Podstawy technologii JavaServer Faces

wg

https://docs.oracle.com/javaee/7/JEETT.pdf

Wykład 4

Technologie internetowe

(5) Dodawanie wybranych znaczników do strony
Kontynuacja zagadnień z wykładu 3

rozdziały 10 -11

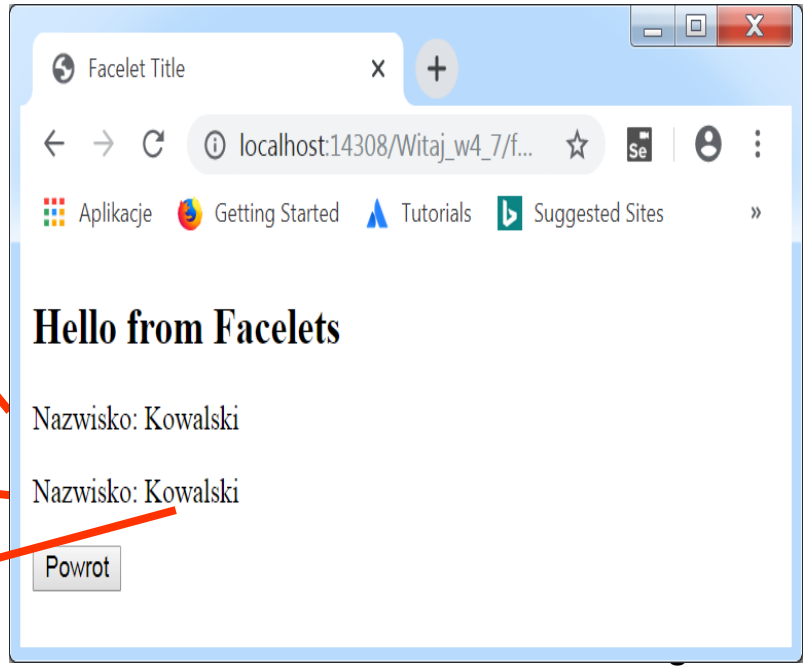
(5) Dodawanie wybranych znaczników do strony

Znaczniki pól wyjściowych

- **h:outputFormat**
- **h:outputLabel**
- **h:outputLink**
- **h:outputText**

Zastosowanie znacznika h:outputLabel do renderowania Label (atrybut value komponentu h:outputText reprezentuje tekst komponentu **h:outputLabel** – można zastąpić go atrybutem **value** tego komponentu)

```
<h:outputLabel id="l2">
  <h:outputText id="l3"
    value="Nazwisko: #{personalia.name}"
  />
</h:outputLabel> <p></p>

<h:outputLabel id="l1" for ="14"
          value="Nazwisko: " />
<h:outputText id="l4"
     value="#{personalia.name}" />
<p></p>
```



(6) Dodawanie wybranych znaczników do strony

Zastosowanie znacznika h:outputLink do renderowanie Hyperlink

<h:outputLink value="javadocs"> **Documentation for this demo**
</h:outputLink>

Tekst zagnieżdżony wyświetla się jako tekst Hyperlinku na stronie.

Wyświetlanie wiadomości za pomocą znacznika h:outputFormat

Umożliwia wklejanie do komunikatu wartości atrybutów obiektów

<h:outputFormat
value="Hello, **{0}!** You are visitor number **{1}** to the page.">
<f:param value="#{hello.name}" />
<f:param value="#{bean.numVisitor}"/>
</h:outputFormat>

np

Hello, **Bill!** You are visitor number **4** to the page.

(7) Dodawanie wybranych znaczników do strony

Zastosowanie znaczników tworzących komponenty poleceń i nawigacji.

Znaczniki:

h:commandButton jest renderowany jako przycisk

h:commandLink jako hyperlink.

Znaczniki te używają atrybuty:

- **action** – łańcuch określający wywoływaną metodę od obiektu typu Managed Bean. Metoda zwraca łańcuch określający adres strony, która zostaje wywołana. Atrybut może zawierać bezpośrednio adres tej strony
- **actionListener** - wskazanie wywołanej metody o określonym nagłówku od obiektu typu ziarno, obsługującej zdarzenie

Przykład a (podobna obsługa atrybutu action w przykładach 2, 3, 4, 5, 6)

```
<h:commandButton id="powrot"
                 value="#{bundle.rezultat2.akcja}"
                 action="/faces/index2"/>
```

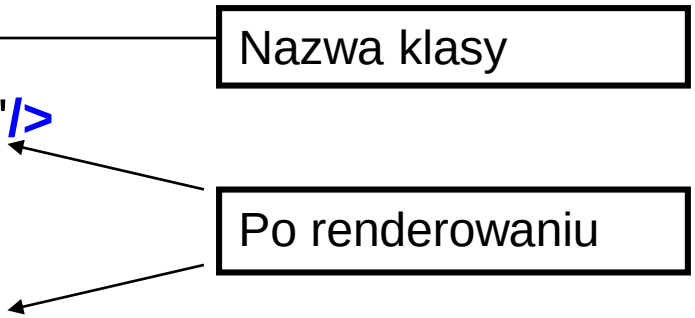
Przykład b

```
<h:commandLink
    id="Duke"
    action="bookstore"
    actionListener="#{actionBean.chooseBookFromLink}"
    value="#{bundle.dodaj_produkt2_akcja}" />
```

Atrybut **action** zawiera adres strony, która zostanie wywołana, a jednocześnie zostanie wykonana metoda **chooseBookFromLink** od obiektu typu **actionBean** (atrybut **actionListener**).

Przykład c (renderuje JavaScript!)

```
<h:commandLink id="Duke" action="bookstore">
   <f:actionListener
        type="LinkBookChangeListener" />
  <h:outputText value="#{bundle.Book201}"/>
</h:commandLink>
```



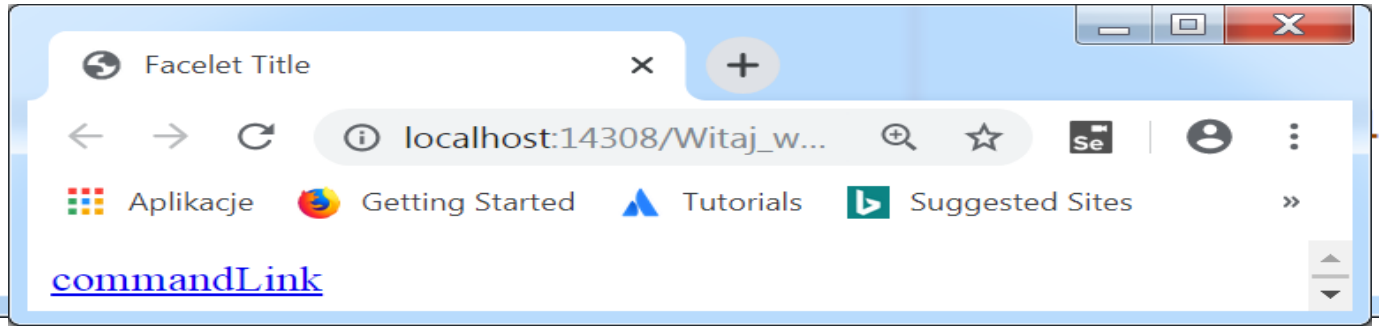

```
  <a id="_idt16:Duke" href="#"
        onclick="mojarra.jsfcljs(document.getElementById('j_idt16'),
                { 'j_idt16:Duke' : 'j_idt16:Duke'}, ' '); return false; ">
   My Early Years: Growing Up on Star7, by Duke</a>
```

Przykład c (cd)

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head>
  <title>Facelet Title</title>
</h:head>
<h:body>
  <h:form>
      <h:commandLink id="Duke" action="rezultat">
          <h:outputText value="commandLink"/>
      </h:commandLink>
   </h:form>
</h:body>
</html>
```

Po renderowaniu

```
<a id="j_idt5:Duke" href="#"
onclick="mojarra.jsfcljs(document.getElementById('j_idt5'),
{'j_idt5:Duke':'j_idt5:Duke'},'');return false">commandLink</a>
```



(8) Dodawanie wybranych znaczników do strony

Dodawanie grafiki za pomocą znacznika h:graphicImage

```
<h:graphicImage id="mapImage" url="/resources/obrazy/Desert.jpg"/>
```

lub

```
<h:graphicImage id="mapImage" name="Desert.jpg"
                    library= "obrazy" alt="#{bundle.ChooseBook}"  />
```

lub

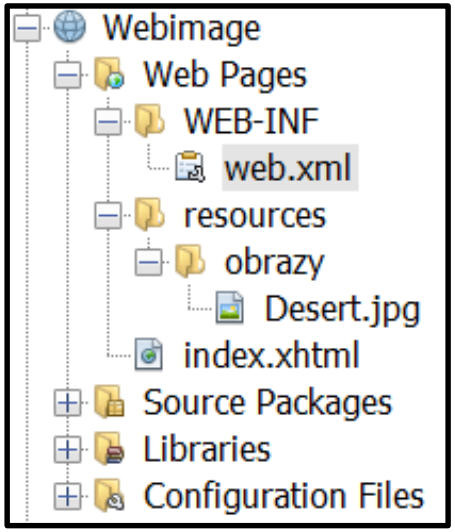
```
<h:graphicImage value="#{resource['obrazy:Desert.jpg']}"/>
```

Równoważna definicja za pomocą arkusza stylu css (np desert.css):

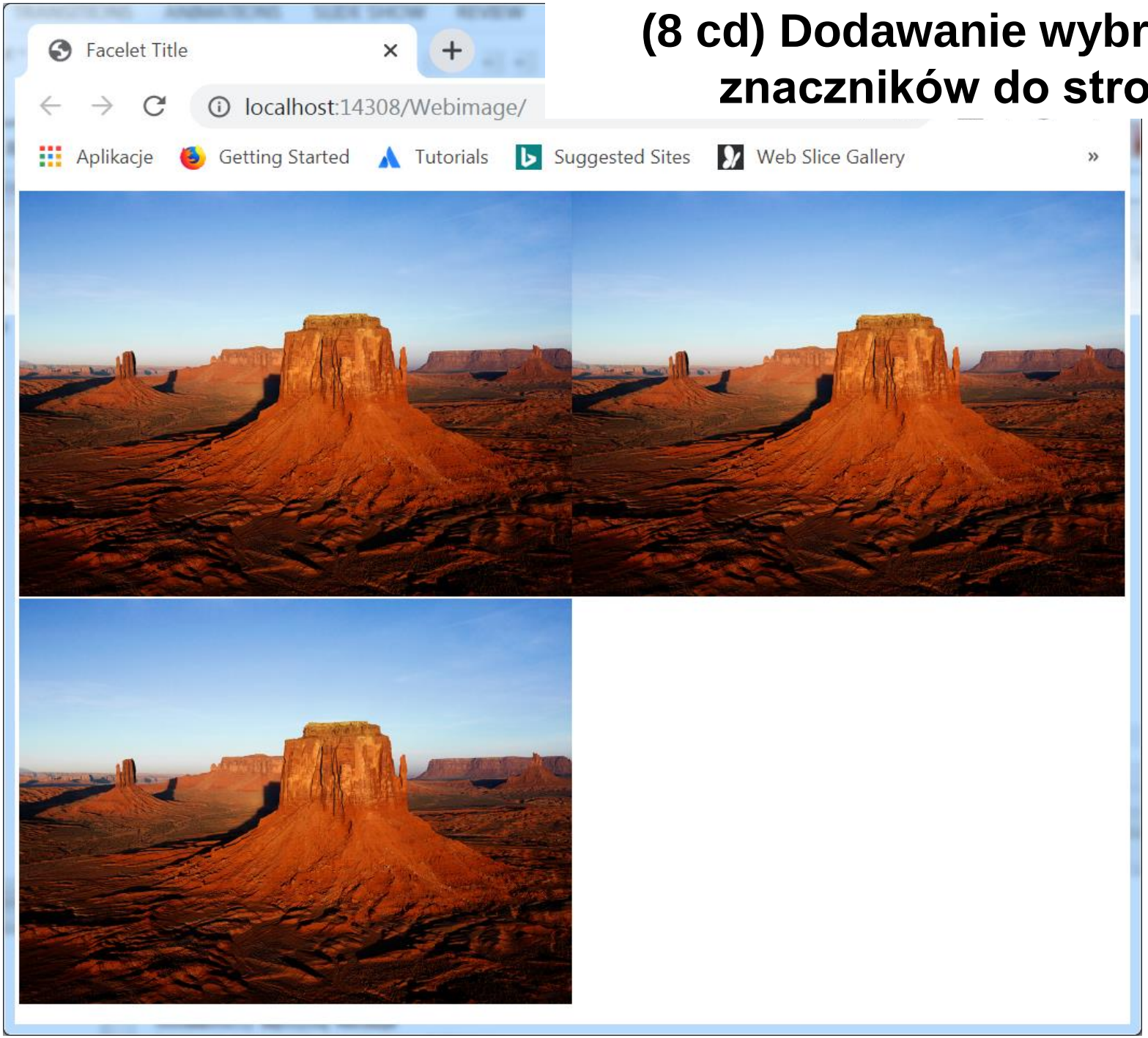
```
header{position: relative; height: 1500px; width:2000px;
  background: #fff url(#{resource['obrazy:Desert.jpg']});
}
```

(8 cd) Dodawanie wybranych znaczników do strony

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      >
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <h:graphicImage  id="mapImage"
                         url="/resources/obrazy/Desert.jpg"/>
        <h:graphicImage id="mapImage" name="Desert.jpg"
                        library="obrazy" alt="Wybór obrazu"/>
        <h:graphicImage value="#{resource['obrazy:Desert.jpg']}"/>
    </h:body>
</html>
```

