domyślnych – po uruchomieniu programu na stronie **dodaj_produkt2.xhtml** pojawia się format danych typu **int i float** w polach do wprowadzania ceny i promocji.



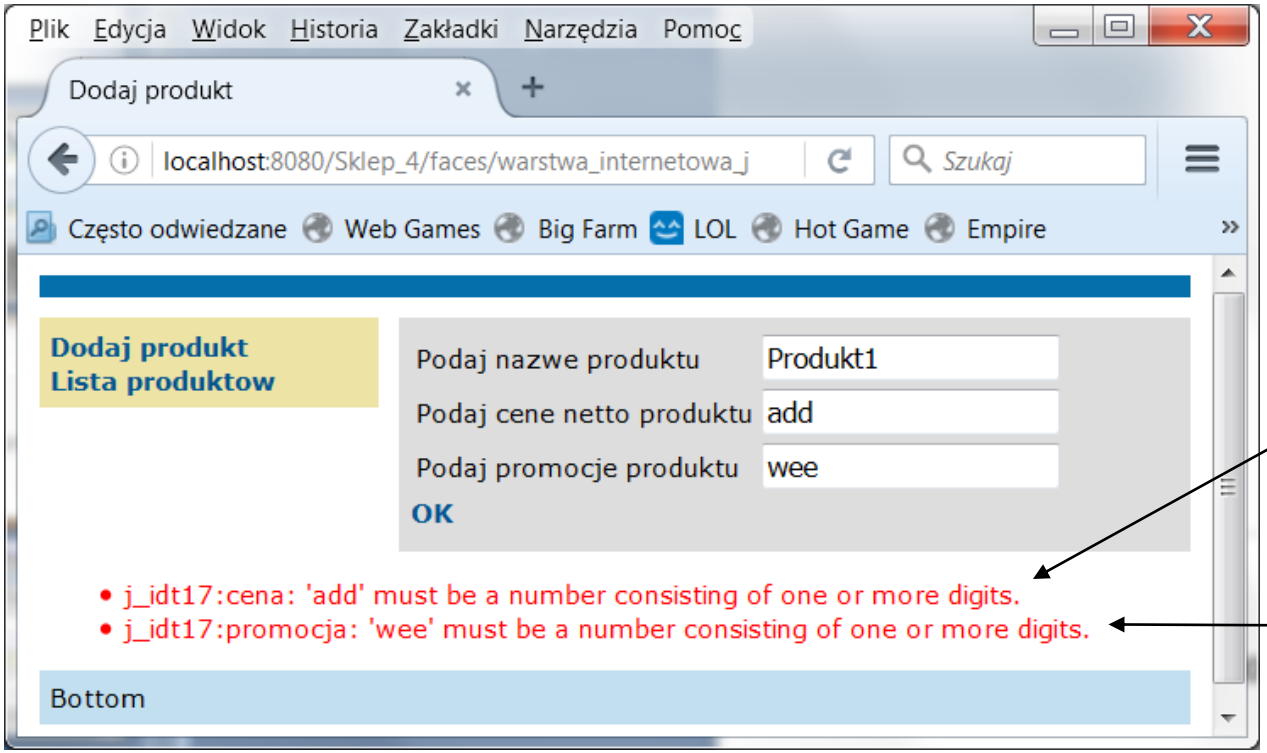






Format wyświetlanych danych obecnie wynika ze zmiany formatu danych na stronie **dodaj_produkt2.xhtml** oraz **rezultat2.xhtml** w klasie **Managed_produkt**, a wcześniej z konwersji typu **float** i **int** na typu String przez metodę **dane_produktu()** klasy **Fasada_warstwy_biznesowej,** wywołanej w metodzie klasy **Managed_produkt: dane_produktu.** Nadal wyświetlanie danych na stronie **lista_produktów.xhtml** wynika jedynie z konwersji danych w metodzie **items()** w klasie **Fasada_warstwy_biznesowej,** wywołanej w metodzie **utworz_DataModel()** w klasie **Managed_produkt**.

Efekt działania domyślnych konwerterów w przypadku wprowadzenia niepoprawnych danych



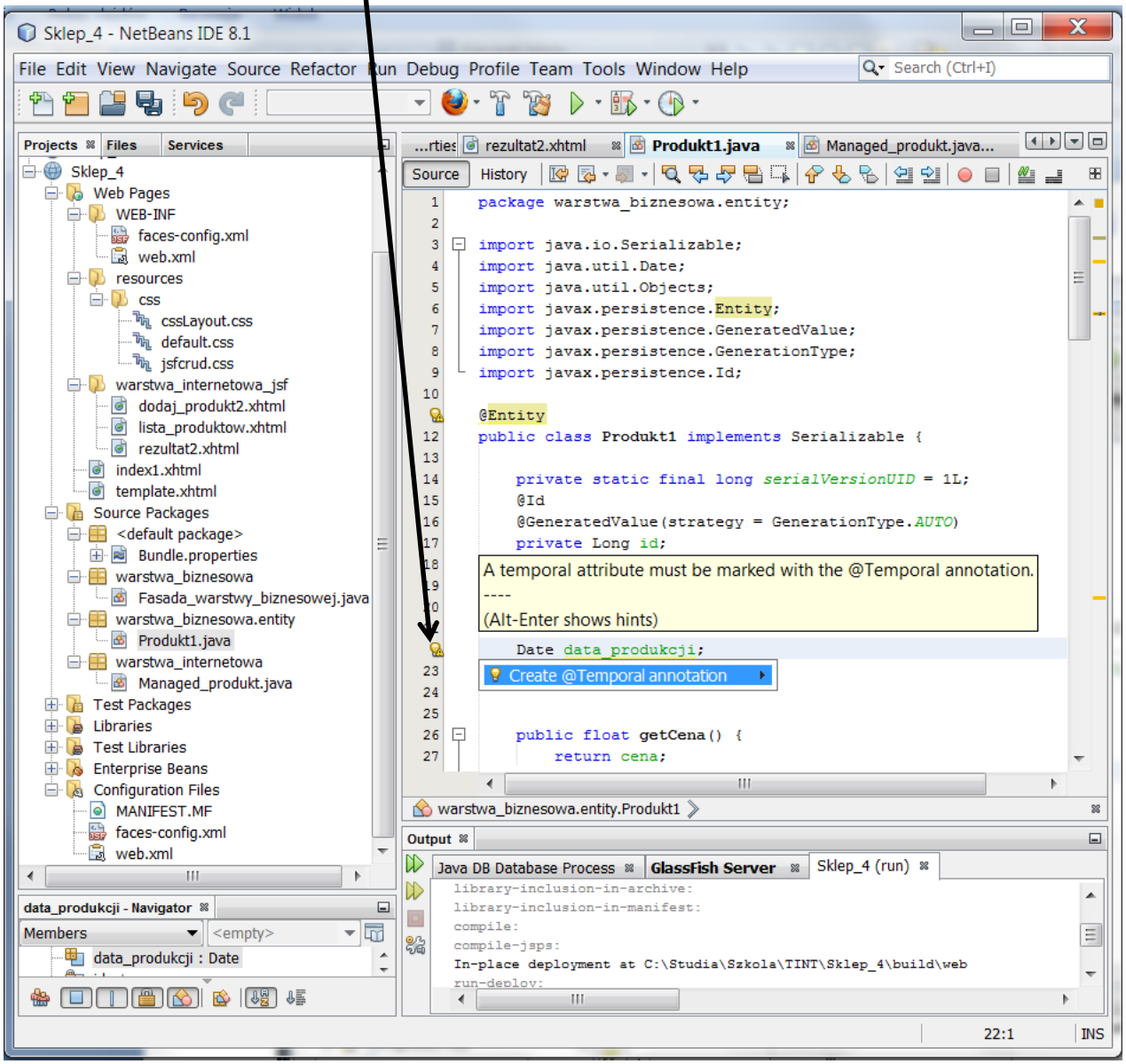
Efekt działania konwertera domyślnego typu **Float**

Efekt działania konwertera domyślnego typu **Integer**

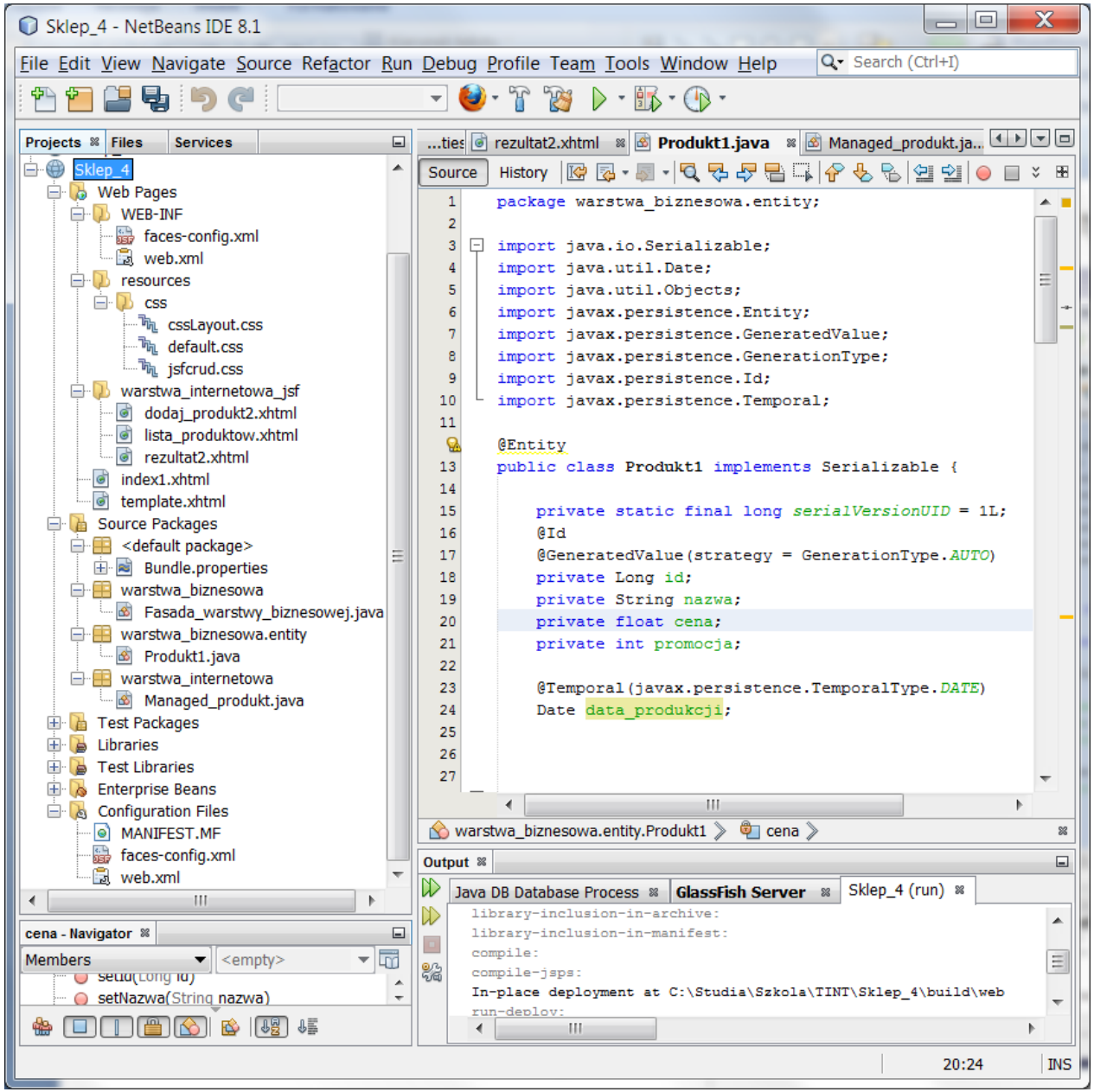
4. Należy podobnie zmodyfikować typ atrybutu **cena_brutto** na typ **float**, podobnie jak atrybut **cena** obiektu typu **Managed_produkt.**

5. Modyfikacja kodu klasy **Produkt1** – dodanie **atrybutu data_produkcji** typu **Date**. W celu dodania importu klasy **Date** należy wykonać **Fix Imports**.

W klasie typu Entity należy dodać sugerowaną adnotację do atrybutu typu **Date** klikając lewym klawiszem myszy na żółtą kropkę z lewej strony linii atrybutu, a następnie na **Create @Temporal annotation**



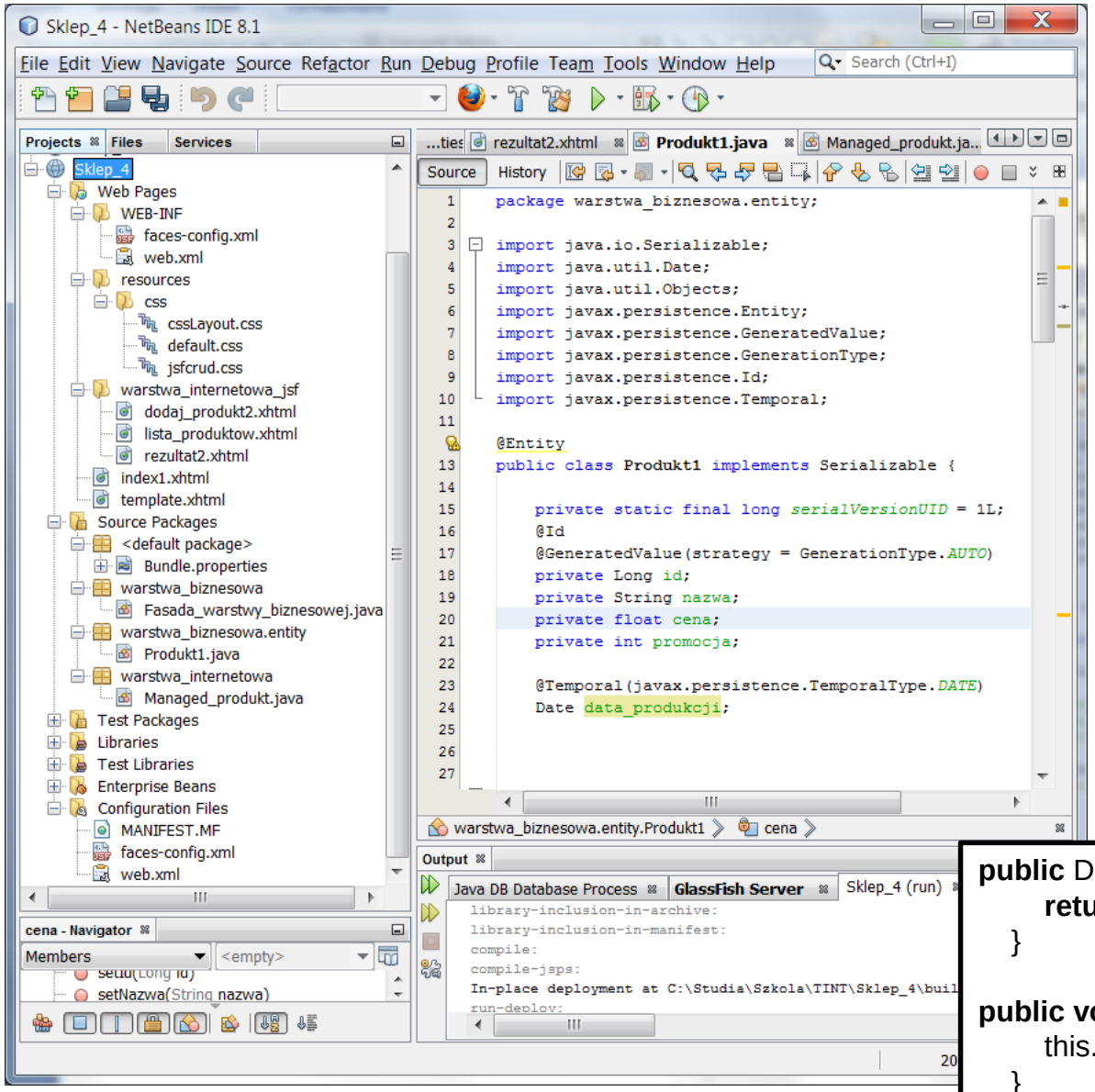
Dodanie sugerowanej adnotacji do atrybutu **data_produkcji** typu **Date** - dodana sugerowana adnotacja **@Temporal(javax.persistence.TemporalType.DATE)** do atrybutu **data_produkcji** typu **Date**