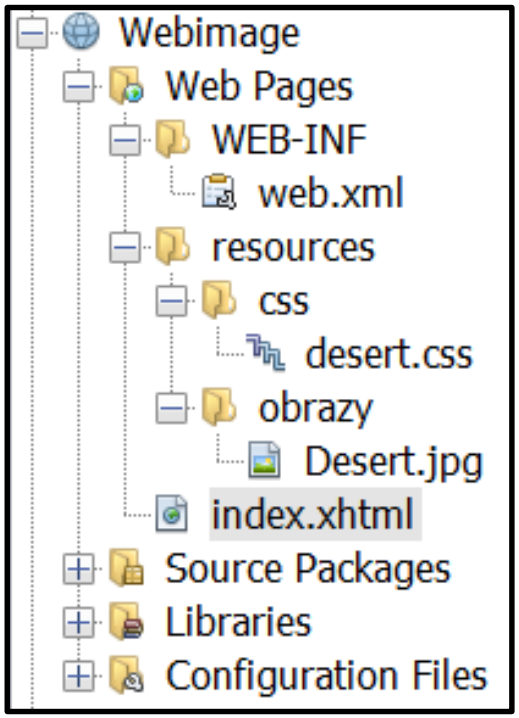


(8 cd) Dodawanie wybranych znaczników do strony

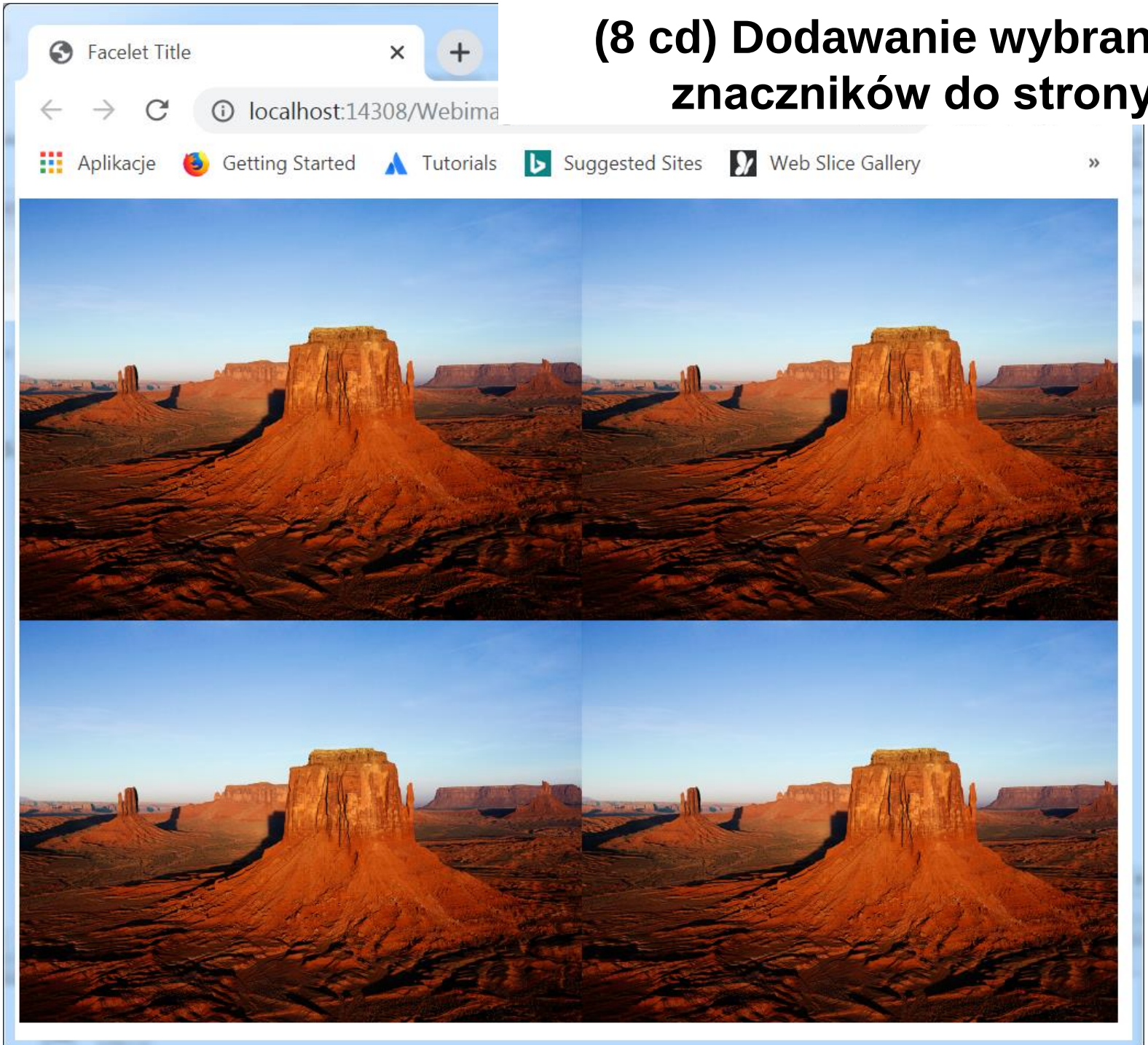


(8 cd) Dodawanie wybranych znaczników do strony

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      >
    <h:head>
        <h:outputStylesheet library="css" name="desert.css"/>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <header>
        </header>
    </h:body>
</html>
```



(8 cd) Dodawanie wybranych znaczników do strony



(9) Znaczniki określające ułożenie elementów - Przykład 8

h:panelGrid – atrybuty:
columns, columnClasses, footerClass,
headerClass, panelClass, rowClasses - **arkusze stylów**
layout – **wyświetla wiersze tabeli**
h:panelGroup

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

Strona główna – **index.xhtml**. Umożliwia ona uruchomienie strony **dodaj_produkt.xhtml za pomocą znacznika h:link**

Strona do dodawania danych produktu – **dodaj_produkt.xhtml**

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

<h:body>
  <h:form>
   <h:panelGrid columns="2">
    <h:outputLabel value="Podaj nazwe produktu"  for="nazwa" />
    <h:inputText   id="nazwa"    title="Podaj nazwe:" value="#{managed_produkt.nazwa} "
                          required="true"  requiredMessage="Blad: Podaj nazwe." >
     </h:inputText>
     <h:outputLabel value="Podaj cene netto produktu" for="cena" />
     <h:inputText    id="cena"     title="Podaj cene:"  value="#{managed_produkt.cena} "
                           required="true"  requiredMessage ="Blad: Podaj promocje.">
     <h:outputLabel value="Podaj promocje produktu" for="promocja" />
     <h:inputText   id="promocja"  title="Podaj promocje:" ="Blad: Podaj promocje." >
                         value="#{managed_produkt.promocja} "
                         required="true"  requiredMessage="Blad: Podaj promocje." >
     </h:inputText>
    </h:panelGrid>
```

Siatka **panelGrid** umożliwiająca wprowadzanie danych produktu do obiektu typu Managed_produkt w dwóch kolumnach za pomocą komponentów **outputLabel** oraz **inputText**.
Atrybuty **required** i **requiredMessage** obsługują błąd wynikający z braku wprowadzenia danych do komponentów typu **inputText**

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

Znacznik **<h:commandLink** pozwala przetworzyć wprowadzone dane za pomocą metody **dodaj_produk**t i powrócić do strony, której nazwę zwraca metoda **dodaj_produk**t z obiektu klasy **Managed_produk**t (wartość atrybutu **action**) – jest to strona **rezultat.xhtml**:

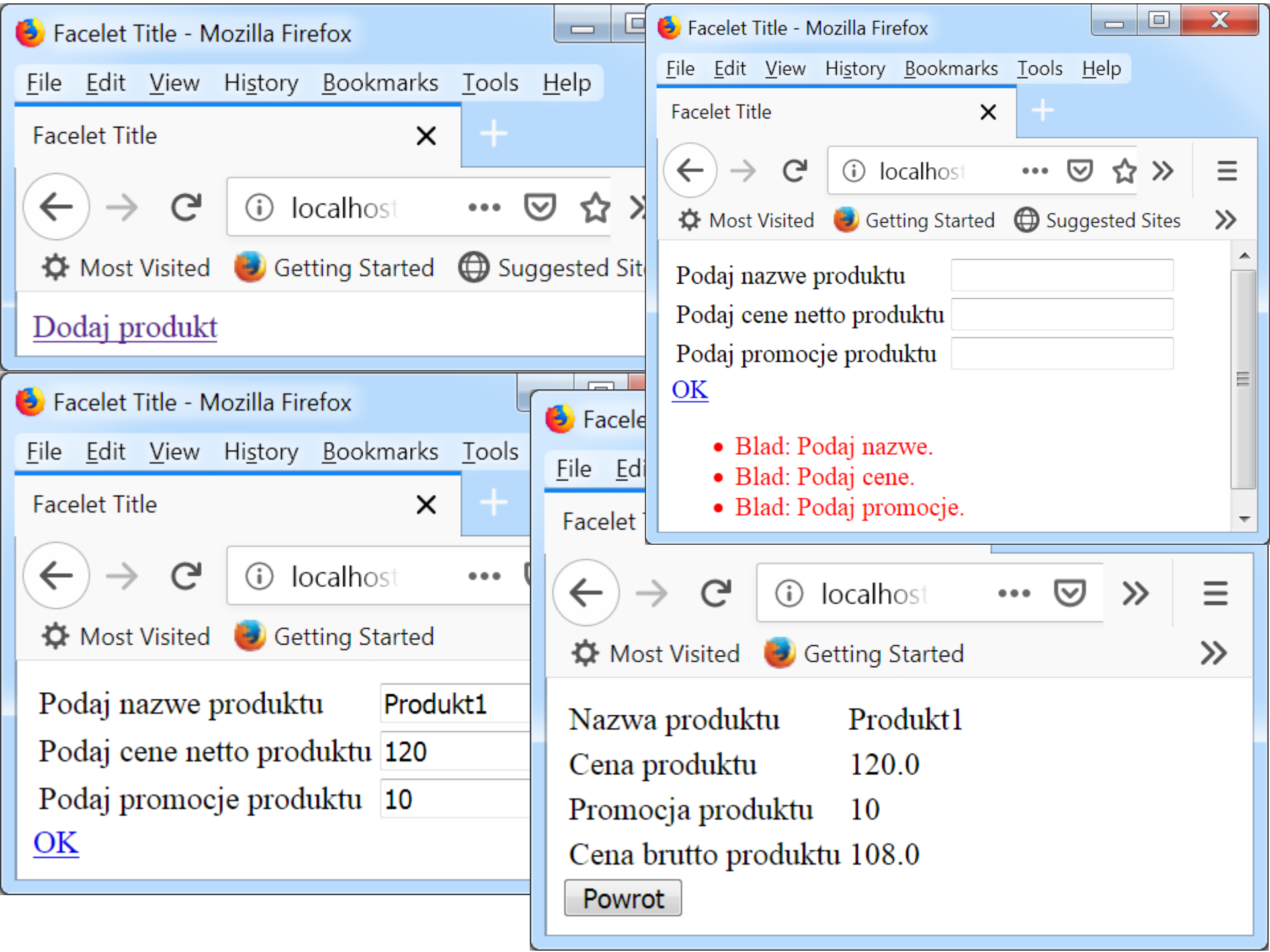
```
public String dodaj_produkt() {
        String[] dane = {nazwa, cena, promocja};
        fasada.utworz_produkt(dane);
        dane_produktu();
        return "rezultat";  }
```

Strona z formularzem do wyświetlania danych produktu - **rezultat.xhtml**

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <h:form>
            <h:panelGrid columns="2">
                <h:outputLabel value="Nazwa produktu"  for="nazwa" />
                <h:outputText id="nazwa" value="#{managed_produkt.nazwa}"/>
                <h:outputLabel value="Cena produktu"  for="cena" />
                <h:outputText id="cena" value="#{managed_produkt.cena}"/>
                <h:outputLabel value="Promocja produktu"  for="promocja" />
                <h:outputText id="promocja" value="#{managed_produkt.promocja}"/>
                <h:outputLabel value="Cena brutto produktu"  for="brutto" />
                <h:outputText id="brutto" value="#{managed_produkt.cena_brutto}" />
            </h:panelGrid>
            <h:commandButton id="powrot" value="Powrot" action="/faces/index"/>
        </h:form>
    </h:body>
</html>
```

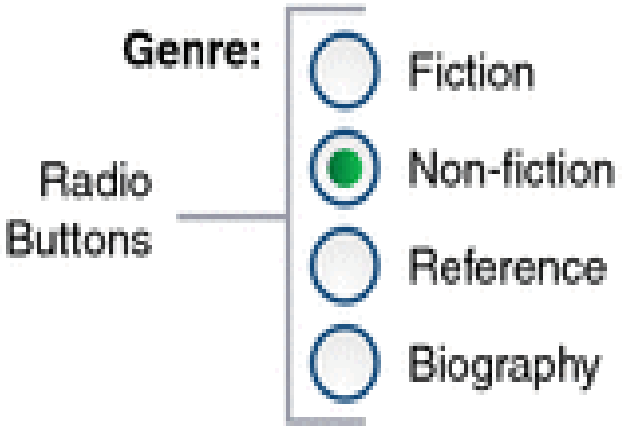
Siatka **panelGrid** umożliwiająca wyświetlanie danych produktu pobranych z obiektu typu **Managed_produkt** w dwóch kolumnach za pomocą komponentów **outputLabel** i **outputText**

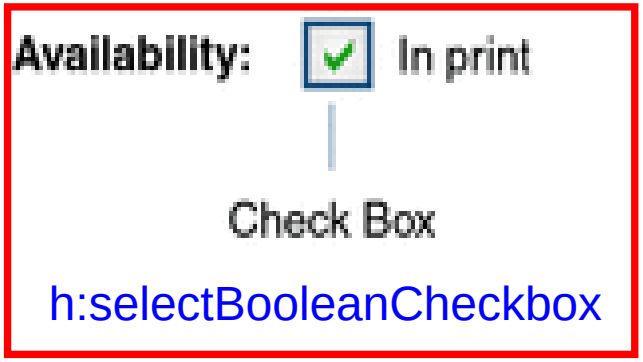
Znacznik **<h:commandButton** pozwala powrócić do strony głównej **index** (wartość atrybutu **action**)


Facelet Title - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Facelet Title
localhost
Most Visited Getting Started Suggested Sites
Dodaj produkt
Facelet Title - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Facelet Title
localhost
Most Visited Getting Started Suggested Sites
Podaj nazwe produktu
Podaj cene netto produktu
Podaj promocje produktu
OK
Blad: Podaj nazwe.
Blad: Podaj cene.
Blad: Podaj promocje.
Facelet Title - Mozilla Firefox
File Edit View History Bookmarks Tools
Facelet Title
localhost
Most Visited Getting Started
Podaj nazwe produktu Produkt1
Podaj cene netto produktu 120
Podaj promocje produktu 10
OK
Facele
File Edi
Facelet
localhost
Most Visited Getting Started
Nazwa produktu Produkt1
Cena produktu 120.0
Promocja produktu 10
Cena brutto produktu 108.0
Powrot

(10)Znaczniki wyświetlające komponenty wyboru jednej opcji

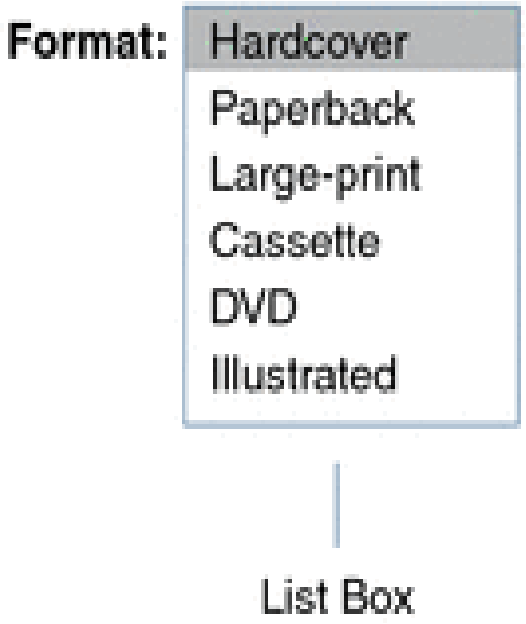
h:selectOneRadio





h:selectBooleanCheckbox

h:selectOneMenu



h:selectOneListbox

Zastosowanie zagnieżdżonych znaczników wyboru opcji f:selectItem w komponentach wyboru

Zalety f:selectItem

- dane z listy są definiowane z danych podanych na stronie
- niewiele kodu należy umieścić w ziarnie związanym z komponentem

Przykład wyświetlania rezultatów wyboru (ComboBox, drop-down list)