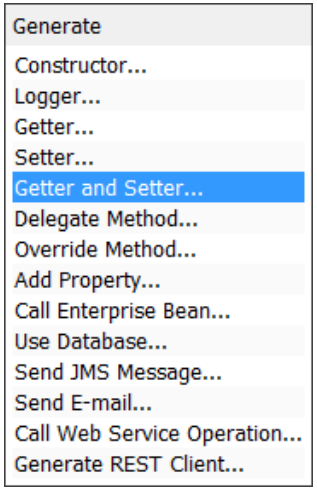


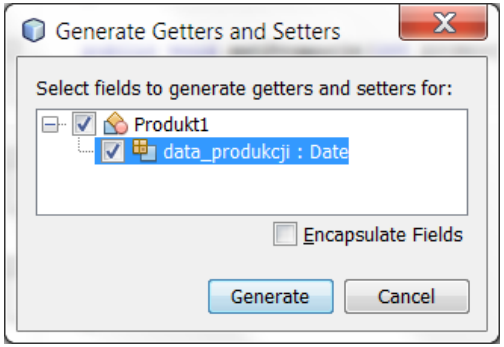
Fragment kodu klasy **Produkt** po dodaniu nowego atrybutu
private Date data_produkcji;

```
package warstwa_biznesowa.entity;

import  java.io.Serializable;
Import  java.util.Date;
Import  javax.persistence.Entity;
import  javax.persistence.GeneratedValue;
Import  javax.persistence.GenerationType;
import  javax.persistence.Id;
Import  javax.persistence.Temporal;

@Entity
public class Produkt1 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;
    @Temporal(javax.persistence.TemporalType.DATE)
    private Date data_produkcji;
```

Dodanie metod dostępu typu set i get do atrybutu **data_produkcji** typu **Date** – należy prawym klawiszem myszy kliknąć na powierzchnię edytora i wybrać **Insert Code…** i następnie wybrać **Setter and Getter**… i na kolejnym formularzu zaznaczyć atrybut **data_produkcji** i kliknąć na **Generate.**








```
public Date getData_produkcji() {
        return data_produkcji;
    }

public void setData_produkcji(Date data_produkcji) {
        this.data_produkcji = data_produkcji;
    }
```

6. Modyfikacja kodu klasy **Fasada_warstwy_biznesowej**, wynikająca z dodania nowego atrybutu typu **Date** do klasy **Produkt1**

```
package warstwa_biznesowa;

import java.util.LinkedList;
import java.util.Date;
import javax.ejb.Stateless;
import warstwa_biznesowa.entity.Produkt1;

@Stateless
public class Fasada_warstwy_biznesowej {

    static long klucz = 0;
    private LinkedList<Produkt1> produkty = new LinkedList();
    boolean stan = false;

   public LinkedList<Produkt1> getProdukty()                    {    return produkty;  }

   public void setProdukty(LinkedList<Produkt1> produkty)       {    this.produkty = produkty;    }

   public void utworz_produkt(String dane[], Date data) {
        Produkt1 produkt = new Produkt1();
        klucz++;
        produkt.setId(new Long(klucz));
        produkt.setNazwa(dane[0]);
        produkt.setCena(Float.parseFloat(dane[1]));
        produkt.setPromocja(Integer.parseInt(dane[2]));
        produkt.setData_produkcji(data);
        dodaj_produkt(produkt);
    }
```

Metoda zmodyfikowana - po wywołaniu w warstwie internetowej otrzymuje dane produktu jako parametr **dane** oraz datę produkcji w parametrze **data** w liście parametrów

```
public String[] dane_produktu() {
    if (stan) {
        Produkt1 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 data = ""+produkt.getData_produkcji().getTime();
        String dane[] = {nazwa, cena, promocja, cena_brutto, data};
        return dane;
     }
    return null;
}
```

Metoda **getTime** zwraca liczbę milisekund od stycznia 1, 1970, 00:00:00 GMT reprezentowaną przez obiekt typu **Date**.

Metoda zmodyfikowana - po wywołaniu w warstwie internetowej zwraca dane produktu w wyniku zwracanym przez metodę, jako tablicę 5 obiektów typu String

```
public ArrayList<ArrayList<String>> items() {
    ArrayList<ArrayList<String>> dane = new ArrayList();
    for (Produkt1 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.getData_produkcji().toString());
        wiersz.add("" + p.cena_brutto());
        dane.add(wiersz);
    }
    return dane;
```

Konwersja daty na następującą postać znakową: **dzień_tygodnia miesiąc numer_dnia_miesiąca gg:mm:ss strefa_czasowa rok** np:
2-11-2016 na:
Wed Nov 02 01:00:00 CET 2016

Metoda zmodyfikowana - po wywołaniu w warstwie internetowej przekazuje przez return dane wszystkich produktów, jako kolekcji elementów typu kolekcja 5 elementów typu String, każdy zawierający dane produktu. Te dane stanowią model danych typu **DataModel**, wyświetlanych w komponencie **dataTable**

7. Zmodyfikowany kod klasy **Managed_produkt** – dodanie atrybutu **data_produkcji** typu **Date.** Należy dodać metody get i set nowego atrybutu **data_produkcji** za pomocą **Insert Code…**

```
package warstwa_internetowa;

import java.util.Date;
import javax.ejb.EJB;
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import warstwa_biznesowa.Fasada_warstwy_biznesowej;

@Named(value = "managed_produkt")
@RequestScoped
public class Managed_produkt {
    @EJB
    private Fasada_warstwy_biznesowej fasada;
    private String nazwa;
    private float cena;
    private int promocja;
    private String cena_brutto;
    private DataModel items;
    private int stan = 1;
    private Date data_produkcji;
```

```
public Date getData_produkcji() {
    return data_produkcji;
  }

public void setData_produkcji(Date data_produkcji) {
    this.data_produkcji = data_produkcji;
  }

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

public void dane_produktu() {
    stan = 1;
    String[] dane = fasada.dane_produktu();
    if (dane == null) {
       stan = 0;
    } else {
       nazwa = dane[0];
       cena = Float.parseFloat(dane[1]);
       promocja = Integer.parseInt(dane[2]);
       cena_brutto = dane[3];
       data_produkcji.setTime(Long.parseLong(dane[4]));
    }
  }
```

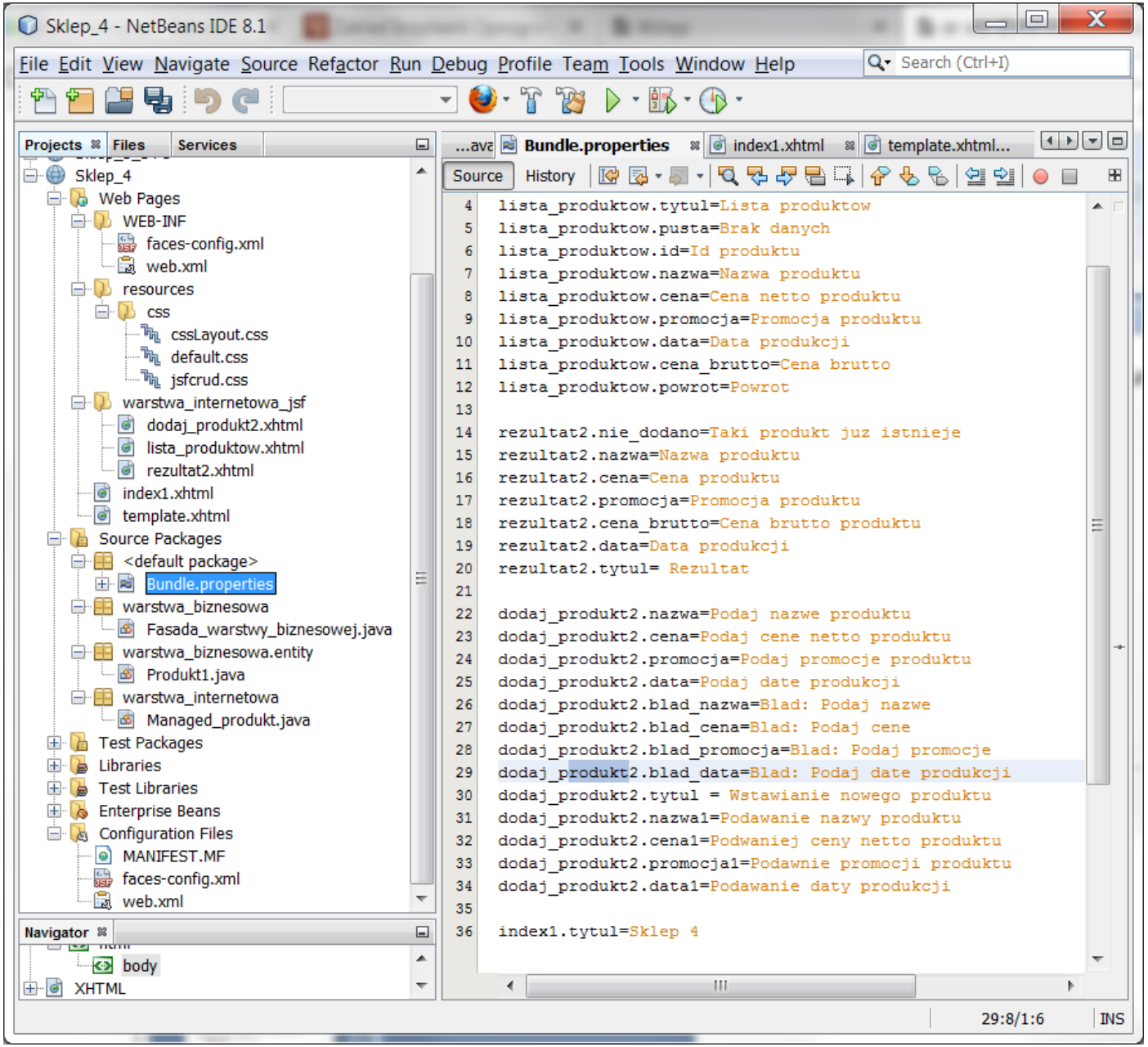
Metoda klasy typu **Fasada_warstwy_biznesowej** po wywołaniu w warstwie internetowej (obiekt klasy typu **Managed_produkt**) otrzymuje dane produktu przez listę parametrów:

- w tablicy **dane**: nazwę, cenę oraz promocję jako dane atrybutów: **nazwa, cena, promocja** obiektu typu **Produkt1**
- oraz parametr **data,** jako wartość atrybutu **data_produkcji** obiektu typu **Produkt1**

Metoda klasy typu **Fasada_warstwy_biznesowej** po wywołaniu w warstwie internetowej zwraca dane produktu jako tablicę 5 obiektów typu String

Metoda **setTime** ustawia aktualną datę w obiekcie typu **Date** na podstawie liczby millisekund, jakie upłynęły od stycznia 1, 1970, 00:00:00 GMT, podanych w liście parametrów metody.

8. Modyfikacja i uzupełnienie komunikatów w pliku **Bundle.properties – przykładowy fragment zawartości pliku**



9. Modyfikacja komunikatu w **pliku index1.xhtml** – pobranie z pliku **Bundle.properties**, w którym dodano komunikat:
index1.tytul=Sklep 4

```
<?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">
            <h:outputText
               value="#{bundle['index1.tytul']}">
            </h:outputText>
          </ui:define>

       </ui:composition>

 </body>
</html>
```

10. Modyfikacja pliku **dodaj_produkt2.xhtml** – komunikaty z pliku **Bundle.properties** oraz dodanie konwertera typu **convertDateTime**

```
<?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['dodaj_produkt2.tytul']}">
          </h:outputText>
        </ui:define>
        <ui:define name="content">
           <h:form>
              <h:panelGrid columns="2">
                 <h:outputLabel value="#{bundle['dodaj_produkt2.nazwa']}"  for="nazwa" />
                 <h:inputText
                    id="nazwa"       title="#{bundle['dodaj_produkt2.nazwa1']}"
                    value="#{managed_produkt.nazwa}"
                    required="true"
                    requiredMessage="#{bundle['dodaj_produkt2.blad_nazwa']}" >
                 </h:inputText>
                 <h:outputLabel value="#{bundle['dodaj_produkt2.cena']}" for="cena" />
                 <h:inputText
                    id="cena"   title="#{bundle['dodaj_produkt2.cena1']}"
                    value="#{managed_produkt.cena}"
                    required="true"
                    requiredMessage="#{bundle['dodaj_produkt2.blad_cena']}" >
                 </h:inputText>
```

10. (cd) Modyfikacja pliku **dodaj_produkt2.xhtml** – komunikaty z pliku Bundle.properties oraz dodanie konwertera typu **convertDateTime**

```
            <h:outputLabel value="#{bundle['dodaj_produkt2.promocja']}" for="promocja" />
            <h:inputText
               id="promocja"
               title="#{bundle['dodaj_produkt2.promocja1']}"
               value="#{managed_produkt.promocja}"
               required="true"
               requiredMessage="#{bundle['dodaj_produkt2.blad_promocja']}" >
            </h:inputText>
            <h:outputLabel value="#{bundle['dodaj_produkt2.data']}"  for="data" />
            <h:inputText
               id="data"
               title="#{bundle['dodaj_produkt2.data1']}"
               value="#{managed_produkt.data_produkcji}"
               required="true"
               requiredMessage="#{bundle['dodaj_produkt2.blad_data']}">
                  <f:convertDateTime pattern="dd-MM-yyyy" />
            </h:inputText>

         </h:panelGrid>
         <h:commandLink  action="#{managed_produkt.dodaj_produkt}" value="OK" />
      </h:form>
   </ui:define>
</ui:composition>

</body>
</html>
```

11. Modyfikacja pliku **rezultat2.xhtml** – komunikaty z pliku Bundle.properties oraz dodanie konwertera typu **convertDateTime**

```
<?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['rezultat2.tytul']}">
                 </h:outputText>
            </ui:define>
            <ui:define name="content">
                <h:form>
                    <h:outputText escape="false" value="#{bundle['rezultat2.nie_dodano']}"
                                  rendered="#{managed_produkt.stan==0}"/>
                    <h:panelGrid columns="2" rendered="#{managed_produkt.stan!=0}">
                        <h:outputLabel value="#{bundle['rezultat2.nazwa']}"  for="nazwa" />
                        <h:outputText id="nazwa" value="#{managed_produkt.nazwa}"/>
                        <h:outputLabel value="#{bundle['rezultat2.cena']}"  for="cena" />
                        <h:outputText id="cena" value="#{managed_produkt.cena}"/>
                        <h:outputLabel value="#{bundle['rezultat2.promocja']}"  for="promocja" />
                        <h:outputText id="promocja" value="#{managed_produkt.promocja}"/>
```

11 (cd) Modyfikacja pliku **rezultat2.xhtml** – komunikaty z pliku **Bundle.properties** oraz dodanie konwertera typu **convertDateTime**

```
            <h:outputLabel value="#{bundle['rezultat2.data']}" for="data"/>

            <h:outputText id="data" value="#{managed_produkt.data_produkcji}">
                <f:convertDateTime pattern="dd-MM-yyyy" />
             </h:outputText>

             <h:outputLabel value="#{bundle['rezultat2.cena_brutto']}"  for="brutto" />
             <h:outputText id="brutto" value="#{managed_produkt.cena_brutto}" />
          </h:panelGrid>
          <h:commandButton id="powrot" value="#{bundle['lista_produktow.powrot']}"
                                action="/faces/index1"/>
        </h:form>
      </ui:define>
    </ui:composition>
  </body>
</html>
```

12. Modyfikacja pliku **lista_produktow.xhtml** – komunikaty z pliku **Bundle.properties**, sposób korzystania z modelu **items** komponentu **dataTable** oraz wyświetlenie daty produkcji

```
<?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_produktow.tytul']}"></h:outputText>
         </ui:define>
         <ui:define name="content">
            <h:form styleClass="jsfcrud_list_form">
               <h:outputText escape="false" value="#{bundle['lista_produktow.pusta']}"
                        rendered="#{managed_produkt.items.rowCount == 0}"/>
               <h:panelGroup rendered="#{managed_produkt.items.rowCount > 0}">
                  <h:dataTable value="#{managed_produkt.items}" var="item" border="0"
                               cellpadding="2" cellspacing="0"
                               rowClasses="jsfcrud_odd_row,jsfcrud_even_row"
                               rules="all" style="border:solid 1px">
```

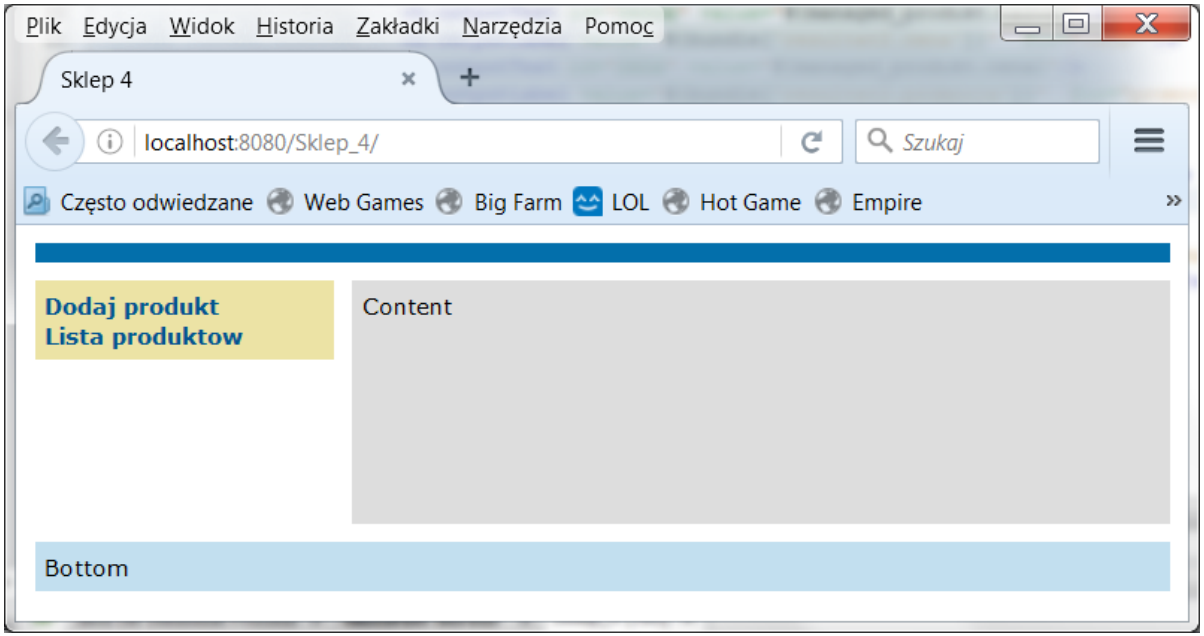
12 (cd). Modyfikacja pliku **lista_produktow.xhtml** – komunikaty z pliku Bundle.properties, sposób korzystania z modelu **items** komponentu **dataTable** oraz wyświetlenie daty produkcji

```
<h:column>
        <f:facet name="header">
           <h:outputText value="#{bundle['lista_produktow.id']}"/>
        </f:facet>
        <h:outputText value="#{item.get(0)}"/>
  </h:column>
  <h:column>
        <f:facet name="header">
           <h:outputText value="#{bundle['lista_produktow.nazwa']}"/>
        </f:facet>
        <h:outputText value="#{item.get(1)}"/>
   </h:column>
    <h:column>
        <f:facet name="header">
           <h:outputText value="#{bundle['lista_produktow.cena']}"/>
        </f:facet>
        <h:outputText value="#{item.get(2)}"/>
     </h:column>
     <h:column>
        <f:facet name="header">
           <h:outputText value="#{bundle['lista_produktow.promocja']}"/>
        </f:facet>
        <h:outputText value="#{item.get(3)}"/>
       </h:column>
```

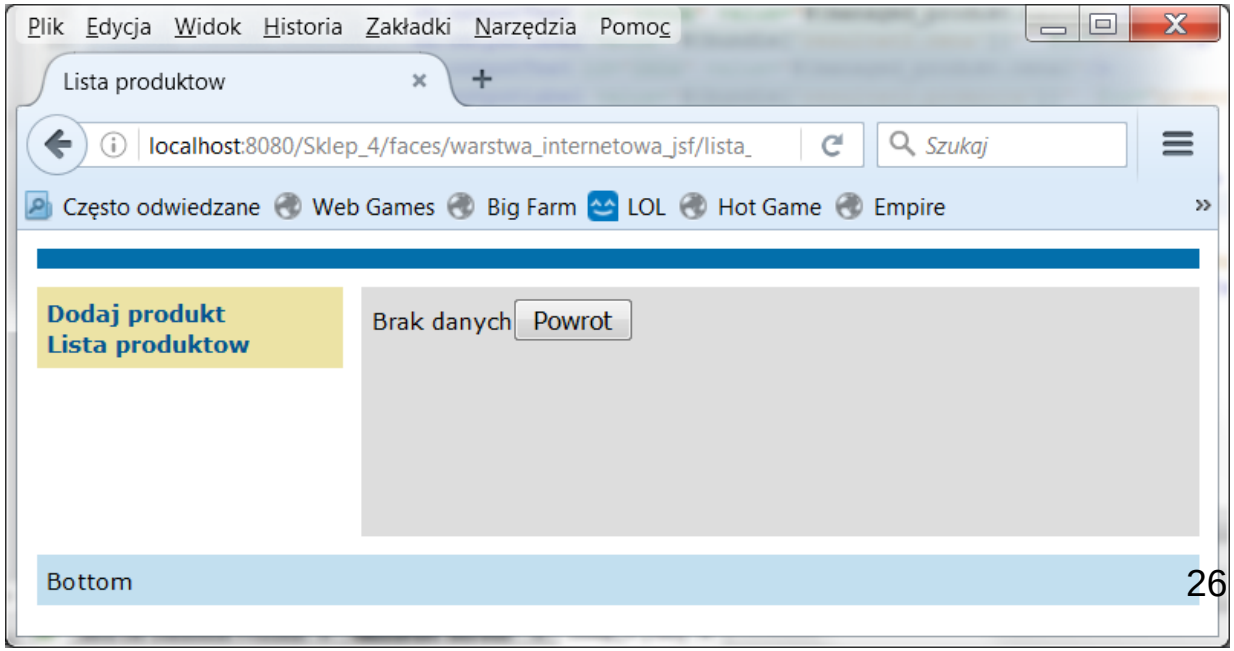
12 (cd). Modyfikacja pliku **lista_produktow.xhtml** – komunikaty z pliku Bundle.properties, sposób korzystania z modelu **items** komponentu **dataTable** oraz wyświetlenie daty produkcji

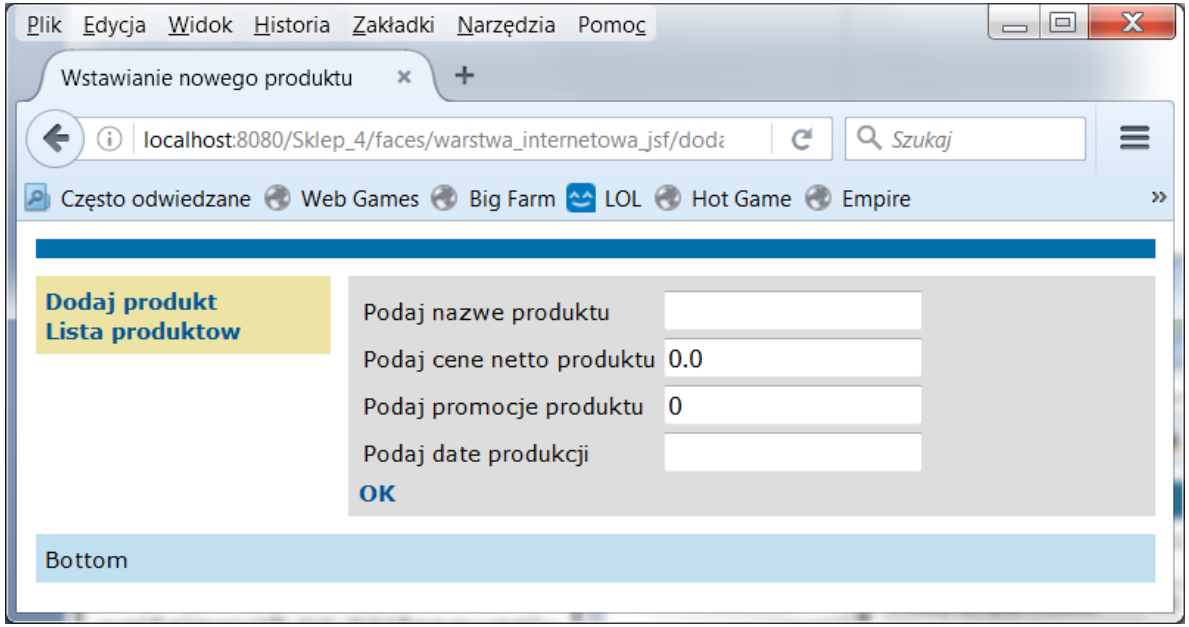
```
                <h:column>
                    <f:facet name="header">
                       <h:outputText value="#{bundle['lista_produktow.data']}"/>
                    </f:facet>
                    <h:outputText value="#{item.get(4)}"/>
                </h:column>
                <h:column>
                    <f:facet name="header">
                       <h:outputText value="#{bundle['lista_produktow.cena_brutto']}"/>
                    </f:facet>
                    <h:outputText value="#{item.get(5)}"/>
                </h:column>

            </h:dataTable>
        </h:panelGroup>

        <h:commandButton id="powrot" value="#{bundle['lista_produktow.powrot']}"
                                 action="/faces/index1"/>

    </h:form>

  </ui:define>
</ui:composition>

</body>
</html>
```

13. Uruchomienie aplikacji Sklep_4



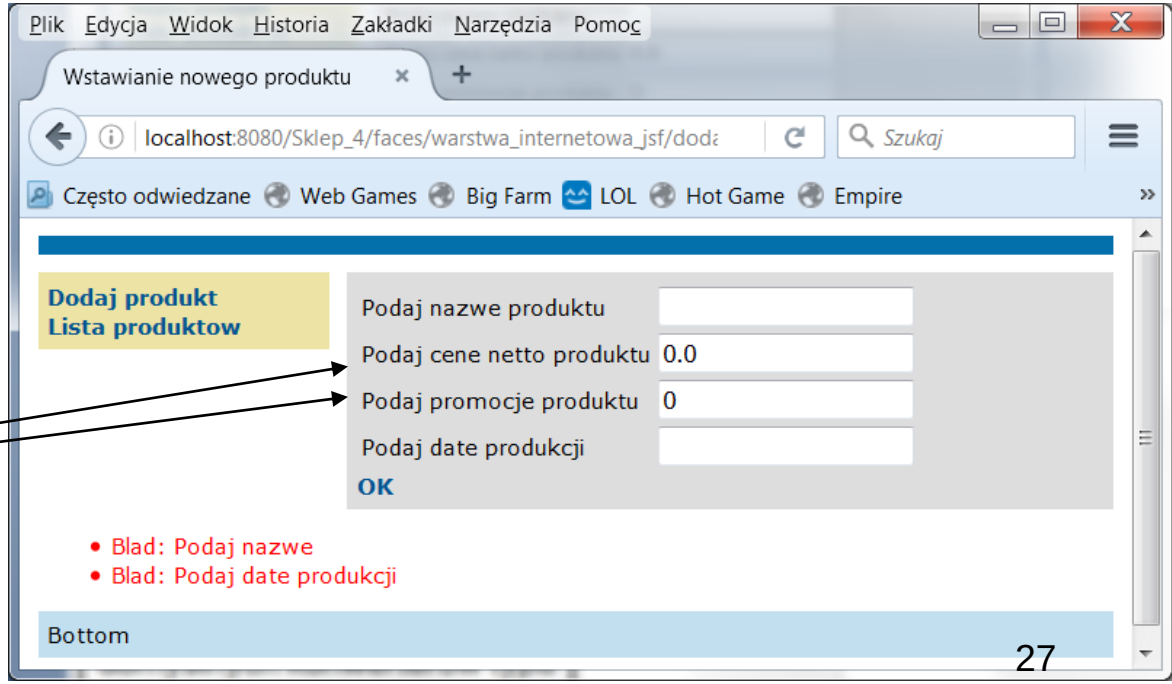
Efekt po kliknięciu na link
Lista produktow



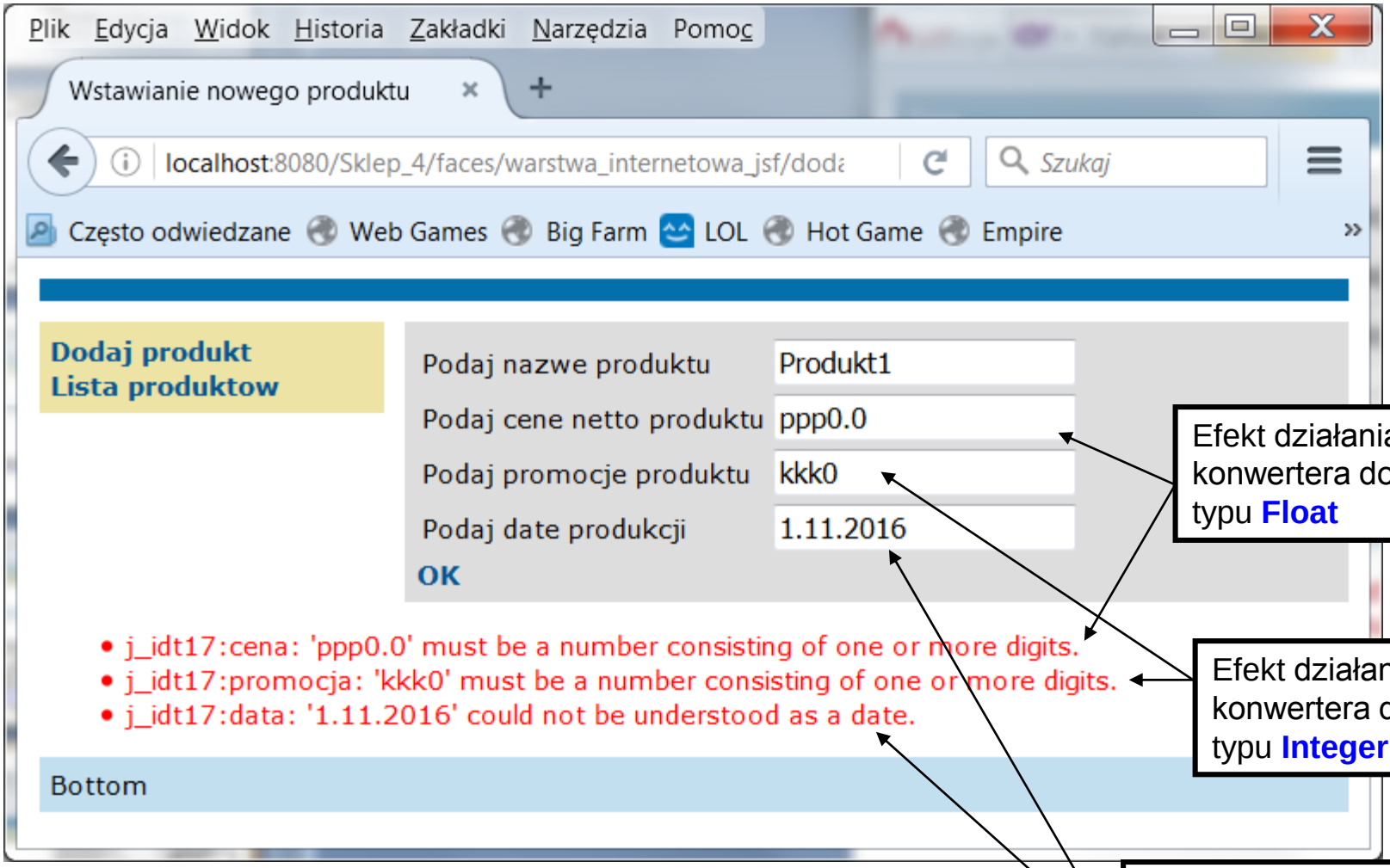


13(cd). Widok strony po kliknięciu na link **Dodaj produkt**

Widok strony po kliknięciu na przycisk **OK.**
Efekt uzyskany dzięki domyślnym wartościom pól wejściowych po zastosowaniu domyślnych konwerterów typu **Float i Integer** oraz polom **required i requiredMessage** w pozostałych polach wejściowych