```
<h:selectOneMenu id="shippingOption" required="true"
                 value="#{cashier.shippingOption}">
  <f:selectItem itemValue="2" itemLabel="#{bundle.QuickShip}"/>
  <f:selectItem itemValue="5" itemLabel="#{bundle.NormalShip}"/>
  <f:selectItem itemValue="7" itemLabel="#{bundle.SaverShip}"/>
</h:selectOneMenu>
```

Atrybut **value** jest zbindowany z właściwościa ziarna, która przechowuje aktualnie wybraną pozycję reprezentowaną przez **itemValue** lub pierwszą, jeśli nie dokonano wyboru. Atrybut **itemLabel** służy do wyświetlania pozycji wyboru.

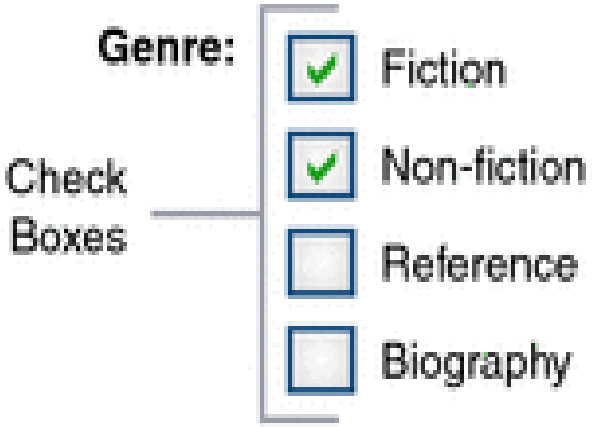
Pozostałe znaczniki definiuje się podobnie.

(11)Komponenty wyświetlające komponenty wyboru wielu opcji

h:selectManyCheckbox – wyświetlany jako zbiór check box

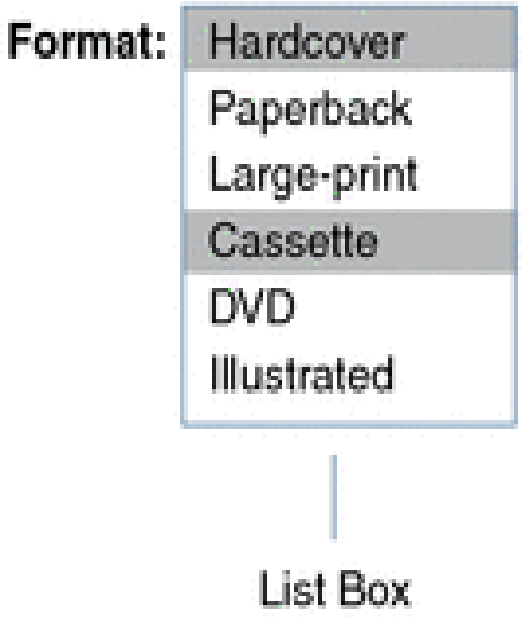
h:selectManyListbox - wyświetlany jako drop-down menu

h:selectManyMenu – wyświetlany jako list box



h:selectManyCheckbox

h:selectManyListbox



h:selectManyMenu

Zastosowanie zagnieżdżonych znaczników wyboru opcji f:selectItems w komponentach wyboru

Znaczniki reprezentują komponenty zagnieżdżane w innych komponentach służących do wyboru jednego (**f:selectItem**) lub kilku elementów (**f:selectItems**).

Zalety f:selectItems:

- są reprezentowane przez różne typy pojemników: Array, Map oraz Collection, zawierających elementy jako zwykłe obiekty Javy (POJO – Plain Old Java Object)
- można łączyć listy różnych komponentów w jeden komponent
- wartości komponentu mogą być generowane dynamicznie podczas działania programu

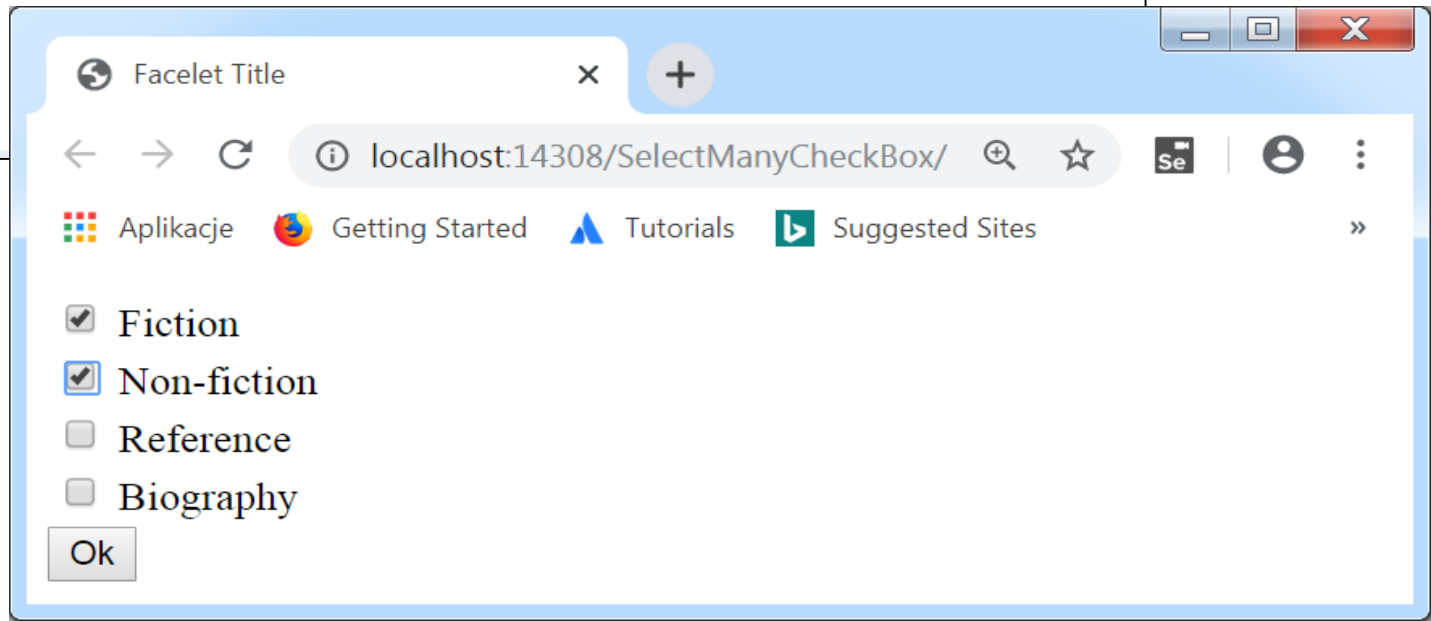
Po zastosowaniu znacznika h:selectManyCheckbox

```
<h:selectManyCheckbox  id="newslettercheckbox"
                       layout="pageDirection"
                       value="#{cashier.newsletters}">
        <f:selectItems value="#{cashier.newsletterItems}"/>
 </h:selectManyCheckbox>
 <h:outputText value="#{bundle.NewsletterThanks}"
               rendered="#{!empty  cashier.newsletters}"/>
  <ul>
   <ui:repeat value="#{cashier.newsletters}" var="nli">
      <li><h:outputText value="#{nli}" /></li>
    </ui:repeat>
 </ul>
```

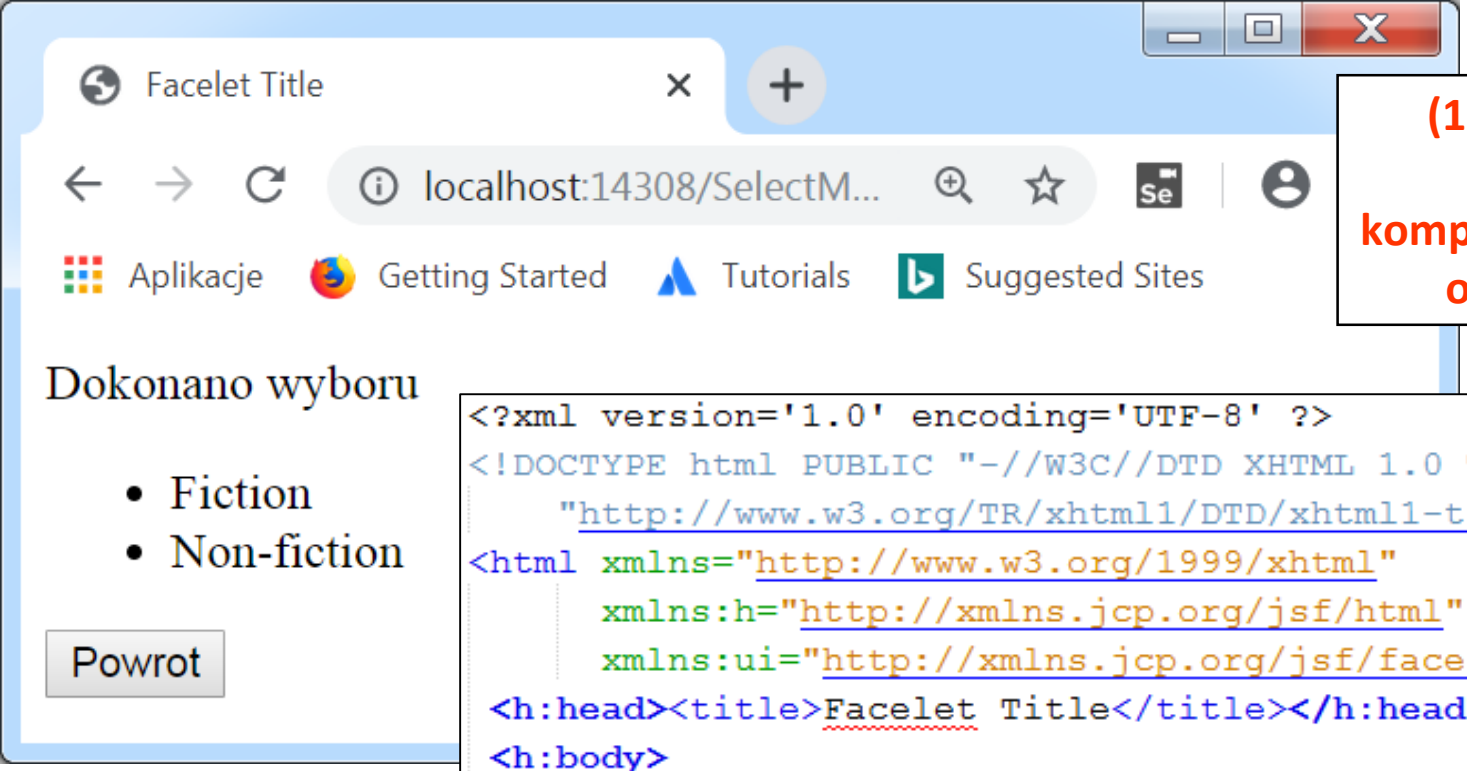
Elementy kolekcji **cashier.newsletterItems** są generowane programowo.
Blok wyboru jest wyświetlany przez znacznik **h:selectManyCheckbox** Atrybut **value** znacznika **h:selectManyCheckbox** jest zbindowany z właściwością ziarna, która przechowuje aktualnie wybrane pozycje ze zbioru reprezentowanego przez **f:selectItems value** lub pierwszą, jeśli nie dokonano wyboru. Wybrane pozycje są wyświetlane w znaczniku **ui:repeat** oraz komunikat w znaczniku **h:outputText**, gdy zbiór wybranych pozycji nie jest pusty (atrybut **rendered**).

(11 cd)Komponenty wyświetlające komponenty wyboru wielu opcji - przykład

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
 <h:head><title>Facelet Title</title></h:head>
 <h:body>
   <h:form>
     <h:selectManyCheckbox  id="newslettercheckbox"
        layout="pageDirection" value="#{wybor.wybraneopcje}">
       <f:selectItems value="#{wybor.opcjewyboru}"/>
     </h:selectManyCheckbox>
     <h:commandButton  id="ok" value="Ok" action="rezultat">
     </h:commandButton>
   </h:form>
 </h:body>
</html>
```



(11 cd)Komponenty wyświetlające komponenty wyboru wielu opcji – przyklad cd




```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
 <h:head><title>Facelet Title</title></h:head>
 <h:body>
  <h:form>
    <h:outputText value="Dokonano wyboru"
                  rendered="#{!empty wybor.wybraneopcje}"/>
     <ul>
       <ui:repeat value="#{wybor.wybraneopcje}" var="nli">
         <li><h:outputText value="#{nli}" /></li>
       </ui:repeat>
     </ul>
     <h:commandButton id="Powrot" value="Powrot" action="index">
     </h:commandButton>
   </h:form>
  </h:body>
</html>
```

(11 cd)Komponenty wyświetlające komponenty wyboru wielu opcji – przyklad cd

```
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import javax.faces.model.SelectItem;

@Named(value = "wybor")
@RequestScoped
public class Wybor {
    private String[] wybraneopcje;
    private  SelectItem[] opcjewyboru = {
        new SelectItem("Fiction"),
        new SelectItem("Non-fiction"),
        new SelectItem("Reference"),
        new SelectItem("Biography")
    };
    public Wybor() { }
    public SelectItem[] getOpcjewyboru() {return opcjewyboru;}
    public void setOpcjewyboru(SelectItem[] opcjewyboru) {
        this.opcjewyboru = opcjewyboru;     }
    public String[] getWybraneopcje()     {return wybraneopcje;}
    public void setWybraneopcje(String[] wybraneopcje) {
        this.wybraneopcje = wybraneopcje; }
}
```

(12) Zastosowanie komponentu h:dataTable

Komponent pozwala wyświetlać dane powiązane relacyjnie w postaci tabeli. Wspiera on wyświetlanie kolekcji obiektów reprezentujących dane aplikacji (atrybut **value,** gdzie atrybut **var** deklaruje obiekt tej kolekcji). Znacznik **h:column** reprezentuje kolumnę tabeli z danymi, uzyskanymi w wyniku iteracji po każdym rekordzie danych (atrybuty obiektu, elementy tablicy itp. deklarowanych w **var**) w źródle danych, które są wyświetlane w wierszach tabeli.

```
<h:dataTable id="items"
  captionStyle="font-weight:bold"
  columnClasses="list-column-center, list-column-left, list-column-right, list-column-center "
  footerClass="list-footer"
  headerClass="list-header"
  rowClasses="list-row-even, list-row-odd"
  styleClass="list-background"
  summary="#{bundle.ShoppingCart}"
  value="#{cart.items}"
  border="1"
  var="item">
```

Atrybuty z przyrostkiem w nazwie: Classes, Class oznaczają nazwy stylów prezentacji elementów tabeli: kolumn, stopki, nagłówka, wierszy, tła

Atrybut **value** – odniesienie do zbioru danych, gdzie każda z danych jest deklarowana za pomocą atrybutu **var.**

```
<h:column>
  <f:facet name="header">
    <h:outputText value="#{bundle.ItemQuantity}" />
  </f:facet>
  <h:inputText id="quantity"
        size="4"
        value="#{item.quantity}"
        title="#{bundle.ItemQuantity}">
     <f:validateLongRange minimum="1"/>
   </h:inputText>
    <h:message for="quantity"/>
</h:column>
 <h:column>
    <f:facet name="header">
       <h:outputText value="#{bundle.ItemTitle}"/>
    </f:facet>
    <h:commandLink action="#{showcart.details}">
         <h:outputText value="#{item.title}"/>
    </h:commandLink>
</h:column>
```

atrybut **var** deklaruje rekord danych, gdzie jego składowe sa prezentowane w poszczególnych kolumnach każdego wiersza tabeli

Kolumna z przyciskami do usuwania wiersza za pomocą metody details

```
……..
<f:facet name="footer"
  <h:panelGroup>
      <h:outputText value="#{bundle.Subtotal}"/>
      <h:outputText value="#{cart.total}" />
            <f:convertNumber currencySymbol="$" type="currency" />
       </h:outputText>
   </h:panelGroup>
 </f:facet>
 <f:facet name="caption">
      <h:outputText value="#{bundle.Caption}"/>
 </f:facet>
</h:dataTable>
```

W tabeli wyświetlane są dane książek w księgarni: liczba kupionych książek w kartach płatniczych, ceny i przyciski pozwalające na usunięcie książek z karty płatniczej.

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt1	100.0	10	Sat Sep 28 02:00:00 CEST 2019	90.0
2	Produkt2	200.0	10	Sun Oct 20 02:00:00 CEST 2019	180.0

Powrot

```
<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">
```

Atrybuty opcjonalne	Zdefiniowane style	Przykłady stylów
captionClass	Tytuł tabeli	
columnClasses	Kolumny tabeli	list-column-center i list-column-right
footerClass	Stopka	
headerClass	Nagłówek	
rowClasses	**Wiersze**	
styleClass	Wygląd całej tabeli	

Dane wyświetlane w komponencie dataTable

- Lista ziaren (beans) `<h:dataTable value="#{managed_produkt.items}" var="item"`
- Tablica ziaren
- Pojedyncze ziarno
- Obiekt typu javax.faces.model.DataModel
- Obiekt java.sql.ResultSet
- Obiekt javax.servlet.jsp.jstl.sql.Result
- Obiekt javax.sql.RowSet.

Komponent może wyświetlić wszystkie dane lub ich podzakres określając granice za pomocą atrybutów first i rows

Id produktu	Nazwa produktu	Cena netto produktu	Promocja produktu	Data produkcji	Cena brutto
1	Produkt1	100.0	10	Sat Sep 28 02:00:00 CEST 2019	90.0

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

(13) Wyświetlanie wiadomości o błędach konwersji i walidacji za pomocą znaczników h:message i h:messages

```
<p> <h:inputText id="userNo"
                 title="Type a number from 0 to 10:"
                 value="#{userNumberBean.userNumber}">
    <f:validateLongRange minimum="#{userNumberBean.minimum}”
                         maximum="#{userNumberBean.maximum}"/>
  </h:inputText>
  <h:commandButton id="submit" value="Submit"  action="response"/> </p>
 <h:message showSummary="true" showDetail="false"
            style="color: #d20005;
            font-family: 'New Century Schoolbook', serif;
            font-style: oblique;
            text-decoration: overline"
            id="errors1"
            for="userNo"/>
```

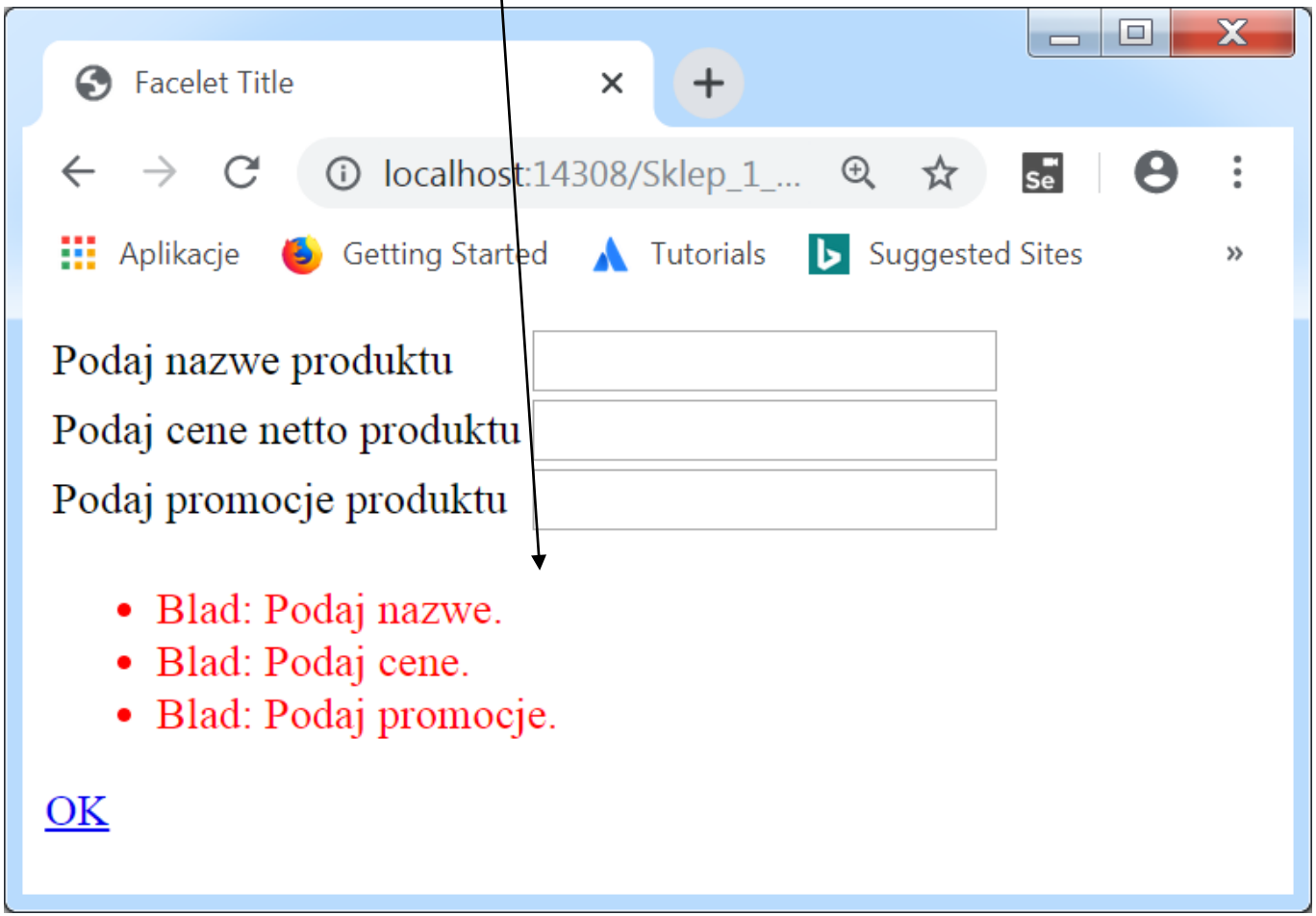
Wiadomośc o błędach za pomoca znacznika **h:message** wyświetla się za przyciskiem **Submit** na stronie i dotyczy wszystkich błędów jednego komponentu.

Przykład 9 – grupowanie wiadomości (kontynuacja przykładu 8)

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
     <h:form>
       <h:panelGrid columns="2">
        <h:outputLabel value="Podaj nazwe produktu"  for="nazwa" />
         <h:inputText
           id="nazwa" title="Podaj nazwe:" value="#{managed_produkt.nazwa}"
           required="true" requiredMessage="Blad: Podaj nazwe." >
         </h:inputText>
         <h:outputLabel value="Podaj cene netto produktu" for="cena" />
         <h:inputText
           id="cena" title="Podaj cene:" value="#{managed_produkt.cena}"
           required="true"  requiredMessage="Blad: Podaj cene." >
         </h:inputText>
         <h:outputLabel value="Podaj promocje produktu" for="promocja" />
         <h:inputText
          id="promocja" title="Podaj promocje:" value="#{managed_produkt.promocja}"
          required="true" requiredMessage="Blad: Podaj promocje." >
         </h:inputText>
       </h:panelGrid>
       <h:panelGroup id="messagePanel" layout="block">
           <h:messages errorStyle="color: red" infoStyle="color: green" />
       </h:panelGroup>
       <h:commandLink  action="#{managed_produkt.dodaj_produkt}" value="OK" />
      </h:form>
    </h:body>
</html>
```

Znacznik **h:messages** wyświetla błędy wszystkich komponentów na stronie.

```
<h:panelGroup id="messagePanel" layout="block">
        <h:messages errorStyle="color: red" infoStyle="color: green" />
</h:panelGroup>
```



(14) Tworzenie odniesień typu URL za pomocą znaczników h:button i h:link

Znaczniki h:commandLink oraz h:commandButton pozwalają na prostszą definicję odniesień typu URL (zastosowanie żądań typu **POST**) – są używane do przesyłania bloków danych do serwera.

Znaczniki **h:button i h:link** pozwalają na definicję odniesienia za pomocą kilku atrybutów typu **name** po znaku **?** i zakończone znakiem separatora **&;** (zastosowanie żądań typu **GET**).

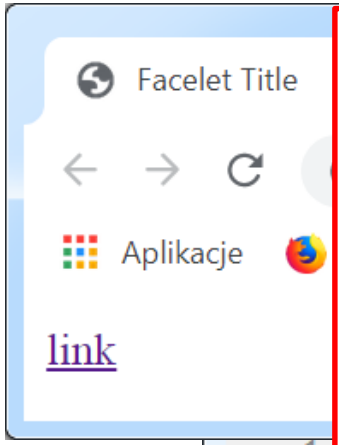
Przykład:

```
<h:link outcome="somepage" value="Message" />
```

jest renderowana na znacznik html <a>:

```
<a href="/simplebookmark/faces/somepage.xhtml">
    Message</a>
```

(14) Tworzenie odniesień typu URL za pomocą znaczników h:button i h:link




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

```
<input type="hidden" name="j_idt5" value="j_idt5" />
<a href="/Witaj_w4_url/faces/rezultat.xhtml">link</a>
<input type="hidden" name="javax.faces.ViewState"
id="j_id1:javax.faces.ViewState:0"
value="-7896283184339793440:-43296511769911911205"
autocomplete="off" />
</form></body>
</html>
```

(15) Używanie parametrów do konfigurowania odniesienia URL

```
<h:body>
  <h:form>
    <h:graphicImage url="duke.waving.gif" alt="Duke waving his hand"/>
    <h2>Hello, #{hello.name}!</h2>
    <p>I've made your
      <h:link outcome="personal" value="personal greeting page!"
              includeViewParams="true">
            <f:param name="Result" value="#{hello.name}"/>
      </h:link>
    </p>
    <h:commandButton id="back" value="Back" action="index" />
  </h:form>
</h:body>
```

Efekt:

http://localhost:8080/bookmarks/faces/personal.xhtml?Result=Timmy

Używanie parametrów do konfigurowania odniesienia URL (cd)

Równoważna postać deklarowania widoczności wartości parametrów wyświetlanych w adresie URL na stronie internetowej