13(cd). Działanie **konwerterów domyślnych** oraz typu **convertDateTime**

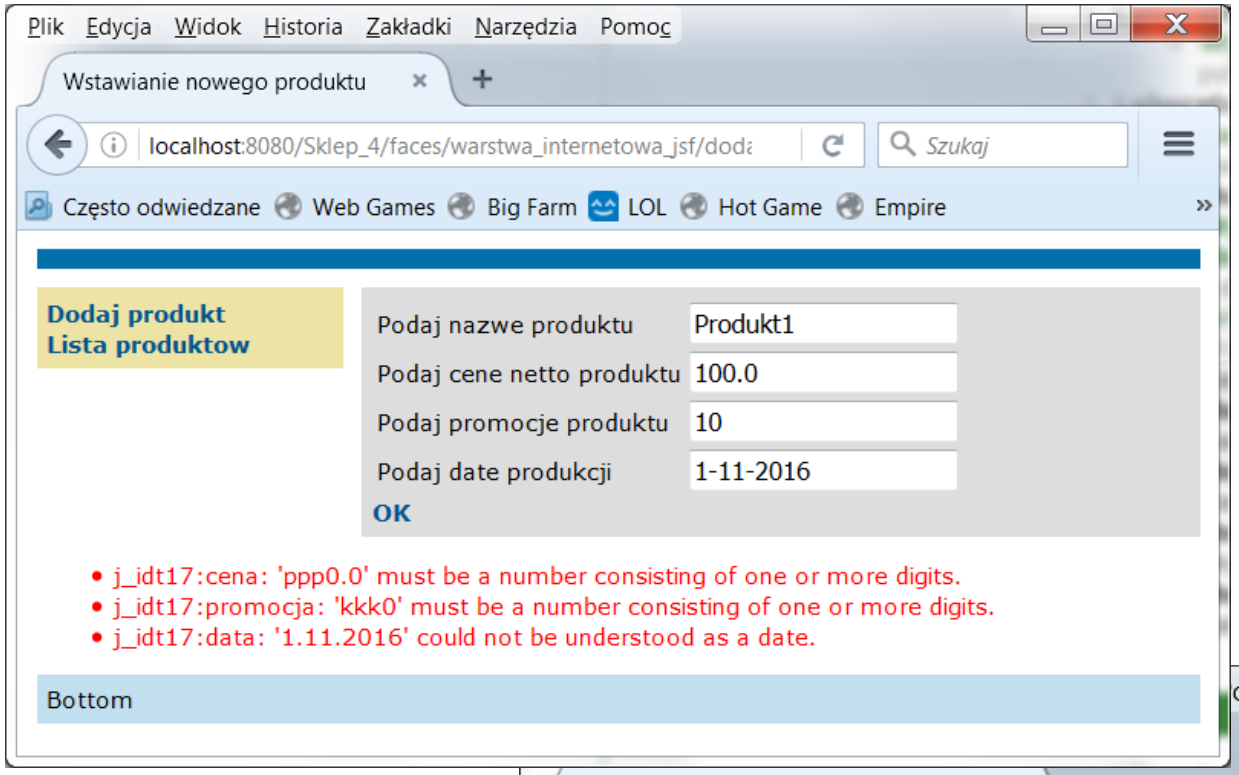


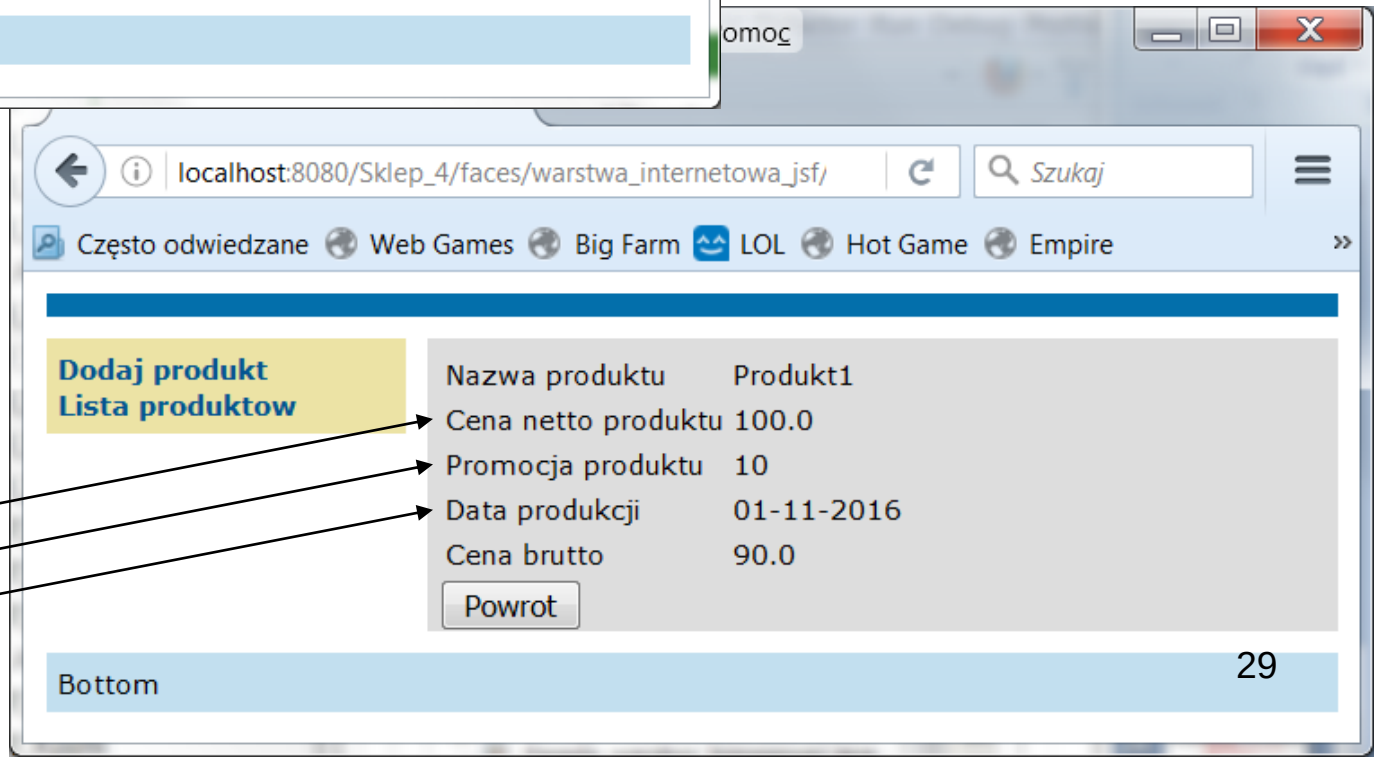
Efekt działania konwertera domyślnego typu **Float**

Efekt działania konwertera domyślnego typu **Integer**

Efekt działania konwertera typu **convertDateTime**

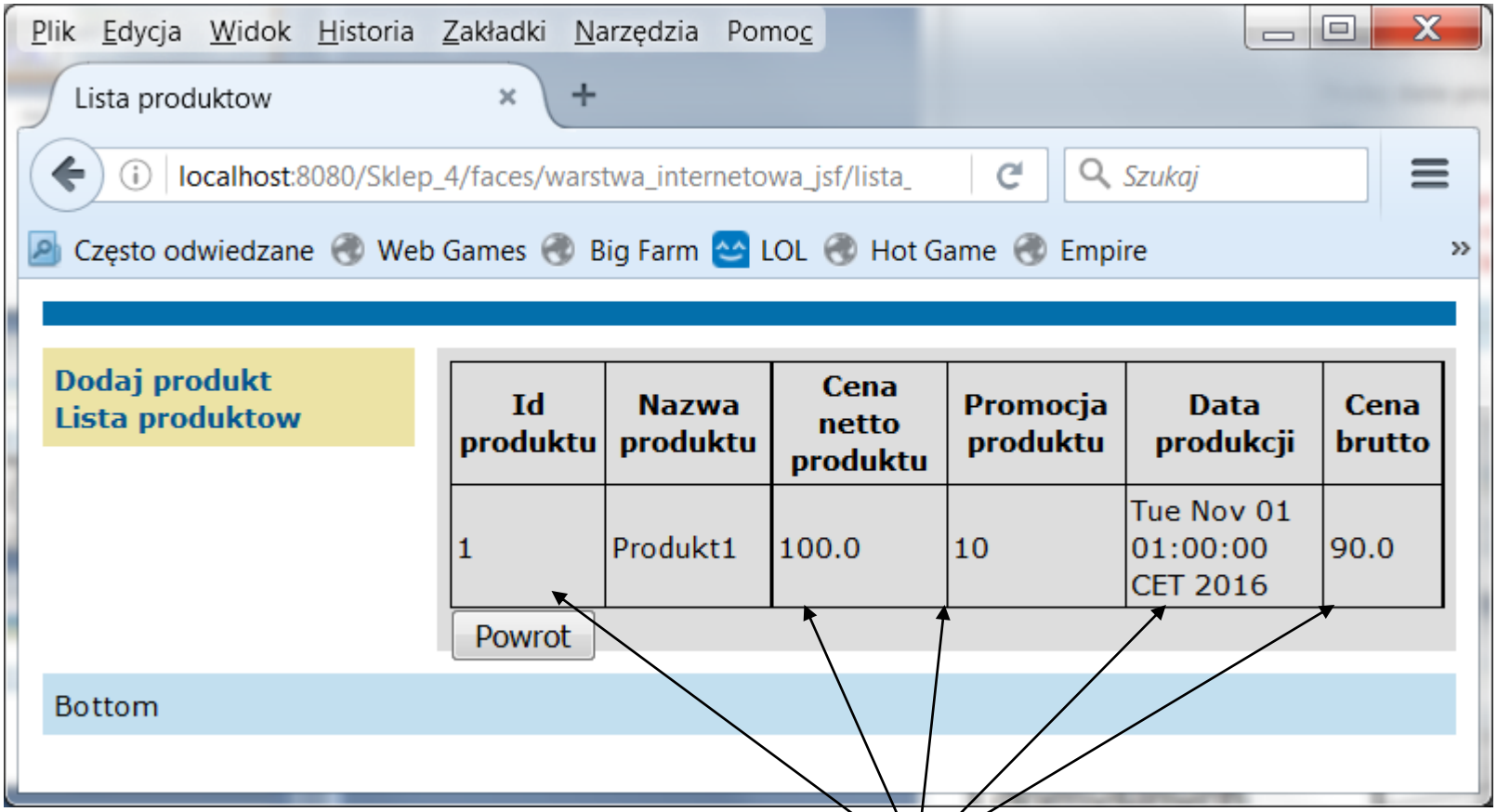
13(cd). Podanie poprawnych danych w formularzu





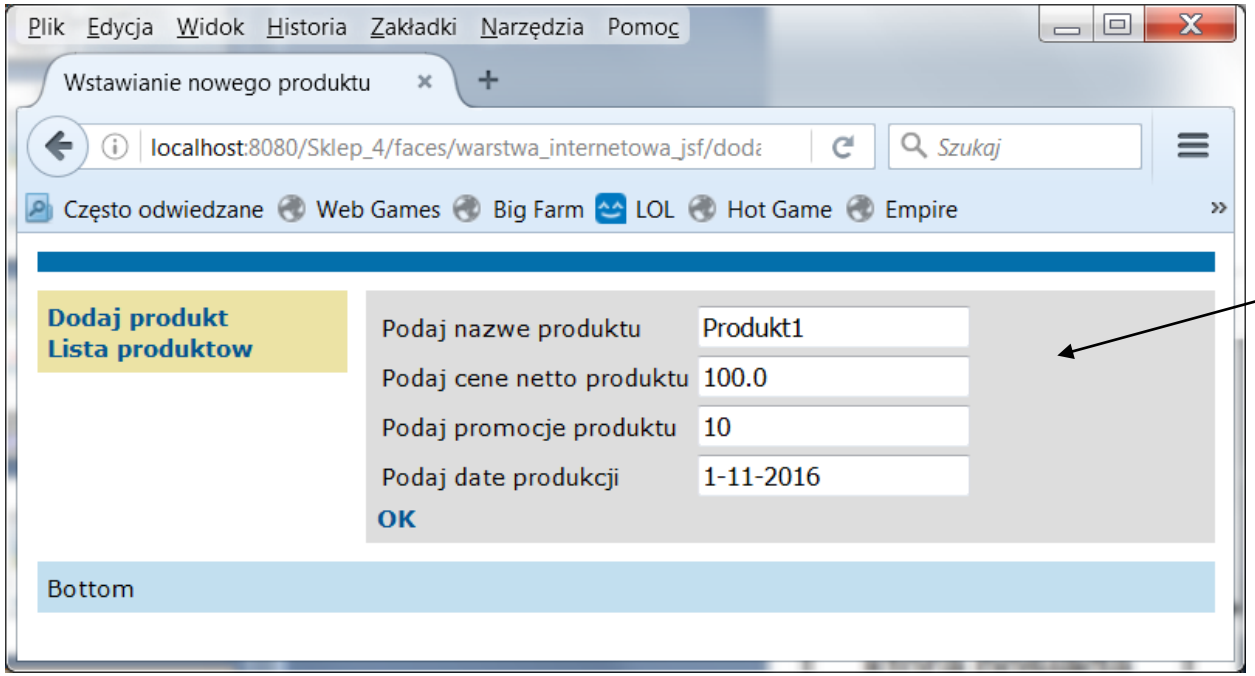
Wynik działania domyślnych konwerterów: **Float, Integer** oraz typu **convertDateTime**