```
<f:metadata>
        <f:viewParam name="Result" value="#{hello.name}" />
</f:metadata>
```

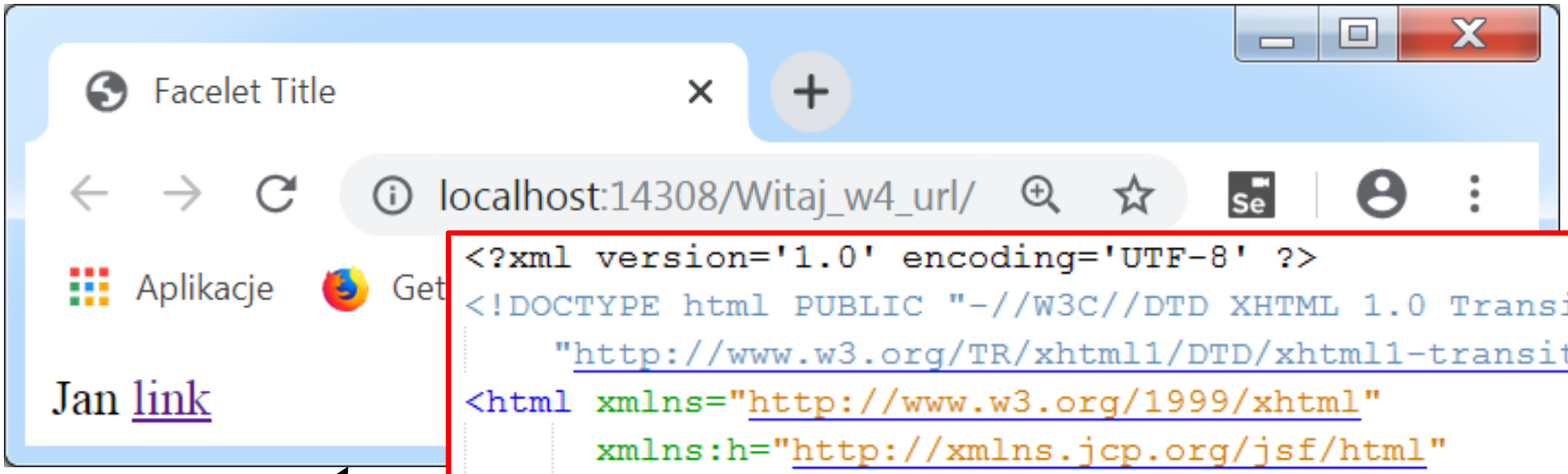
Można odwołać się do wartości właściwości ziarna **hello**

```
<h:outputText value="Howdy, #{hello.name}!" />
```

Efekt:

http://localhost:8080/bookmarks/faces/personal.xhtml?Result=Timmy

Używanie parametrów do konfigurowania odniesienia URL (cd)



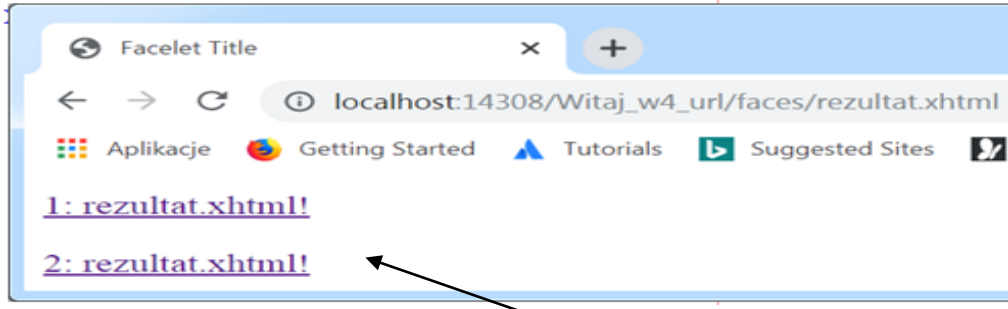
Strona index.xhtml

Definicja znacznika **<f:viewParam** z definicją parametrów wyświetlanych podczas powrotu na stronę **index.xhtml**

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <h:form>
            <f:metadata>
                <f:viewParam name="Result" value="Jan" >
                </f:viewParam>
            </f:metadata>
            <h:outputText value="Jan "/>
            <h:link  outcome="rezultat" value="link">
            </h:link>
        </h:form>
    </h:body>
</html>
```

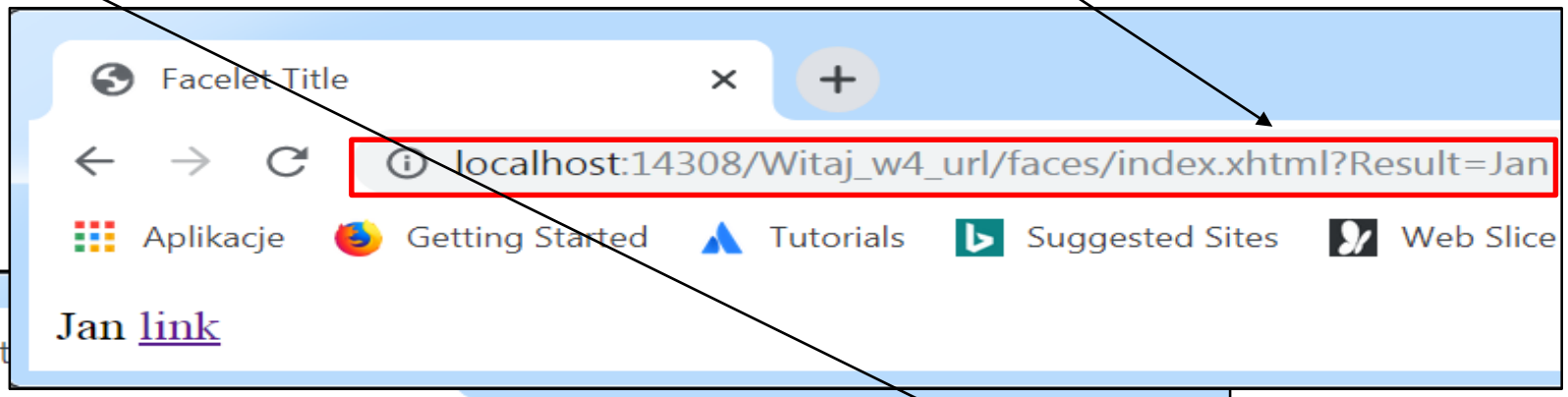
Używanie parametrów do konfigurowania odniesienia URL (cd)

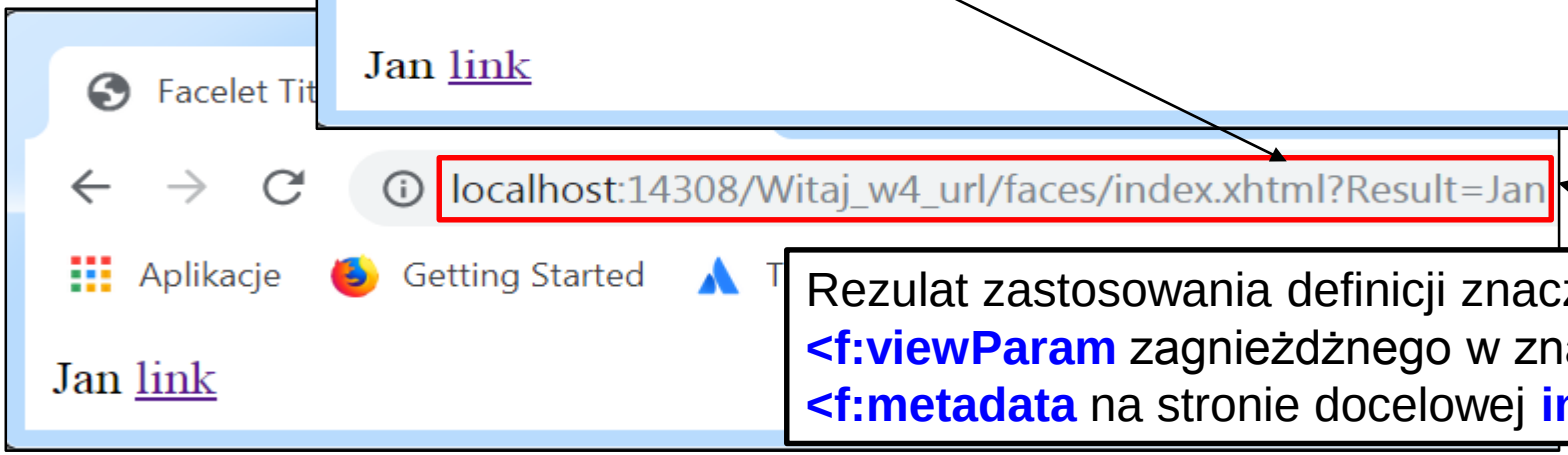
```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
 <h:head><title>Facelet Title</title></h:head>
 <h:body>
  <h:form>
   <h:link outcome="index" value="1: rezultat.xhtml!"
           includeViewParams="true">
    <f:param name="Result" value="Jan"/>
   </h:link>
   <p></p>
   <h:link outcome="index" value="2: rezultat.xhtml!"
           includeViewParams="true">
   </h:link>
  </h:form>
 </h:body>
</html>
```



Strona rezultat.xhtml

Rezulat zastosowania parametrów zdefiniowanych w znaczniku **<f:param** ramach znacznika **<h:link**





Rezulat zastosowania definicji znacznika **<f:viewParam** zagnieżdżnego w znaczniku **<f:metadata** na stronie docelowej **index.xhtml**

(16) Relokacja zasobów za pomocą znaczników h:outputScript i h:outputStylesheet

Przykład a

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://xmlns.jcp.org/jsf/html">
 <h:head id="head">
    <title>Resource Relocation</title>
 </h:head>
 <h:body id="body">
    <h:form id="form">
      <h:outputScript name="hello.js"/>
      <h:outputStylesheet name="hello.css"/>
     </h:form>
 </h:body>
</html>
```

Brak atrybutu **target** powoduje, że podczas renderowania strony odniesienia do tych plików znajdą się w domyślnych miejscach:
body
head

Postać strony po renderowaniu:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
     <title>Resource Relocation</title>
     <link type="text/css" rel="stylesheet"
           href="/context-root/faces/javax.faces.resource/hello.css"/>
   </head>
   <body>
      <form id="form" name="form" method="post" action="..." enctype="...">
          <script type="text/javascript"
                  src="/context-root/faces/javax.faces.resource/hello.js">
           </script>
       </form>
    </body>
</html>
```

Przykład b

```
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head id="head">
      <title>Resource Relocation</title>
  </h:head>
   <h:body id="body">
          <h:form id="form">
              <h:outputScript name="hello.js" target="#{param.location}"/>
              <h:outputStylesheet name="hello.css"/>
           </h:form>
   </h:body>
</html>
```

Atrybut **target** powoduje, że podczas renderowania strony odniesienie do pliku **hello.js** znajdzie się w elemencie **head**

Postać strony po renderowaniu:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
        <title>Resource Relocation</title>
        <link type="text/css" rel="stylesheet"
              href="/context-root/faces/javax.faces.resource/hello.css"/>
        <script type="text/javascript"
             src="/context-root/faces/javax.faces.resource/hello.js">
        </script>
  </head>
  <body>
       <form id="form" name="form" method="post" action="..." enctype="...">
        </form>
  </body>
</html>
```

Uwaga: użycie adnotacji @ResourceDependency zwalnia programistów stron JSF z definiowania lokacji zasobów

np

```
@ResourceDependency(name="jsf.js", library="javax.faces", target="head")
```

Opis znaczników obsługiwanych przez Facelets

rozdział 8 i rozdziały 32 - 33 (komponenty EJB)

- Znaczniki typu UI
- Znaczniki do tworzenia szablonów

Szablony JavaServer Faces

- Pozwalają w łatwy sposób rozszerzać interfejs użytkownika
- Zapewniają wielożywalność elementów interfejsu użytkownika
- Zapobiegają budowanie podobnie skonstruowanych stron
- Ułatwiają wprowadzanie standardów w budowie stron internetowych

Przegląd znaczników JSF - ui

Znacznik	Funkcja znaczników szablonu
ui:component	Tworzy podany komponent i dodaje do drzewa komponentów
ui:composition	**Definiuje kompozycję strony, która opcjonalnie używa szablonu. Zawartość poza znacznikiem jest ignorowana**
ui:debug	Definiuje debugowany komponent, który jest tworzony i dodany do drzewa komponentów
ui:decorate	**Podobny do znacznika ui:composition, ale nie pomija zawartości poza znacznikiem**
ui:define	**Reprezentuje formularz danych wejściowych, które mogą być przesłane razem do aplikacji**
ui:fragment	Podobny do znacznika **ui:component** , ale nie pomija zawartości poza znacznikiem
ui:include	Hermetyzuje i wprowadza wielużywalność zawartości do wielu stron
ui:insert	**Wstawia zawartość do szablonu strony**
ui:param	Używany do przekazywania parametrów do dodawanych plików
ui:repeat	Alternatywa znaczników pętli: **c:forEach, h:dataTable**
ui:remove	Usuwa zawartość strony

Przykład 10 - Szablon strony - szablon.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" />
      <title><ui:insert name="title">Default Title</ui:insert></title>
      <h:outputStylesheet name="css/jsf.css"/>
   </h:head>
   <h:body>
      <h1>
         <ui:insert name="title">Default Title</ui:insert>
      </h1>
      <p>
         <ui:insert name="body1">Default Body</ui:insert>
      </p>
      <p>
         <ui:insert name="body2">Default Body</ui:insert>
      </p>
   </h:body>
</html>
```

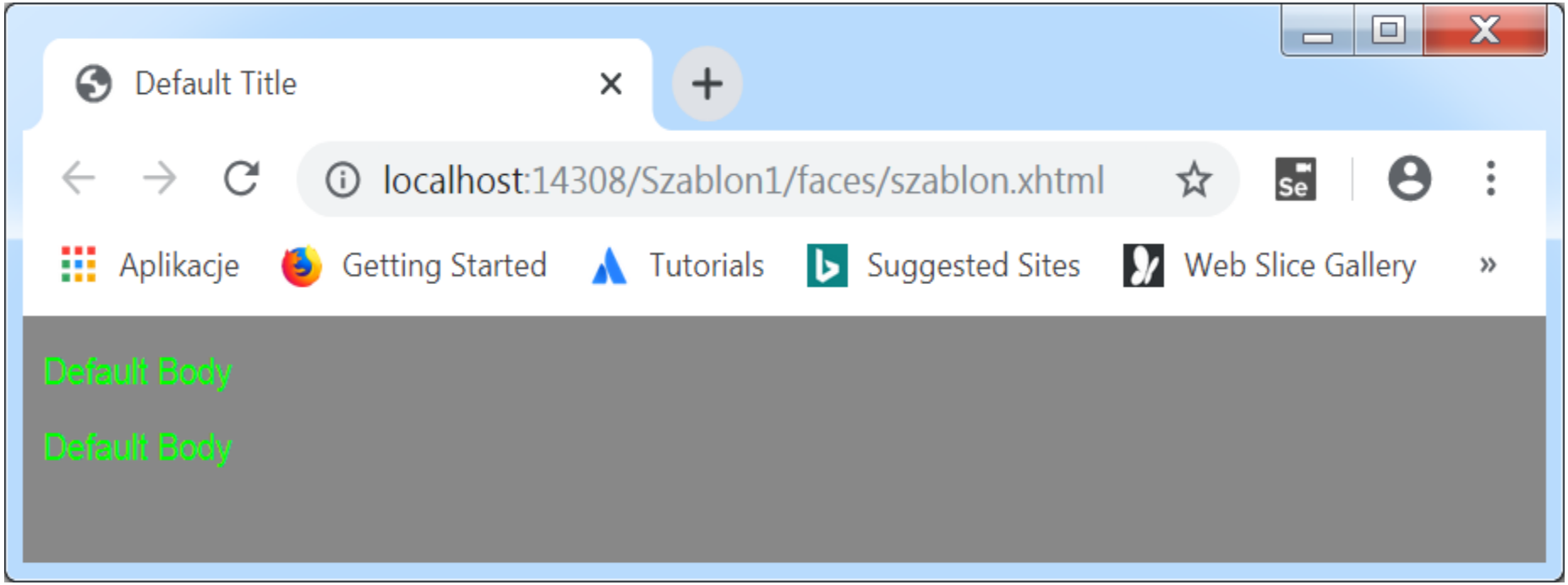
Dyrektywy przestrzeni nazw udostępniających biblioteki użytych typów znaczników

lub
<h:outputStylesheet library ="css" name="jsf.css"/>

Widok szablonu – szablon.xhtml

```
body {
    font-family: Arial, Helvetica, sans-serif;
    color: #00FF00;
    background-color: #888888;
    font-size: small;
}
```

Zawartość arkusza stylów **jsf.css**, zastosowany w szablonie

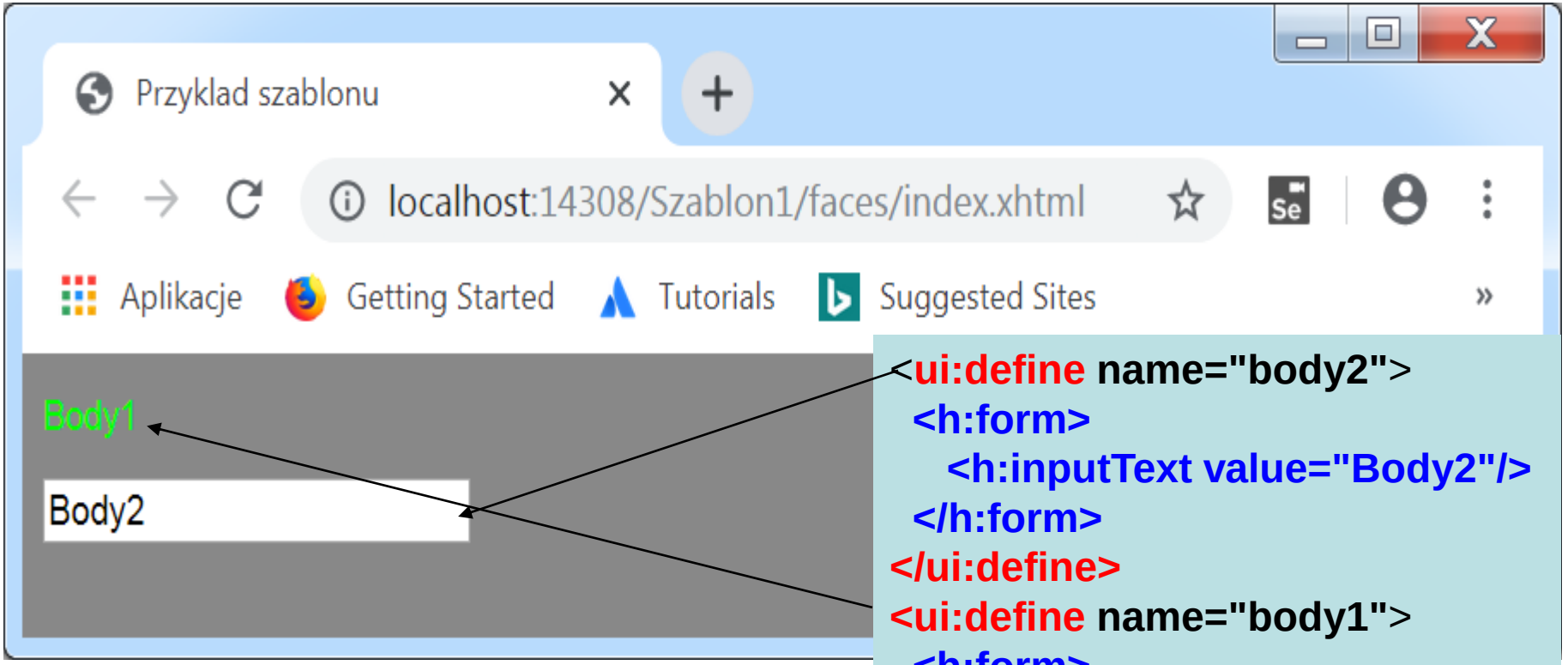


Strona zbudowana na szablonie szablon.xhtml. Rola znacznika ui:composition

```
<?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>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
      <ui:composition template="./szablon.xhtml">
       <ui:define name="title">          Przyklad szablonu
       </ui:define>
       <ui:define name="body2">
              <h:form> <h:inputText value="Body2"/> </h:form>
       </ui:define>
       <ui:define name="body1">
              <h:form>  <h:outputText value="Body1"/> </h:form>
       </ui:define>
      </ui:composition>
      <h:outputText value="Body1"/>
  </h:body>
</html>
```

Dyrektywy przestrzeni nazw udostępniających biblioteki użytych typów znaczników

Strona wykonana za pomocą szablonu szablon.xhtml brak komponentu <h:outputText value="Body1"/> z powodu użycia znacznika ui:composition



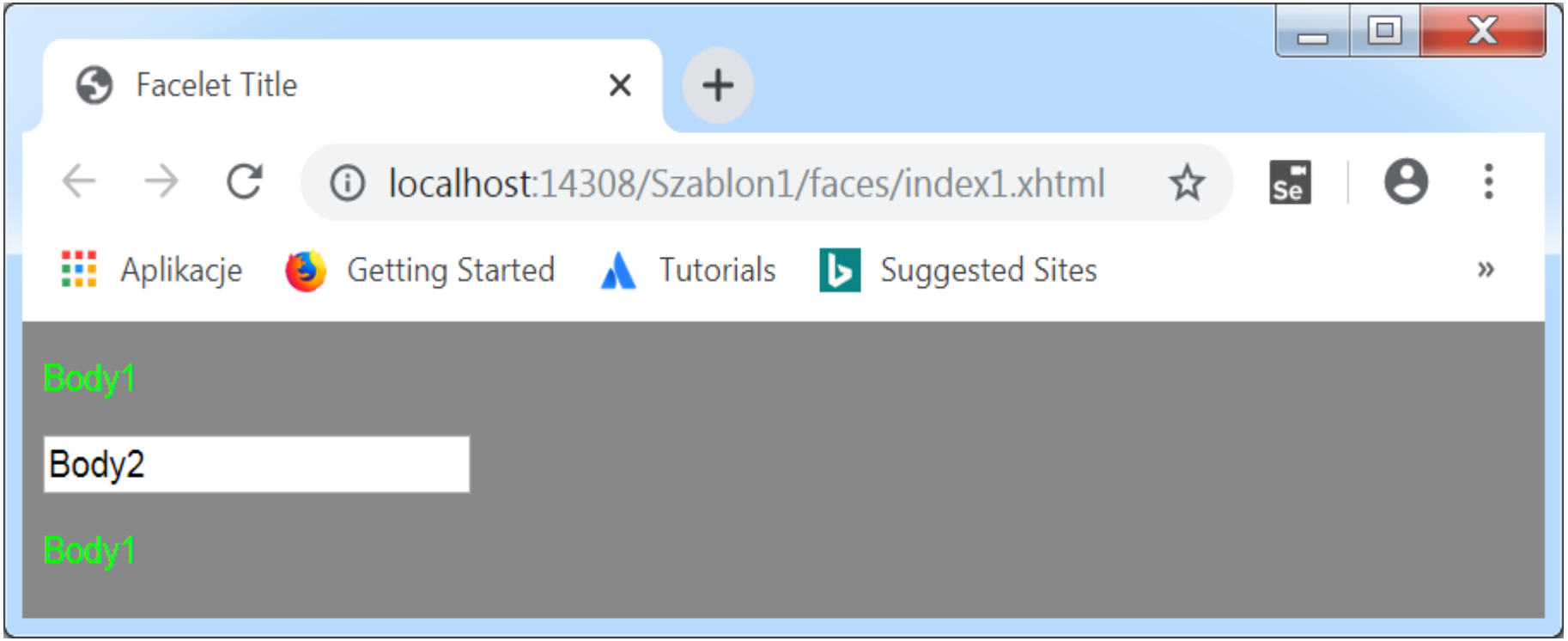

```
<ui:define name="body2">
  <h:form>
    <h:inputText value="Body2"/>
  </h:form>
</ui:define>
<ui:define name="body1">
  <h:form>
    <h:outputText value="Body1/>
  </h:form>
</ui:define>
```

Układ wynika z definicji podanej w szablonie ramach znacznika **ui:composition**

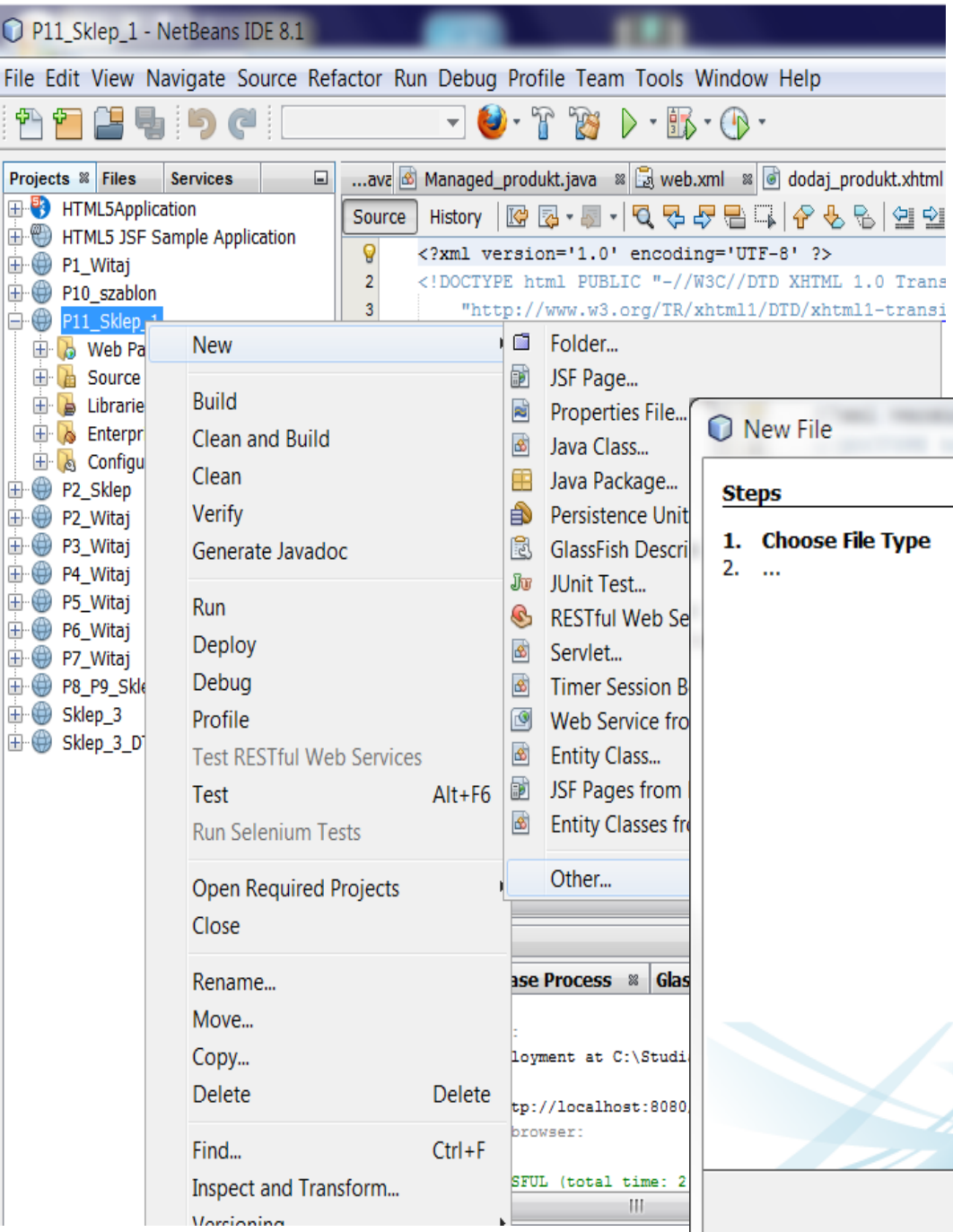
Przykład strony zbudowanej na szablonie szablon.xhtml ui:decorate