13(cd). Dane w komponencie typu **dataTable**



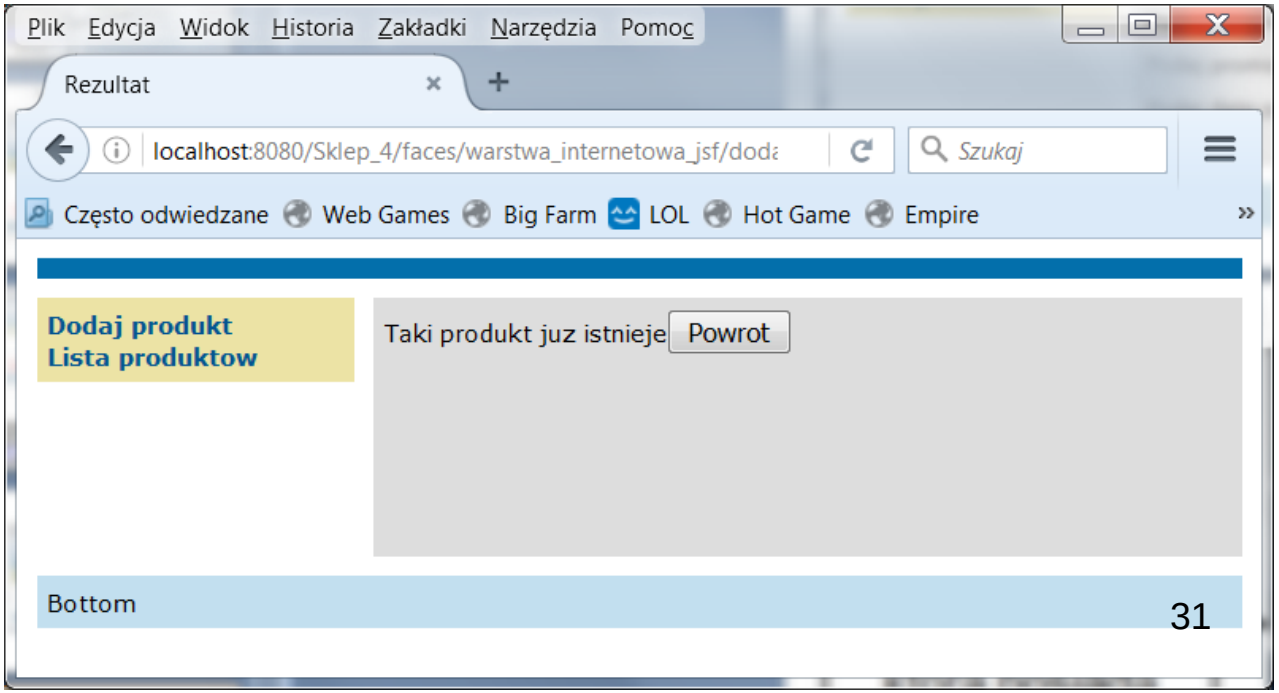

```
public DataModel utworz_DataModel() {
    return new ListDataModel(fasada.items());
}

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

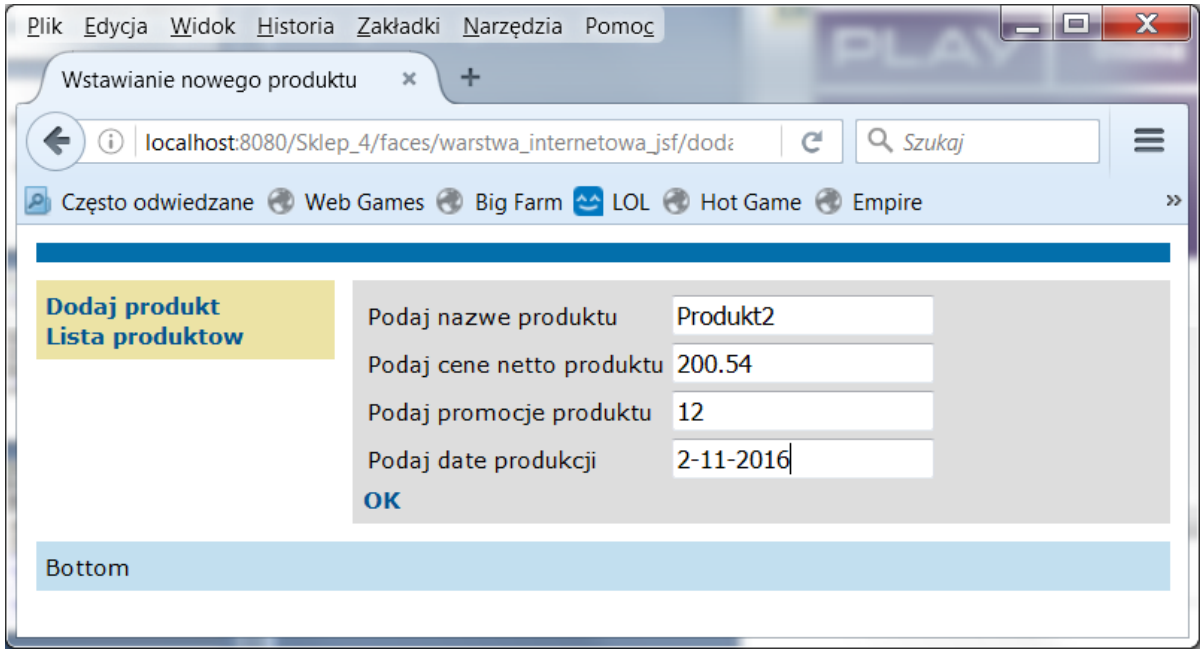
Wynik działania konwersji danych w metodzie **items()** klasy **Fasada_warstwy_biznesowej,** zastosowanej do budowy modelu typu **ListDataModel** zastosowanego w komponencie **<h:dataTable>** za pomocą metody **getItems()** klasy **Managed_produkt**

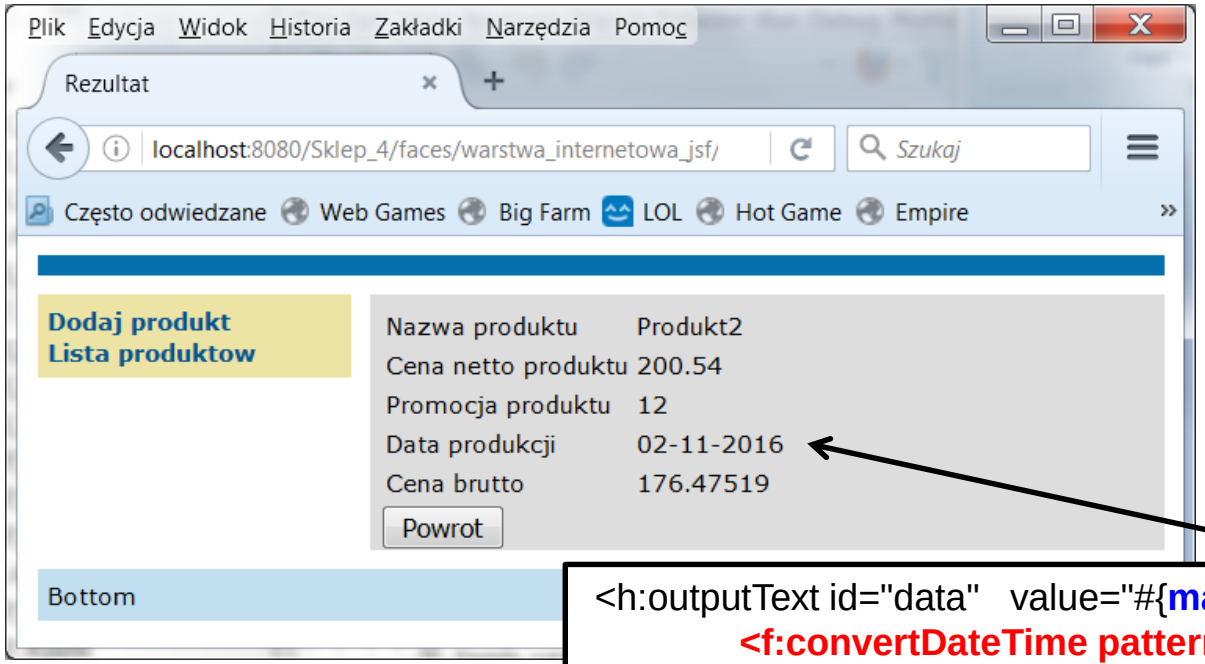


13(cd). Dodanie produktu o nazwie, cenie netto i promocji, którą posiada produkt wstawiony do aplikacji

Efekt po próbie dodania produktu o nazwie, cenie netto i promocji, którą posiada produkt wstawiony do aplikacji



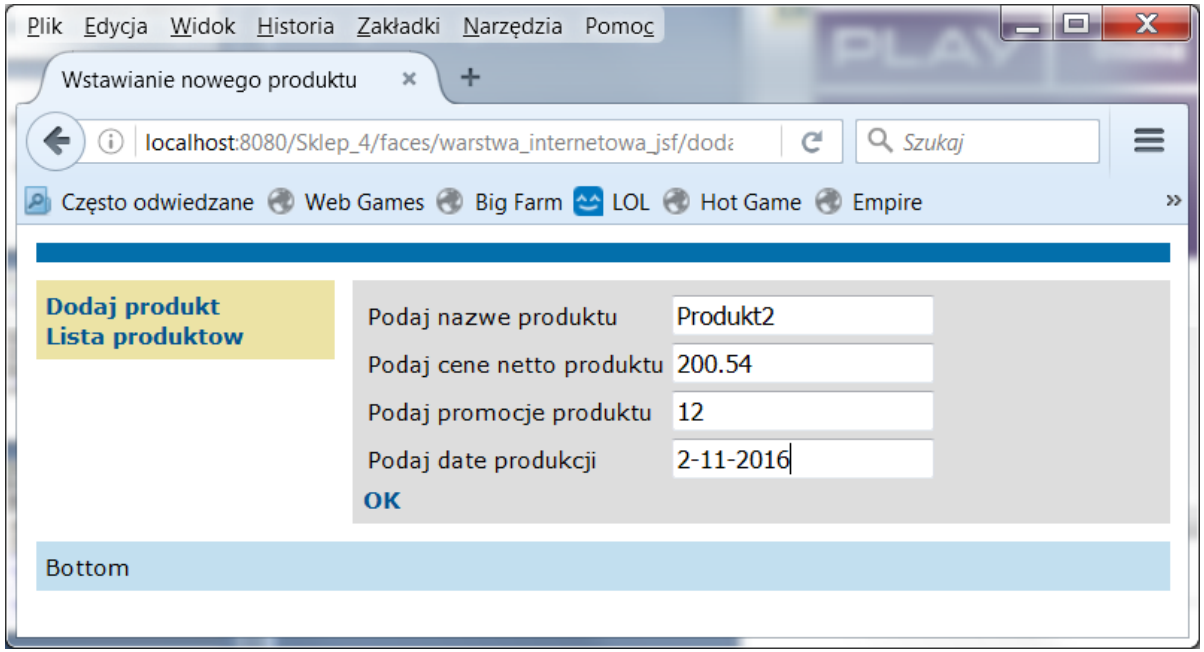


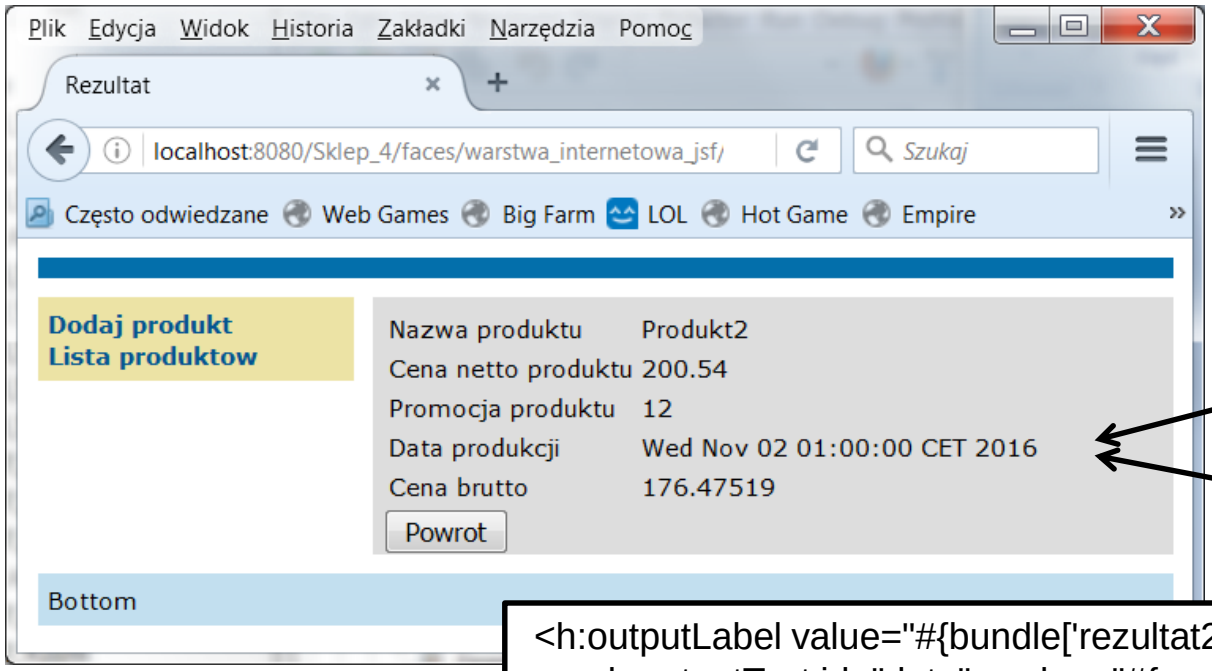


13(cd). Dodanie nowych danych produktu z zastosowaniem konwertera daty typu **convertDateTime** w pliku **rezultat2.xhtml**

```
<h:outputText id="data"   value="#{managed_produkt.data_produkcji}"/>
        <f:convertDateTime pattern="dd-MM-yyyy" />
</h:outputText>
```

13(cd). Dodanie nowych danych bez konwertera daty typu **convertDateTime** w pliku **rezultat2.xhtml.**

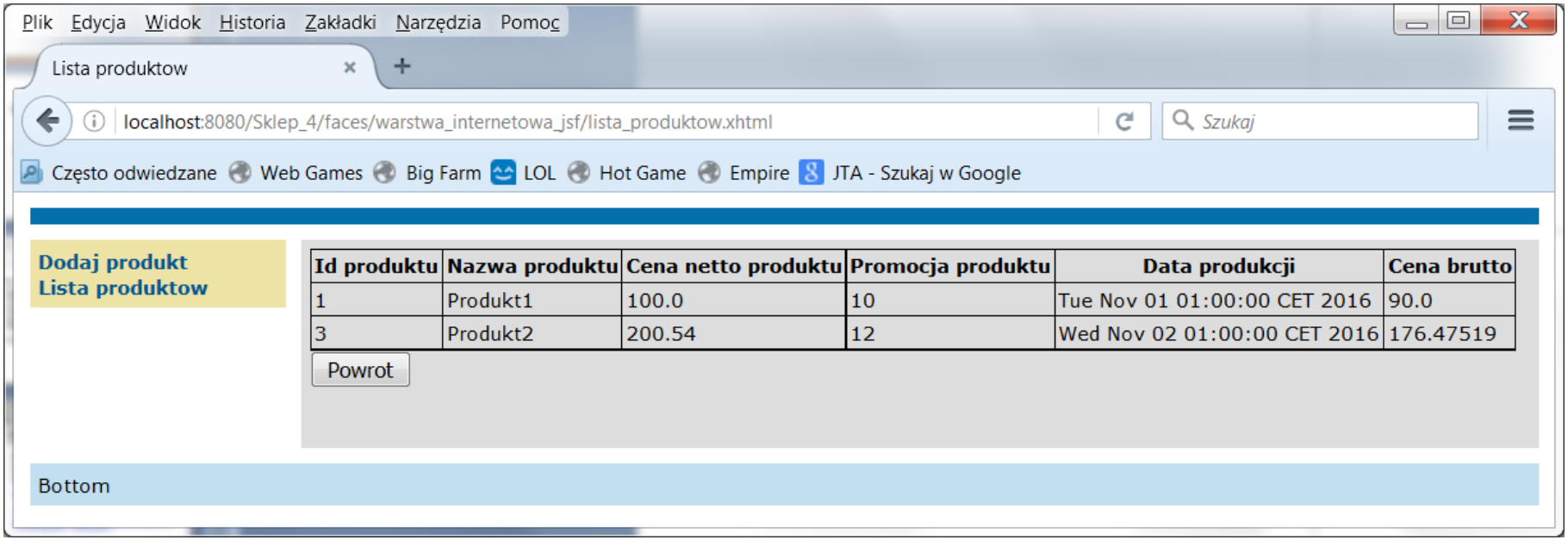




Domyślna konwersja daty typu **Date** na następującą postać znakową:
dzień_tygodnia miesiąc numer_dnia_miesiąca gg:mm:ss strefa_czasowa rok np:
2-11-2016 na:
Wed Nov 02 01:00:00 CET 2016

```
<h:outputLabel value="#{bundle['rezultat2.data']}" for="data"/>
    <h:outputText id="data"   value="#{managed_produkt.data_produkcji}"/>
```

13(cd). Aplikacja po dodaniu nowych danych – widok strony **lista_produktow.xhtml**



Przykład 5
Zastosowanie konwerterów liczbowych typu convertNumber i daty typu convertDateTime

Wykonanie zadania na kopii programu z przykładu 4 (lab4), o nazwie Sklep_5

1. Należy wykonać prace prezentowane na poszczególnych slajdach w aplikacji Sklep_5, wykonanej jako kopia programu Sklep_4

(1) W pliku strony **dodaj_produkt2.xhtml** należy dodać konwerter typu **convertNumber** z atrybutem **binding,** odwołującym się do konwertera zdefiniowanego w klasie obiektu typu **Managed_produkt,** gdzie nadano wartość atrybutu **pattern** metodą **setPattern** - następny slajd. W zależności od domyślnej wartości atrybutu l**ocale** wartości rzeczywiste mają część ułamkową oddzielaną od części całkowitej kropką lub przecinkiem.