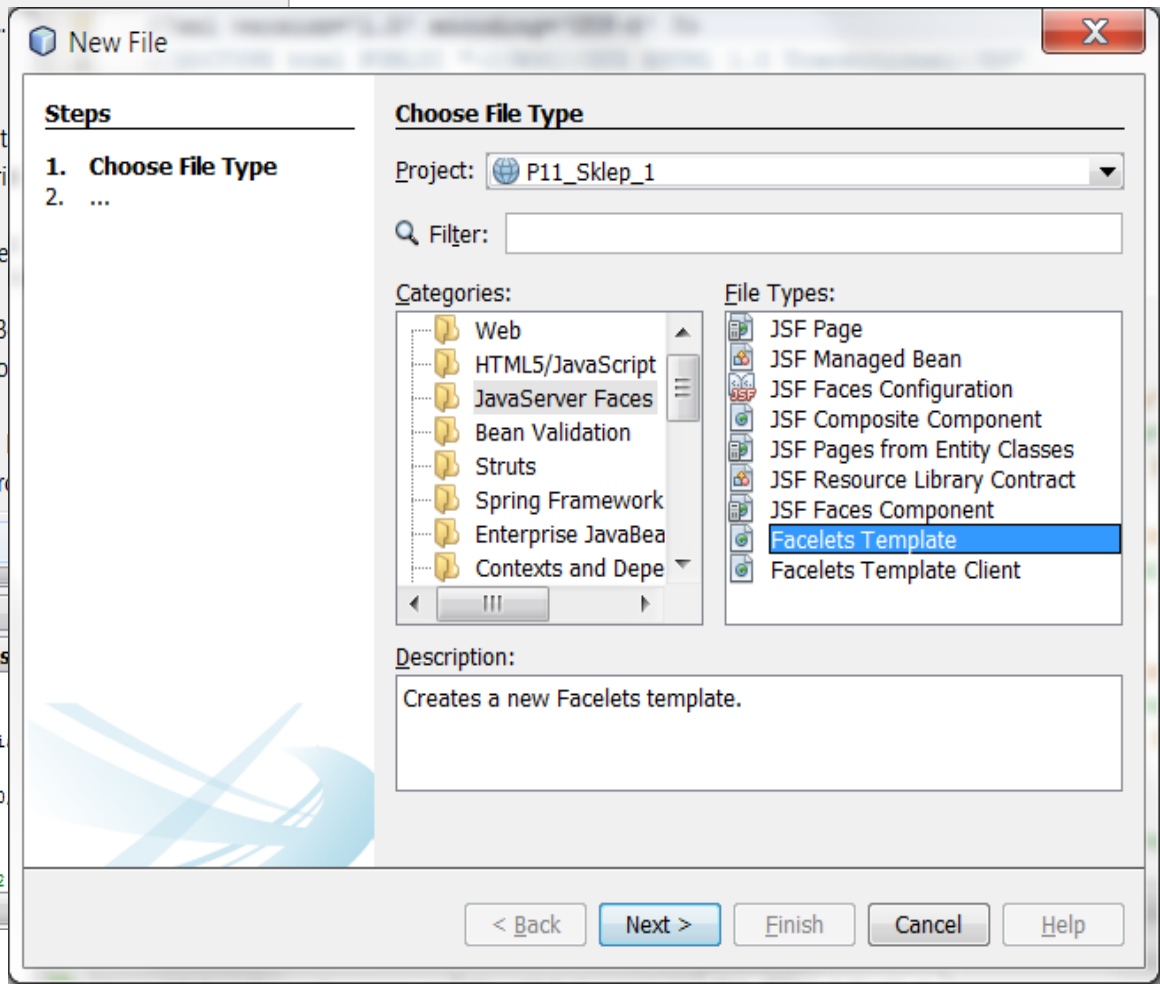
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
  <ui:decorate template="./szablon.xhtml">
   <ui:define name="title"> Przyklad szablonu
   </ui:define>
   <ui:define name="body2">
       <h:form>  <h:inputText value="Body2"/> </h:form>
   </ui:define>
   <ui:define name="body1">
       <h:form> <h:outputText value="Body1"/> </h:form>
    </ui:define>
  </ui:decorate>
    <h:outputText value="Body1"/>
 </h:body>
</html>
```

Strona wykonana za pomocą szablonu szablon.xhtml pojawił się komponent <h:outputText value="Body1"/> z powodu użycia znacznika ui:decorate

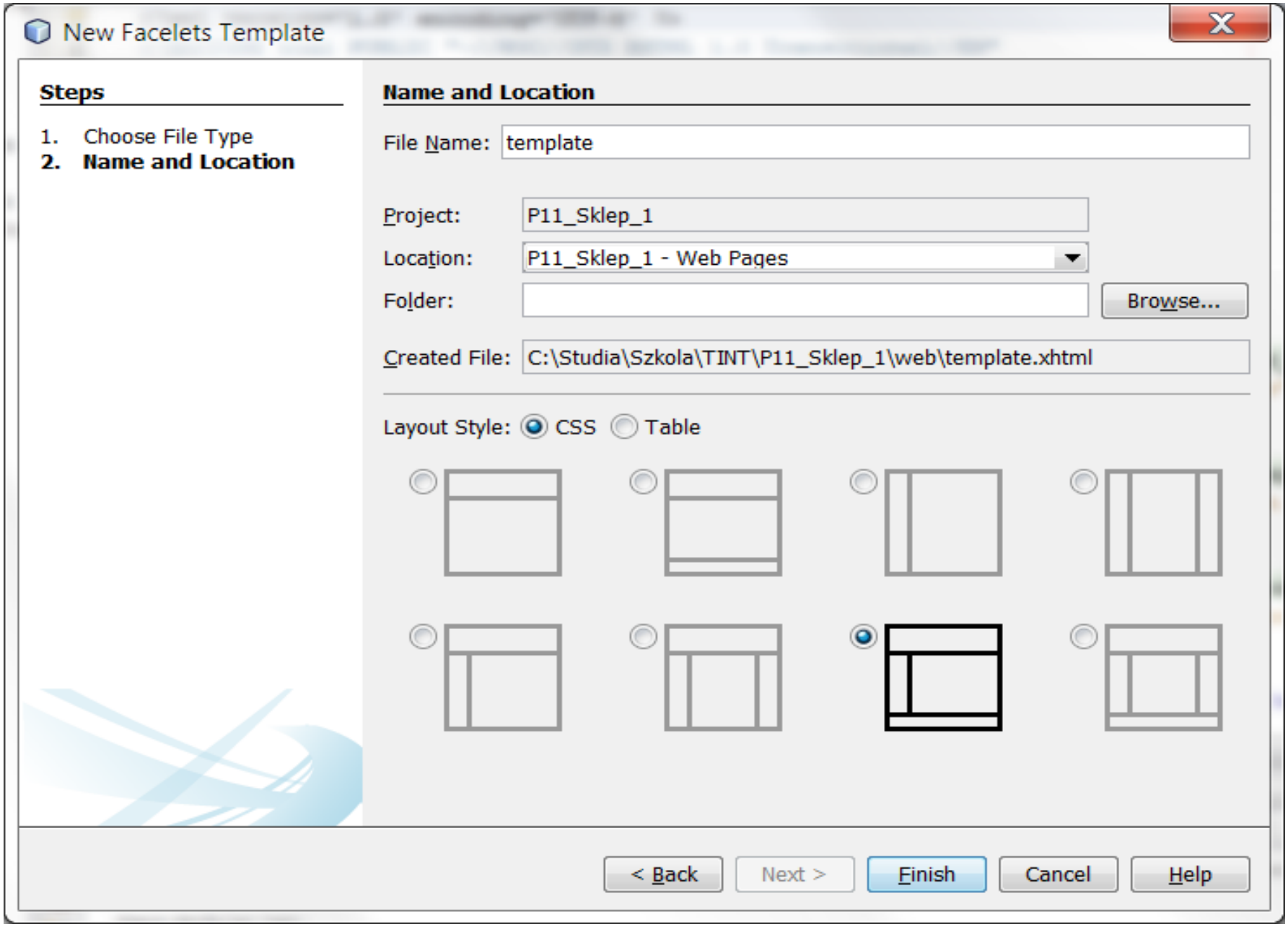


Szablony stron – przykład 11 (kopia programu z przykładu 9) – **Wstawianie szablonu:** należy kliknąć prawym klawiszem na nazwę projektu i wybrać: **New/Other/JavaServer Faces/Facelets Template**





Wybór szablonów stron



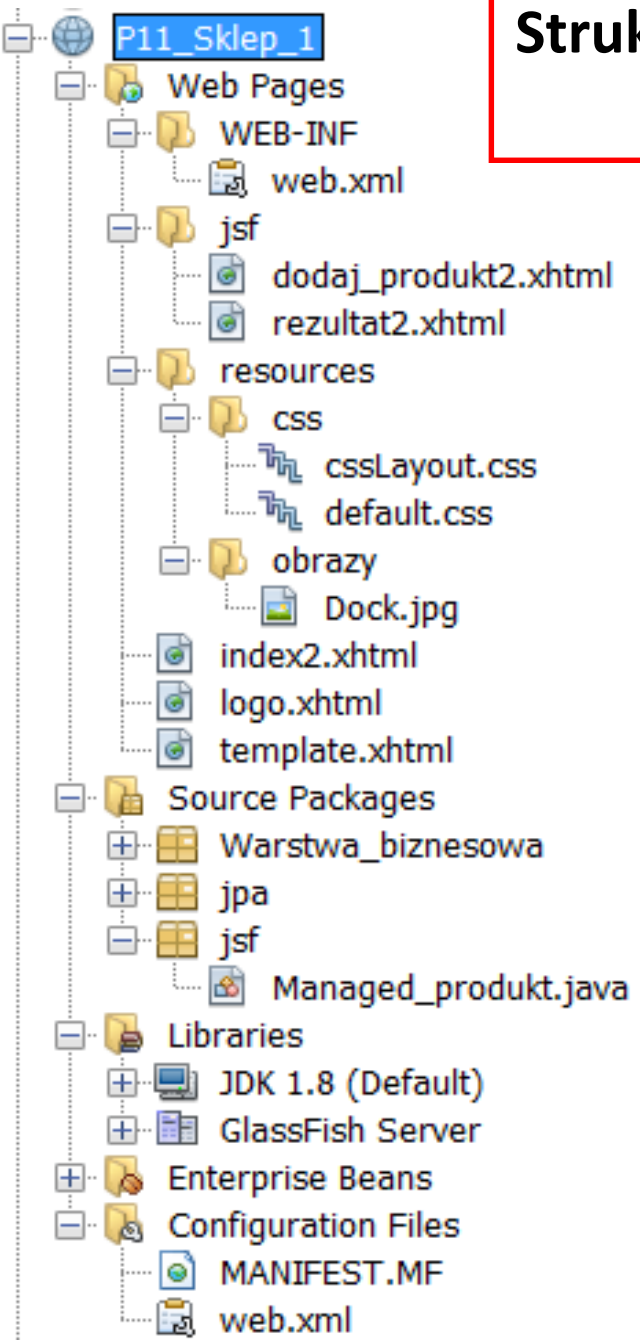
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <h:outputStylesheet name="css/default.css"/>
        <h:outputStylesheet name="css/cssLayout.css"/>
        <title>Facelets Template</title>
    </h:head>
    <h:body>
        <div id="top">
            <ui:insert name="top">Top</ui:insert>
        </div>
        <div>
            <div id="left">
                <ui:insert name="left">Left</ui:insert>
            </div>
            <div id="content" class="left_content">
                <ui:insert name="content">Content</ui:insert>
            </div>
        </div>
        <div id="bottom">
            <ui:insert name="bottom">Bottom</ui:insert>
        </div>
    </h:body>
</html>
```

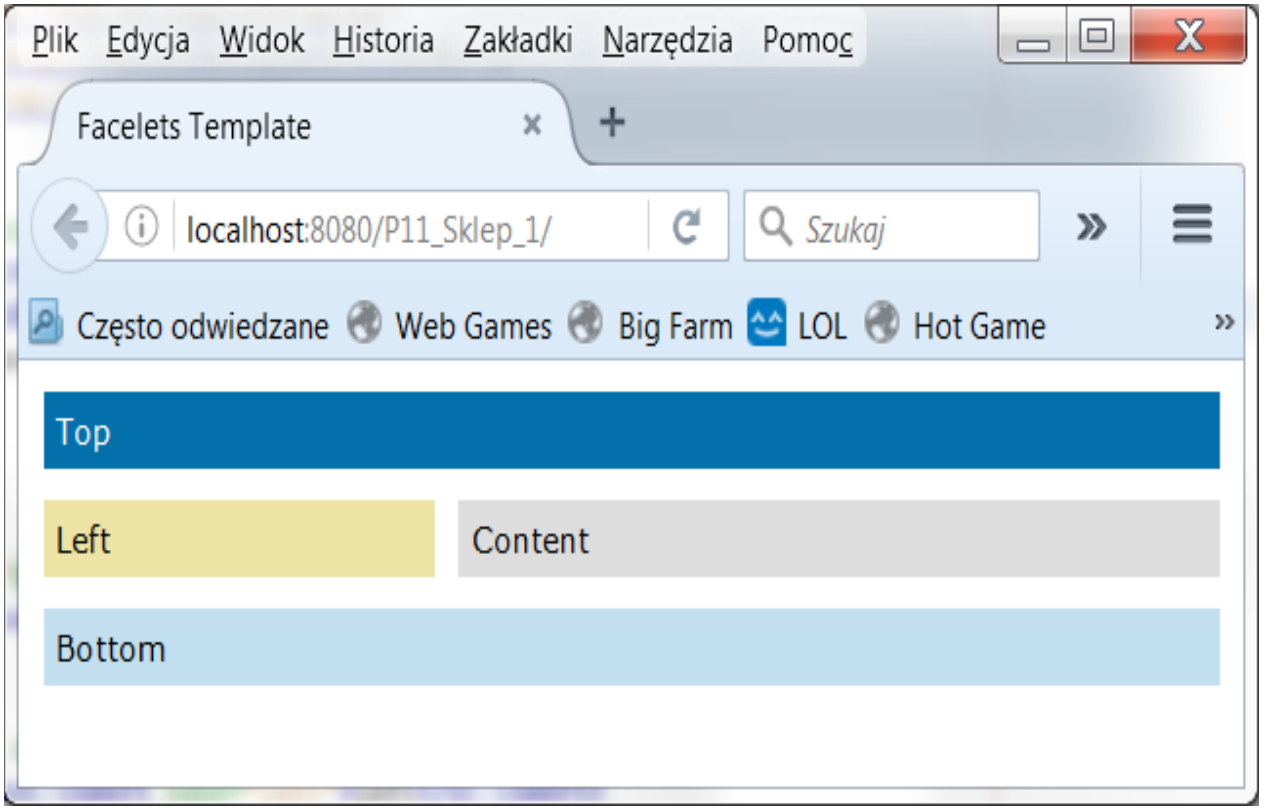
Wygenerowany szablon strony template.xml