```
<h:inputText
   id="cena"
   title="#{bundle['dodaj_produkt2.cena1']}"
   value="#{managed_produkt.cena}"
   required="true"
   requiredMessage="#{bundle['dodaj_produkt2.blad_cena']}" >
      <f:convertNumber binding="#{managed_produkt.number_convert}"  locale="en_US"  />
</h:inputText>
```

Pattern	**Wyjściowy łańcuch dla wartości:** 111111.111	**locale**
######.###	111111,111	**pl**
######.###	111111,111	**pl_PL**
######.###	111111.111	**en**
######.###	111111.111	**en_US**

(2) Należy dodać konwerter typu **NumberConverter**, który pozwala na wprowadzanie danych typu liczba rzeczywista, gdzie część ułamkowa jest oddzielona **przecinkiem** od części całkowitej liczby. Podczas wprowadzania dodawany jest symbol wartości **zł.**

```
@ManagedBean
@RequestScoped
public class Managed_produkt {

    @EJB
    private Fasada_warstwy_biznesowej fasada;
    private String nazwa;
    private float cena;
    private int promocja;
    private String cena_brutto;
    private DataModel items;
    private int stan = 1;
    private Date data_produkcji;

    private NumberConverter number_convert=new NumberConverter();

    public NumberConverter getNumber_convert() {
        this.number_convert.setPattern("######.## zł");
        return number_convert;
    }

    public void setNumber_convert(NumberConverter Number_convert) {
         this.number_convert = Number_convert;
    }
```

(3) W pliku strony **dodaj_produkt2.xhtml** należy dodać konwerter typu **javax.faces.Integer** (znacznik **<f:converter**) za pomocą **atrybutu converterId="javax.faces.Integer"**

```
<h:inputText
    id="promocja"
    title="#{bundle['dodaj_produkt2.promocja1']}"
    value="#{managed_produkt.promocja}"
    required="true"
    requiredMessage="#{bundle['dodaj_produkt2.blad_promocja']}" >
        <f:converter converterId="javax.faces.Integer" />
 </h:inputText>
```

(4) W pliku strony **dodaj_produkt2.xhtml** należy dodać konwerter typu **convertDateTime** z atrybutem **pattern** określającym sposób wprowadzania daty: **numer_dnia-numer_miesiaca-rok**

```
<h:outputLabel value="#{bundle['dodaj_produkt2.data']}" for="data" />
 <h:inputText
        id="data"
        title="#{bundle['dodaj_produkt2.data1']}"
        value="#{managed_produkt.data_produkcji}"
        required="true"
        requiredMessage="#{bundle['dodaj_produkt2.blad_data']}" >
                <f:convertDateTime pattern="dd-MM-yyyy" />
 </h:inputText>
```

(5) W pliku strony **rezultat2.xhtml** należy dodać konwerter **convertNumber**, zdefiniowany w klasie typu **Managed_produkt** oraz należy dodać atrybut **pattern** (anulujący wartość tego atrybutu nadanego metodą **setPattern** w klasie **Managed_produkt**), pozwalający na wyświetlanie liczby rzeczywistej, gdzie część ułamkowa jest wyświetlana **po przecinku** i cała wartość oznaczona jest symbolem **zł -**

```
<h:outputLabel value="#{bundle['lista_produktow.cena']}"  for="cena" />
<h:outputText id="cena" value="#{managed_produkt.cena}">
  <f:convertNumber  binding="#{managed_produkt.number_convert}"
                    pattern="####.### z&#322; -"/>
 </h:outputText>
```

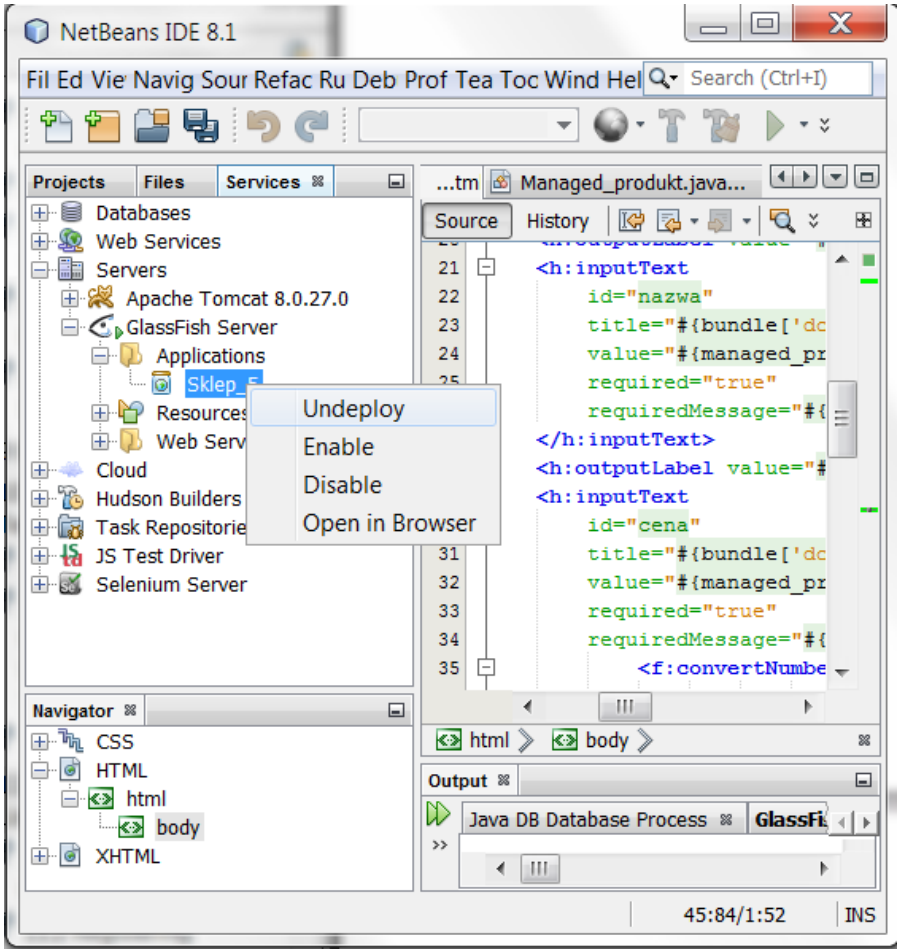
(6) W pliku strony **rezultat2.xhtml** należy dodać konwerter typu **convertDateTime** z atrybutami **dateStyle**, **locale**, **timeStyle** oraz **type**, gdzie dla wybranych wartości atrybutów uzyskano następującą postać wyświetlanej daty:
np. **Saturday, February 2, 2013 12:00:00 AM GMT**

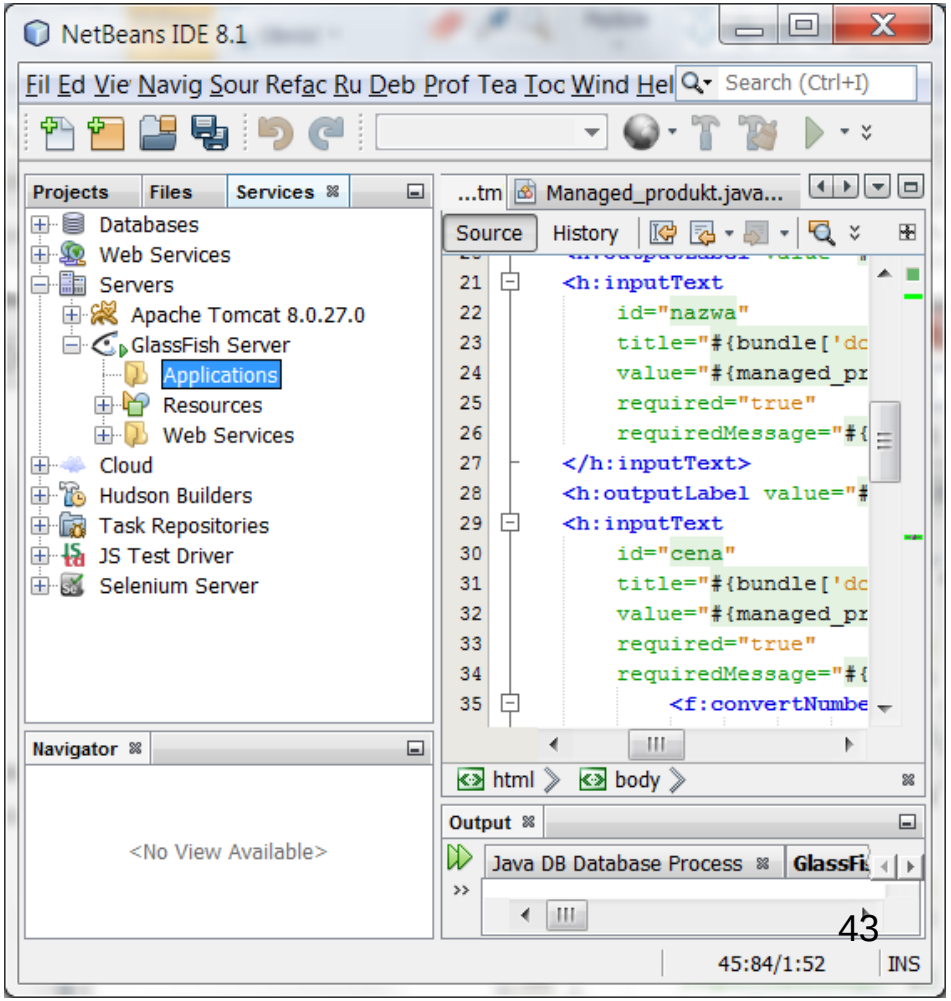
```
<h:outputLabel value="#{bundle['lista_produktow.data']}"   for="data"/>
<h:outputText id="data" value="#{managed_produkt.data_produkcji}">
        <f:convertDateTime dateStyle="full"
                           locale="en_US"
                           timeStyle="long" type="both"/>
</h:outputText>
```

(7) W pliku strony **rezultat2.xhtml** należy dodać konwerter typu **convertNumber** z atrybutami **currencySymbol** (%) i **type** do specyfikowania własnego typu (programisty) i oznaczeń wartości konwertowanej

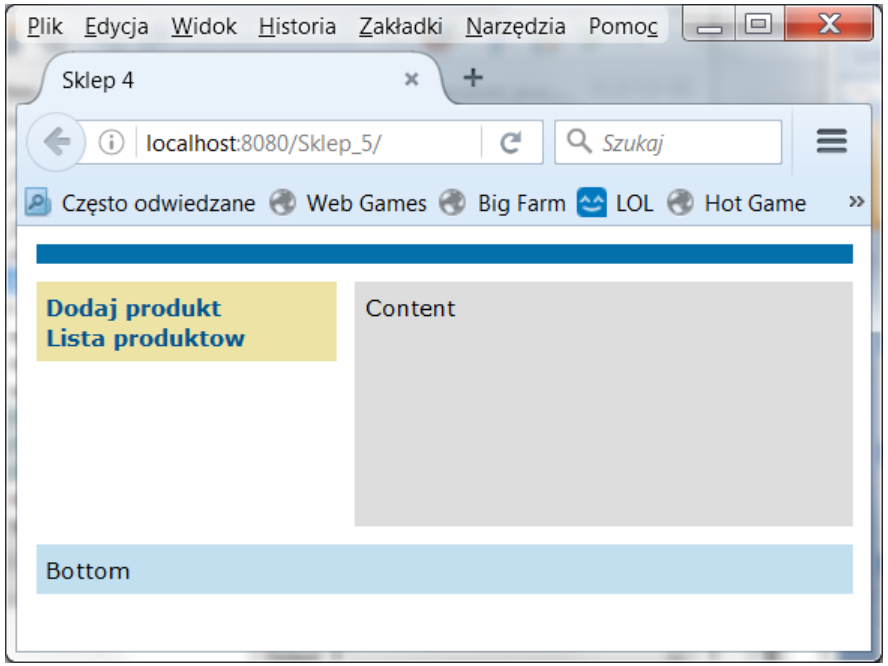
```
<h:outputLabel value="#{bundle['lista_produktow.promocja']}"
                    for="promocja" />
<h:outputText id="promocja" value="#{managed_produkt.promocja}">
      <f:convertNumber currencySymbol="%" type="currency"/>
 </h:outputText>
```

Usuwanie z kontenerów EE aplikacji typu EE

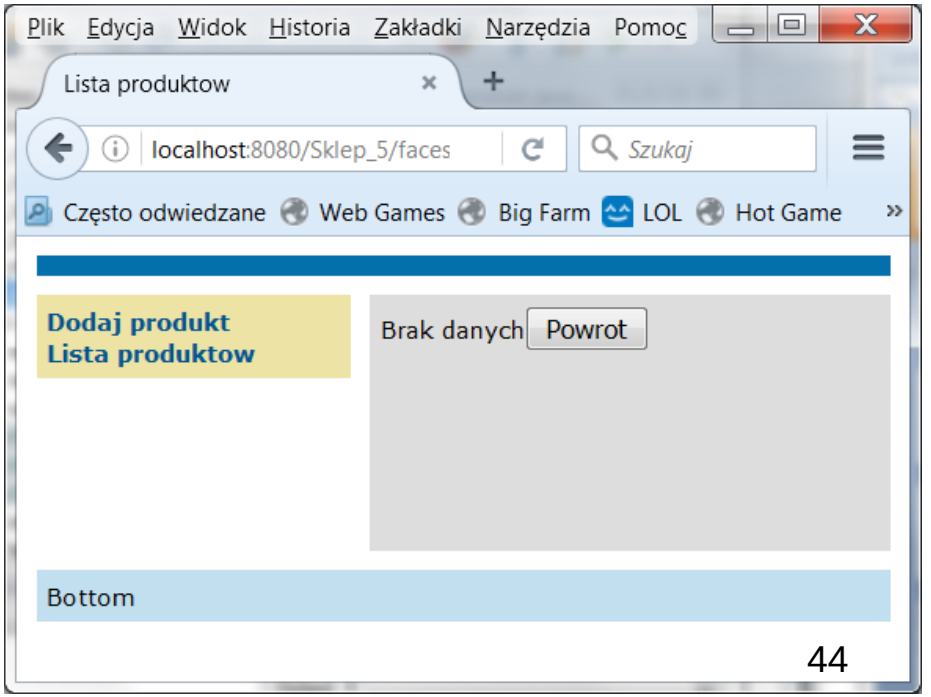


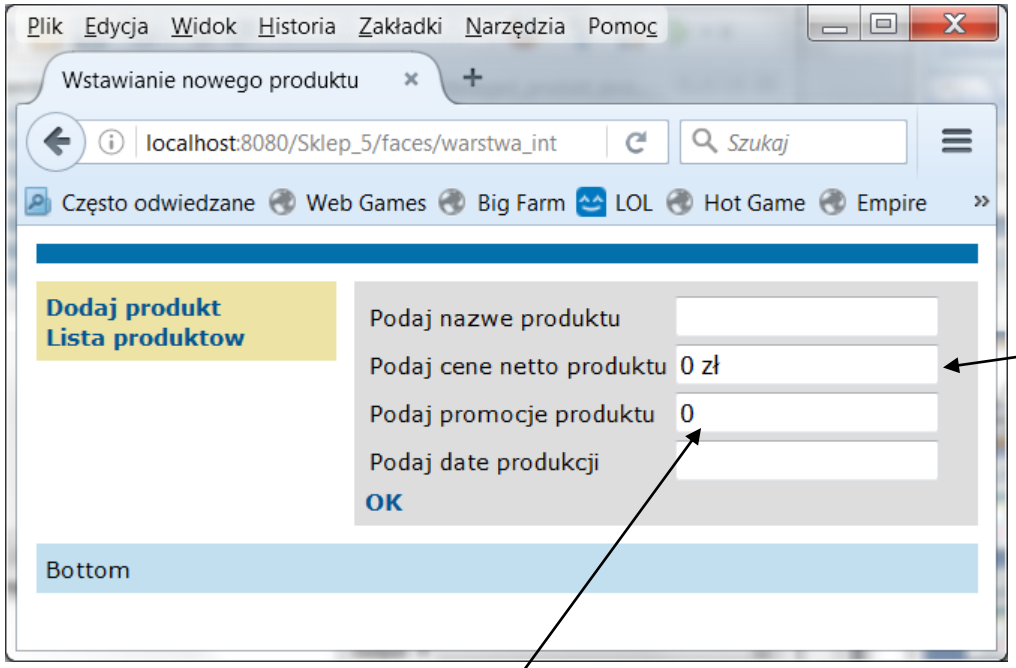


Uruchomienie aplikacji **Sklep_5**



Kliknięcie na link **Lista produktow**

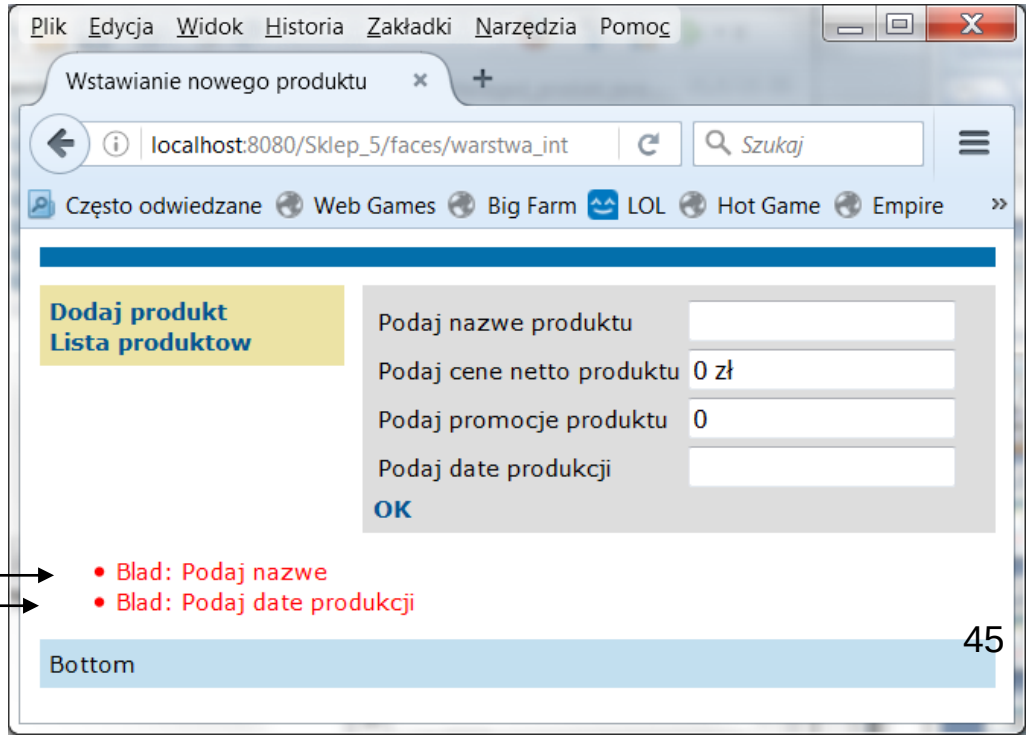




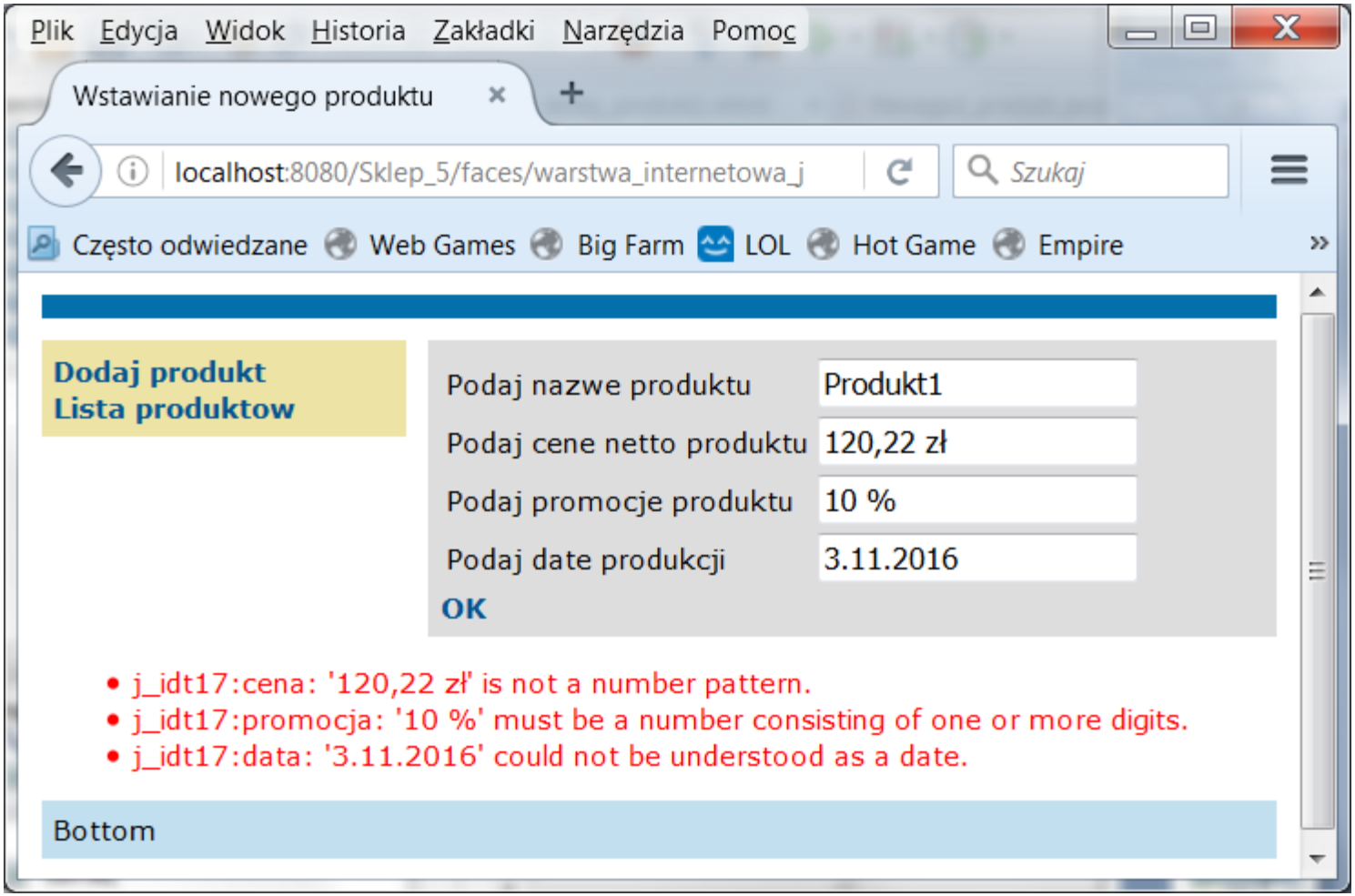
Działamie konwertera typu **convertNumber** zdefiniowanego w pliku Managed_produkt **(slajd(1), slajd (2))**

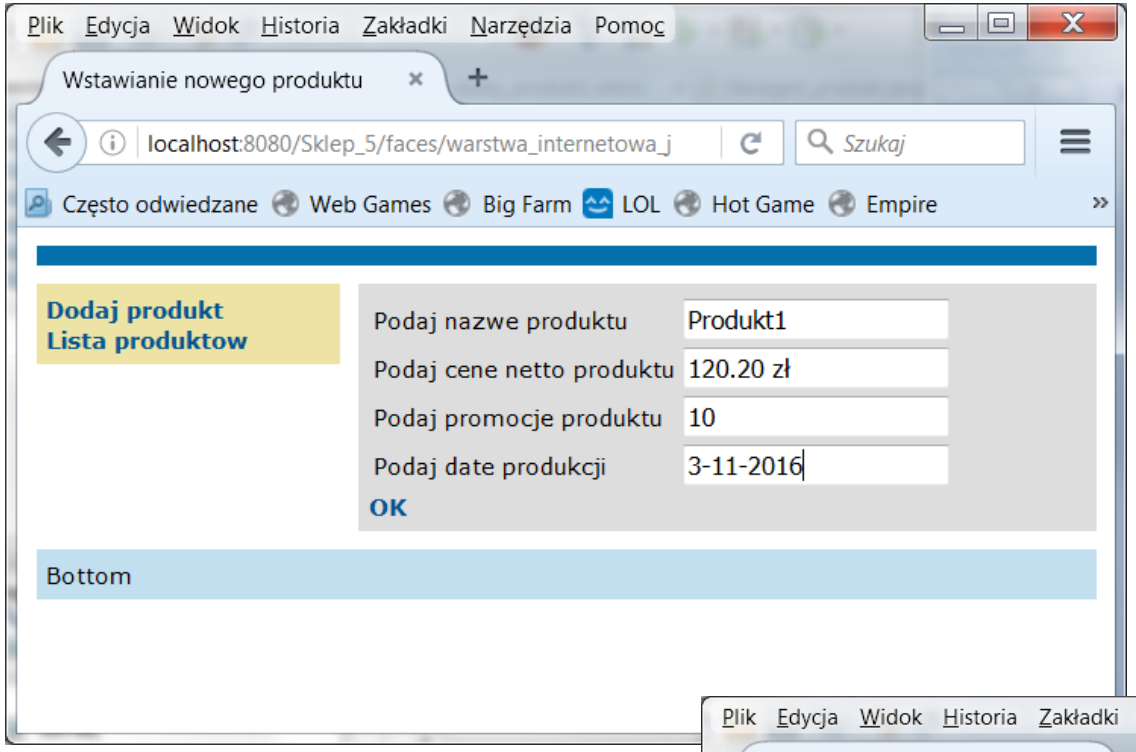
Działanie konwertera typu **Integer** wstawionego przez atrybut **converterId** - **slajd(3)**

Działanie atrybutu **required="true"** oraz **requiredMessage** dla pól bez wartości domyślnej

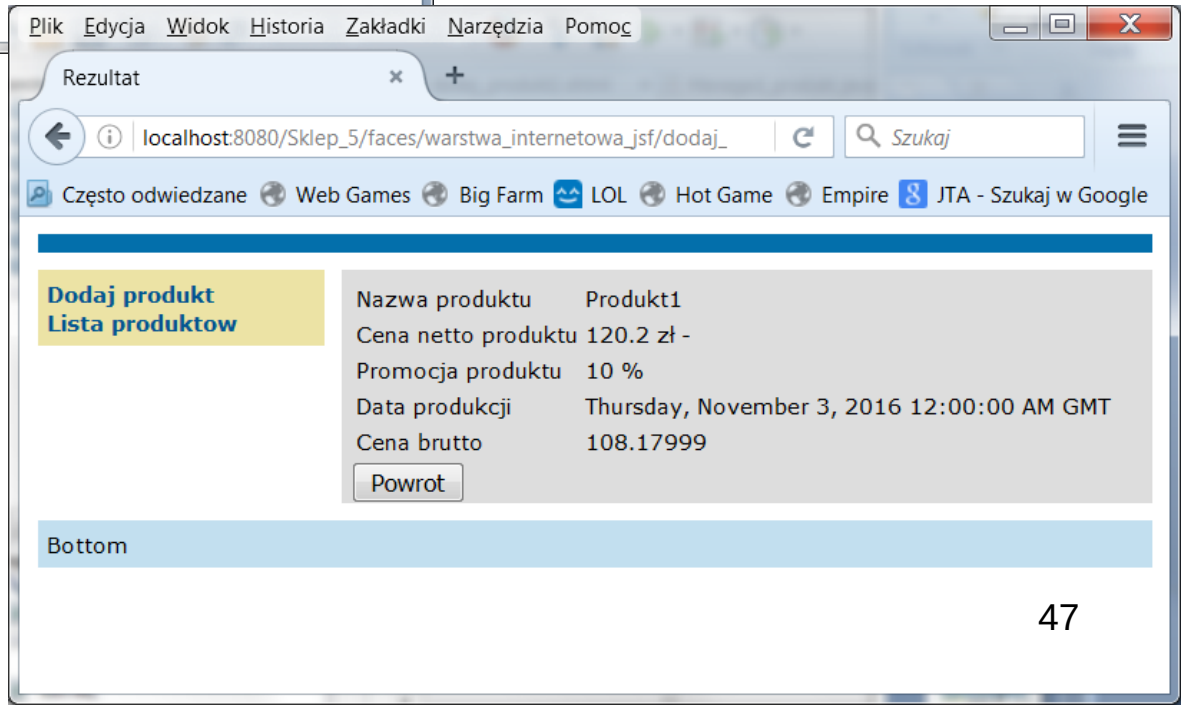


Efekt działania konwerterów domyślnych, typu **convertNumber (slajdy (1), (2), (3))** oraz **convertDateTime (slajd (4))**



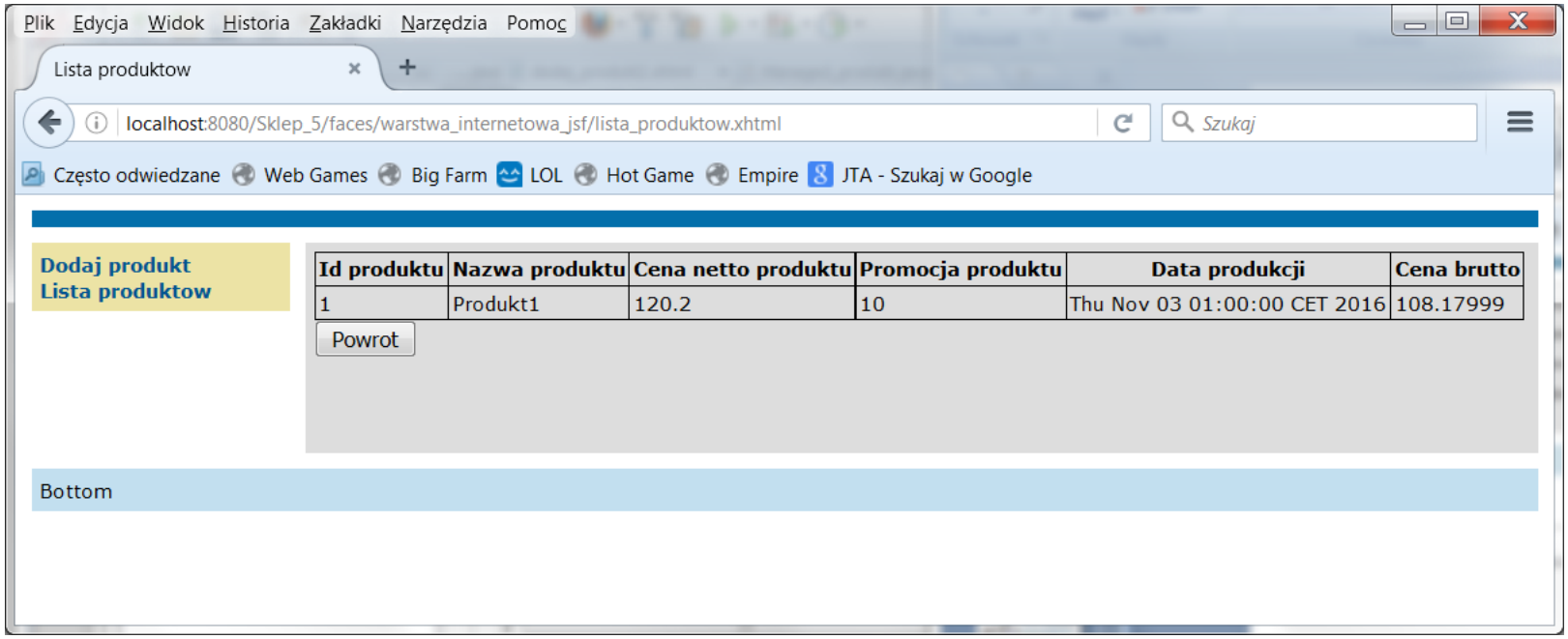


Efekt działania konwerterów typu **convertNumber (slajdy (1) , (2), (3))** oraz **convertDateTime (slajd (4))**

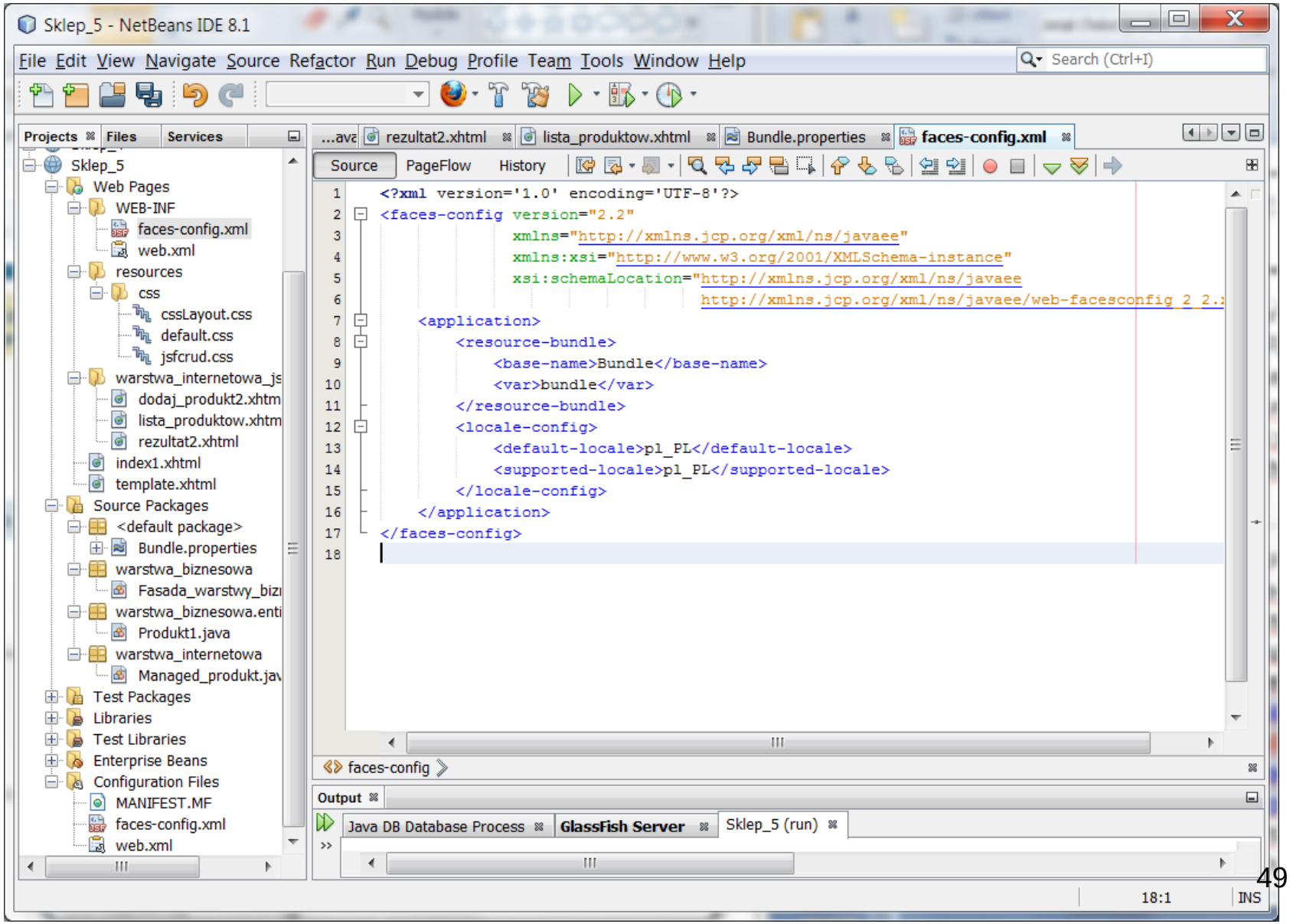


Działanie konwerterów typu **convertNumber (slajd (5), slajd(7))** **convertDateTime (slajd (6))**

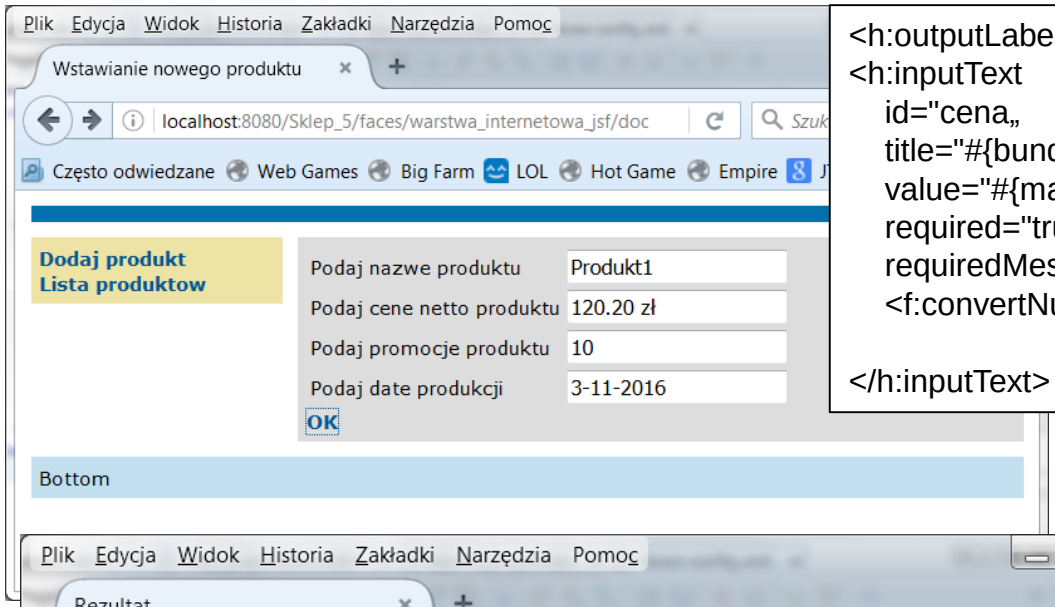
Wynik wprowadzenia nowego produktu prezentowany na stronie **lista_produktow.xhtml**



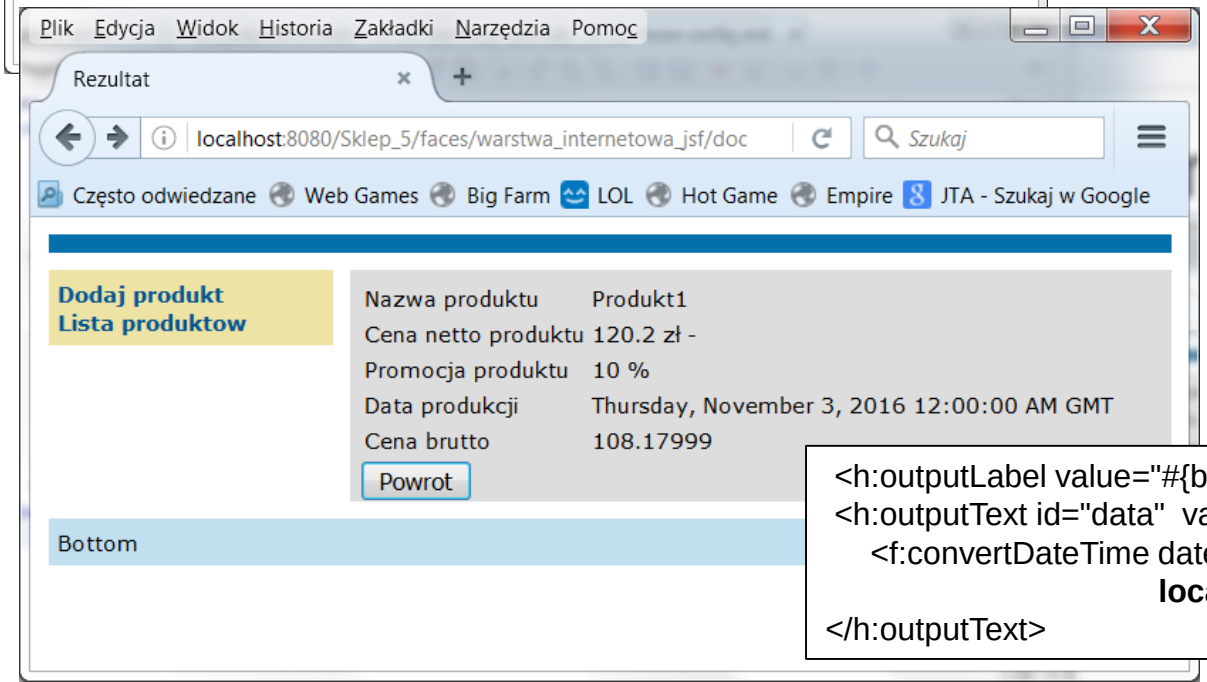
Należy zmienić styl prezentowania daty i liczb zmiennoprzecinkowych w pliku konfiguracyjnym **faces-config.xml,** jednak należy pozostawić atrybuty **locale** (**slajd (1), slajd(6)**)



Po wprowadzeniu danych, nadal obowiązuje styl **en_US** w prezentowaniu liczb i daty, mimo zmiany stylu prezentowania w pliku konfiguracyjnym, ponieważ atrybuty **locale** pozostawione w pliku **dodaj_produkt2.xhtml oraz rezultat2.xhtml** mają wyższy priorytet, niż ustawienia w pliku konfiguracyjnym

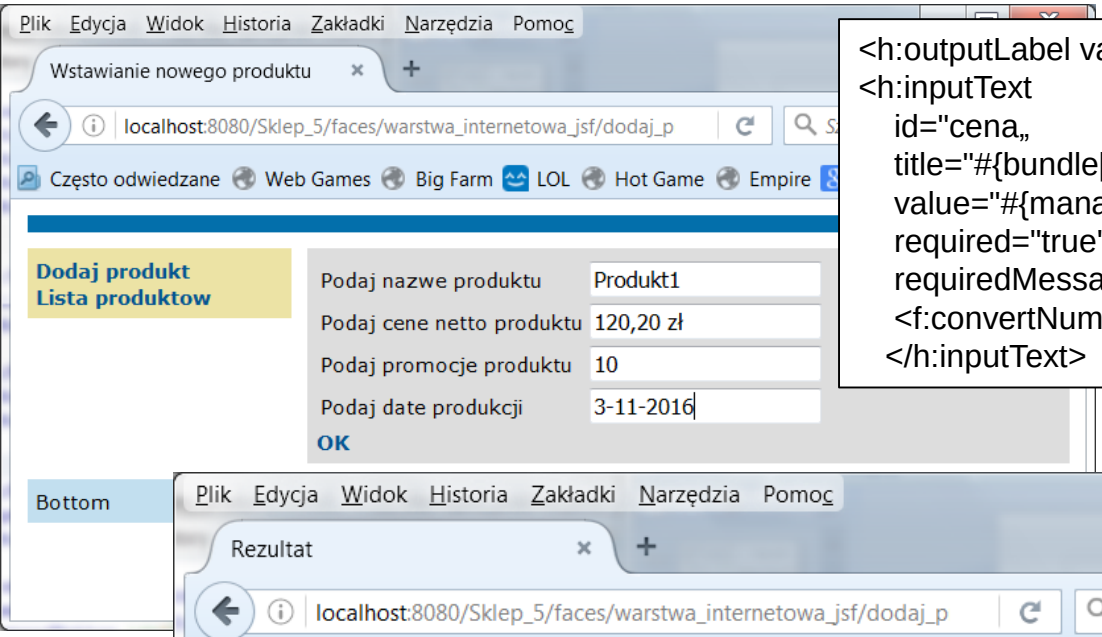



```
<h:outputLabel value="#{bundle['dodaj_produkt2.cena']}" for="cena" />
<h:inputText
    id="cena„
    title="#{bundle['dodaj_produkt2.cena1']}„
    value="#{managed_produkt.cena}"
    required="true"
    requiredMessage="#{bundle['dodaj_produkt2.blad_cena']}" >
    <f:convertNumber binding="#{managed_produkt.number_convert}"
                     locale="en_US" />
</h:inputText>
```

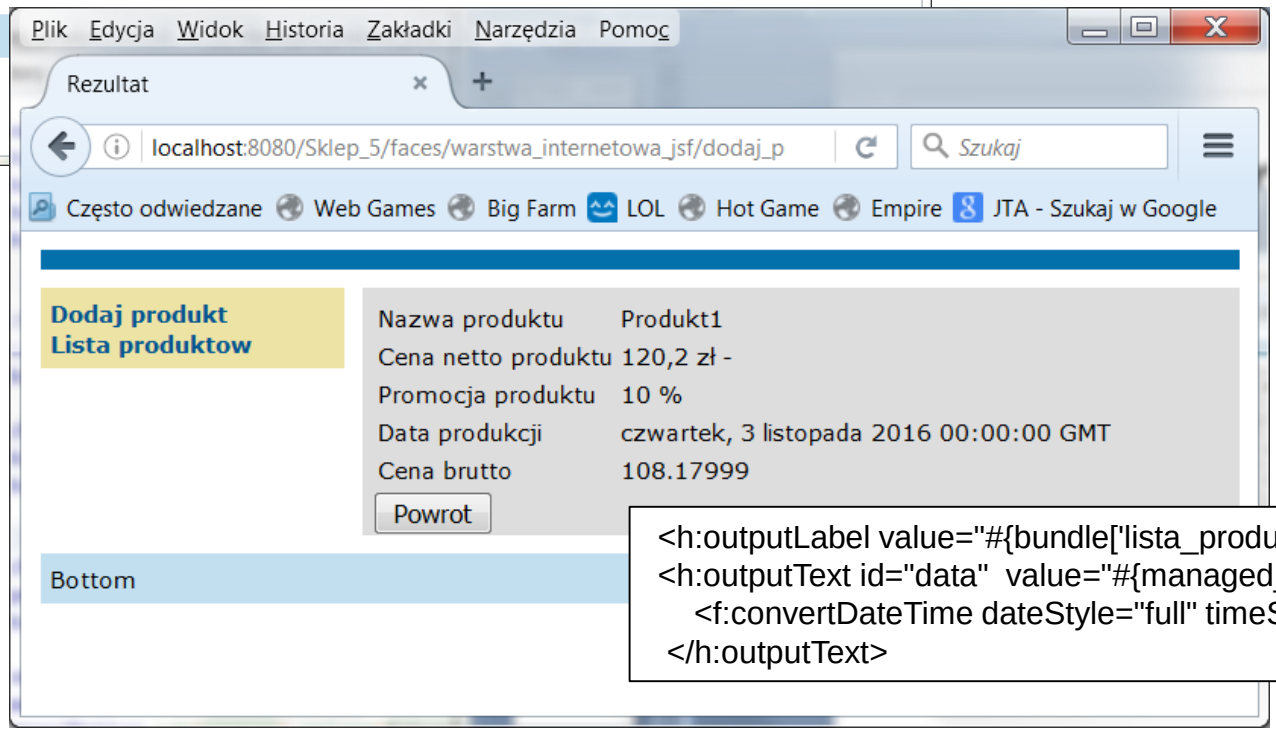



```
<h:outputLabel value="#{bundle['lista_produktow.data']}" for="data"/>
<h:outputText id="data"  value="#{managed_produkt.data_produkcji}">
    <f:convertDateTime dateStyle="full" timeStyle="long" type="both"
                       locale="en_US" />
</h:outputText>
```

Po wprowadzeniu danych, teraz obowiązuje styl pl_PL w prezentowaniu liczb i daty, podany w pliku konfiguracyjnym, ponieważ należało usunąć atrybuty locale = ”**en_US”**, nadające wyższy priorytet w prezentowaniu dat i liczb.

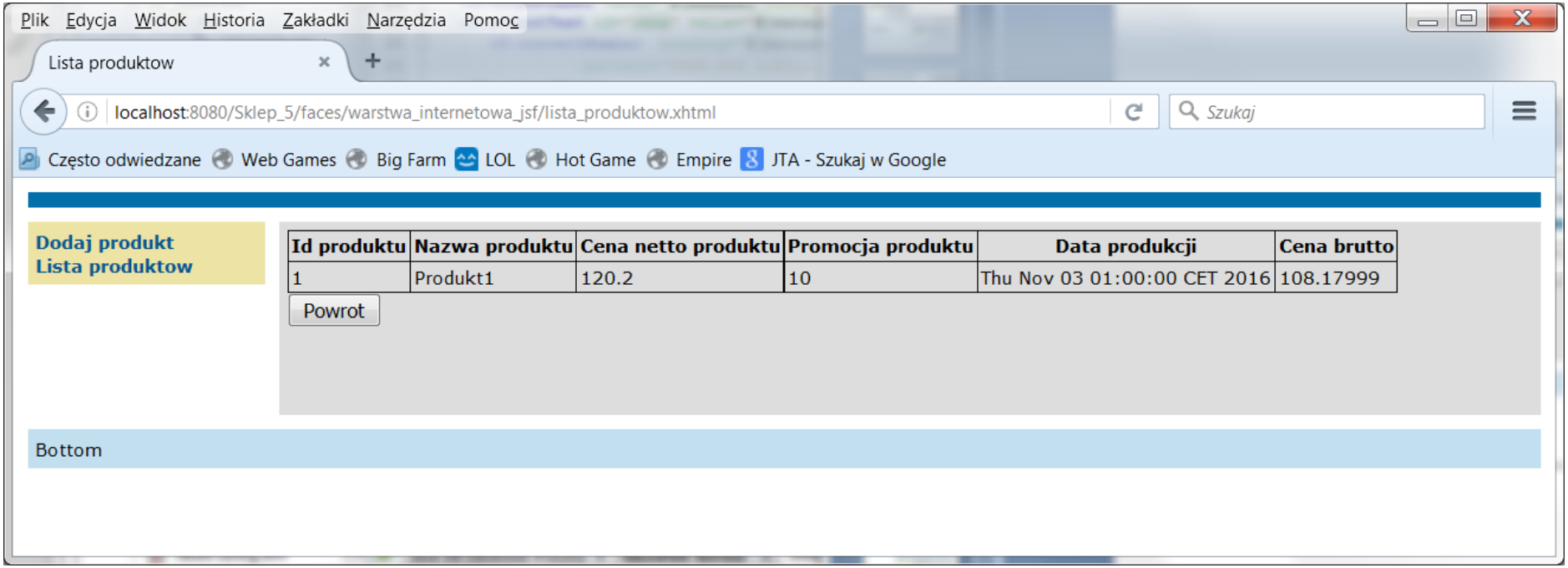



```
<h:outputLabel value="#{bundle['dodaj_produkt2.cena']}" for="cena" />
<h:inputText
   id="cena„
   title="#{bundle['dodaj_produkt2.cena1']}„
   value="#{managed_produkt.cena}"
   required="true"
   requiredMessage="#{bundle['dodaj_produkt2.blad_cena']}" >
   <f:convertNumber binding="#{managed_produkt.number_convert}"/>
  </h:inputText>
```




```
<h:outputLabel value="#{bundle['lista_produktow.data']}" for="data"/>
<h:outputText id="data"  value="#{managed_produkt.data_produkcji}">
   <f:convertDateTime dateStyle="full" timeStyle="long" type="both"/>
 </h:outputText>
```

Wynik wprowadzenia nowego produktu prezentowany na stronie lista_produktow.xhtml



2. Należy dodać zmodyfikować typ promocji na float i dodać konwerter podczas wstawiania na stronie dodaj_produkt2.xhtml i wyświetlania wartości promocji jako typu zmiennoprzecinkowego na stronie rezultat2.xhtml (wyświetlanie ##.## %)
lub
3. Należy dodać konwerter do wyświetlania ceny brutto na stronie rezultat2.xhtml (wyświetlanie w formacie ####.## zł)