Zmodyfikowane ścieżki:
name=./css/default.css
name=./css/cssLayout.css

Struktura projektu widoczna w zakładce Projects

Widok wygenerowanego szablonu strony





Plik szablonu template.xhtml po zmianach – znacznik ui:insert określa miejsca implementowane na stronach opartych na szablonie strony

```
<?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">
            <h:panelGroup>
                <ui:include src="./logo.xhtml" />
                <ui:insert name="top"></ui:insert>
            </h:panelGroup>
        </div>
```

Wprowadzenie do szablonu strony logo.xhtml za pomocą znacznika ui:include

```
<div>
   <div id="left">
    <h:link outcome="/faces/jsf/dodaj_produkt2"
                value="Dodaj produkt"/>
   </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>
```

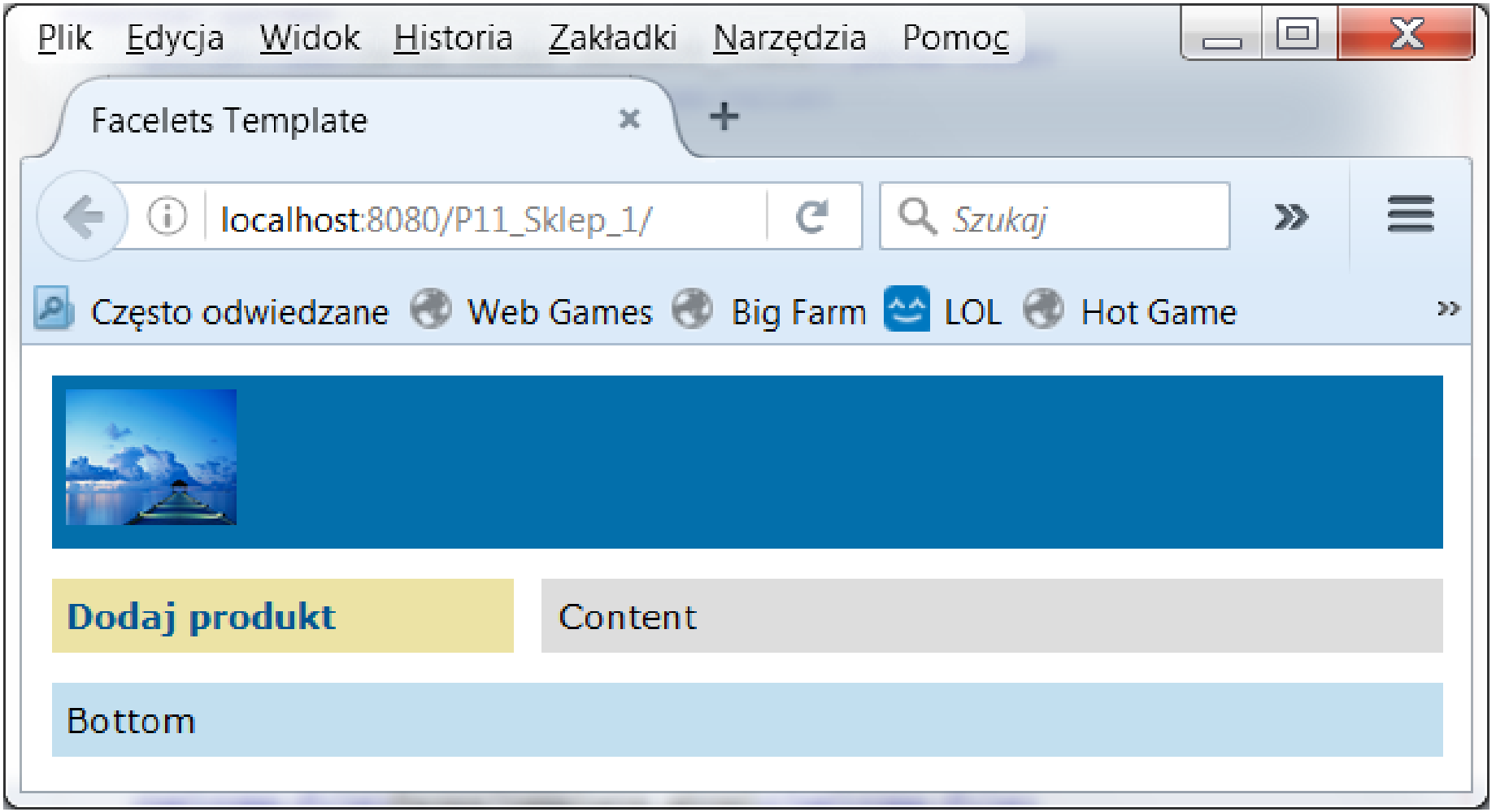
Wprowadzenie do szablonu strony połączenia h:link do strony dodaj_produkt2.xhtml

Wprowadzenie do szablonu strony obsługi komunikatów o błędach

Plik **logo.xhtml** użyty w szablonie stron za pomocą znacznika
<ui:include src="./logo.xhtml" />

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
    <ui:composition>
        <div align="left" style="width: 100%">
            <h:graphicImage value="/resources/obrazy/Dock.jpg"
                                width="59" height="47"
                                title="Molo">
            </h:graphicImage>
        </div>
    </ui:composition>
</html>
```

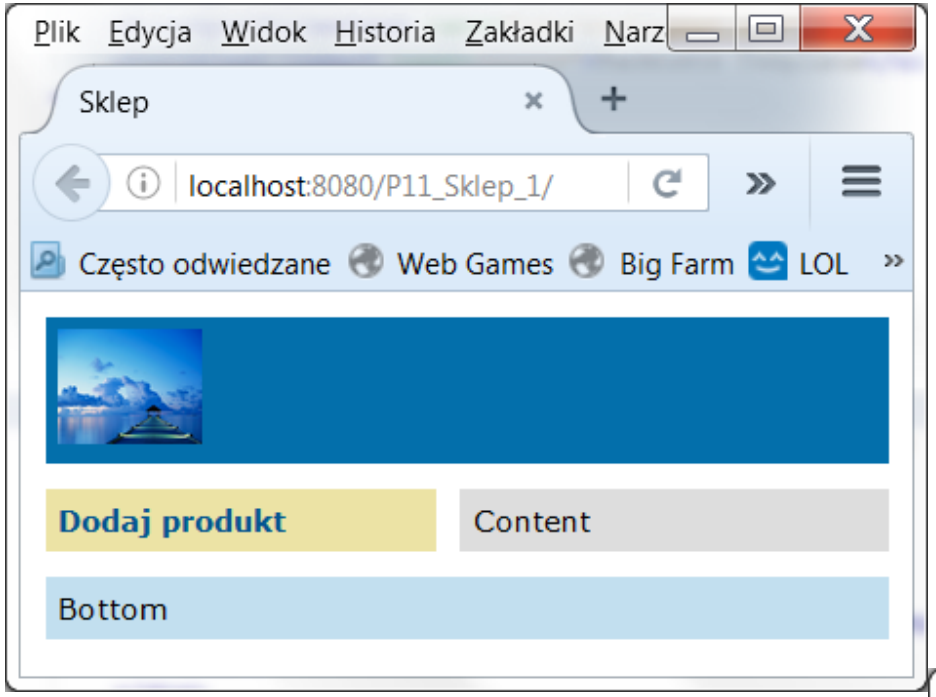
Widok wygenerowanego szablonu strony po zmianach



Strona startowa index2.xhtml

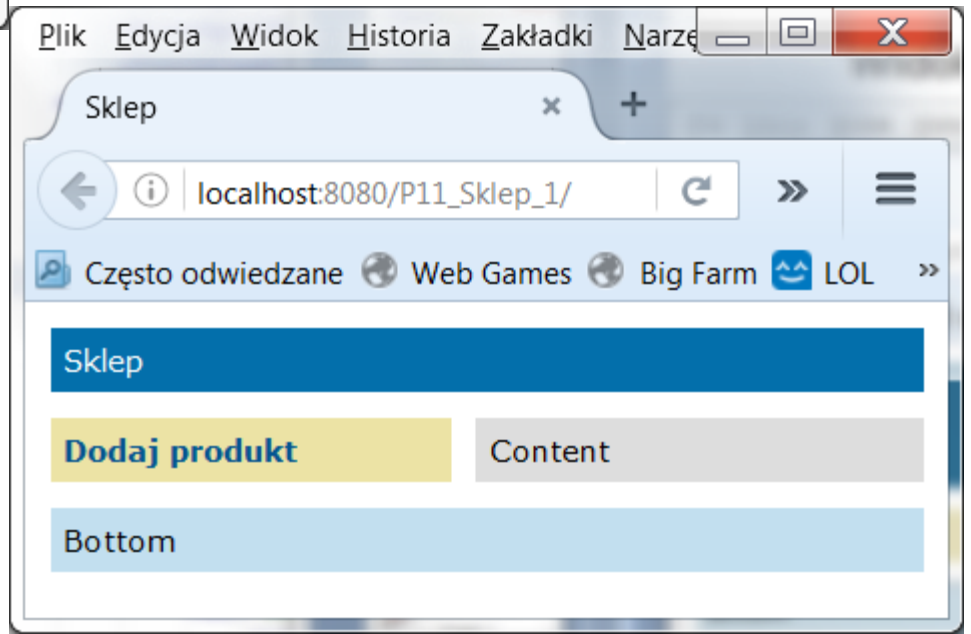
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<body>
      <ui:composition template="./template.xhtml">

          <ui:define name="title">
             Sklep
          </ui:define>

        </ui:composition>
</body>
</html>
```

Widok stron startowej index2.xhtml – różne wersje szablonu




```
<div id="top">
  <h:panelGroup>
   <ui:include src="./logo.xhtml" />
   <ui:insert name="top">
    </ui:insert>
  </h:panelGroup>
 </div>
```

```
<div id="top">
  <h:panelGroup>
   <ui:insert name="top">
      Sklep
   </ui:insert>
  </h:panelGroup>
 </div>
```



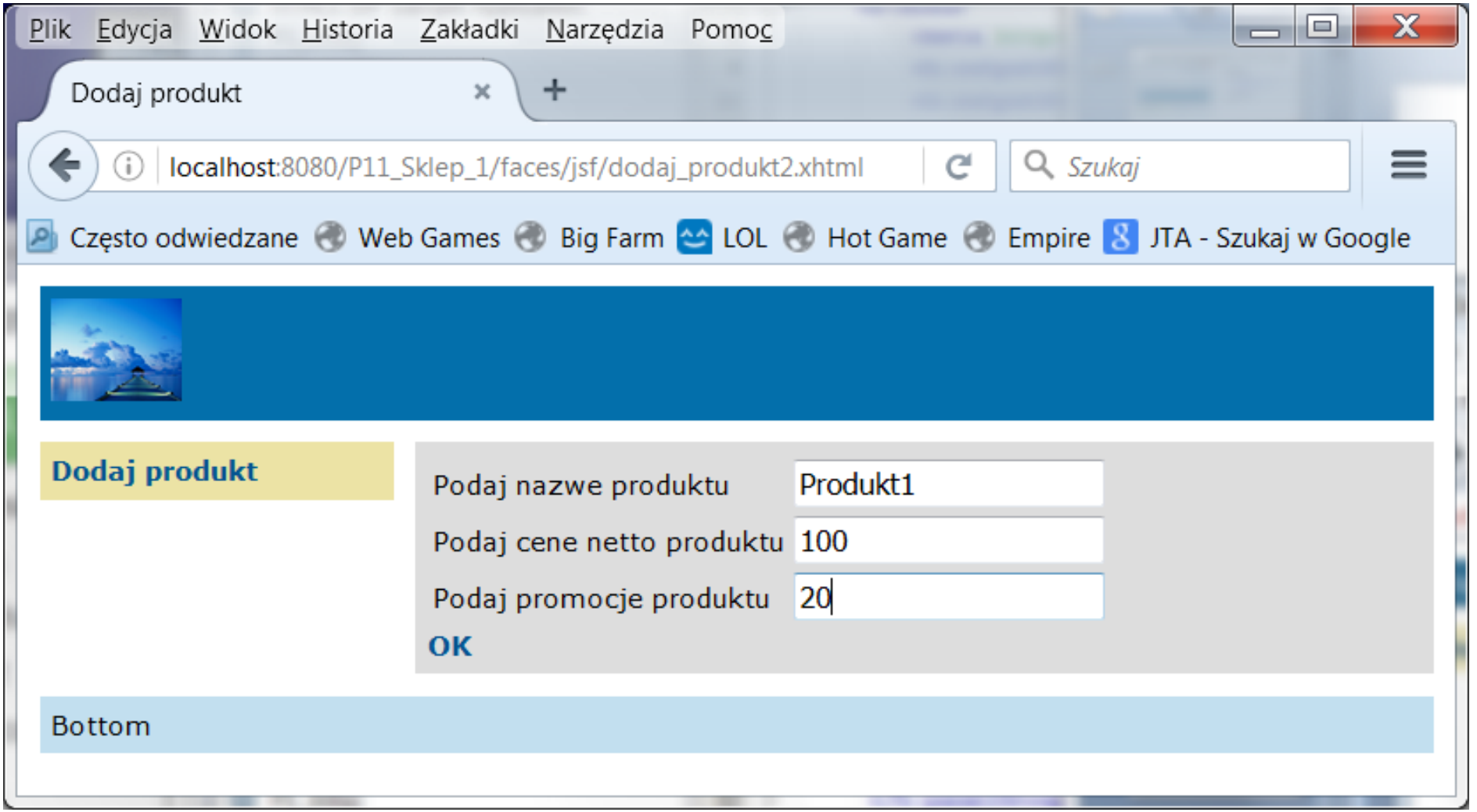
Definicja strony **dodaj_produkt2.xhtml** uruchamianej po kliknięciu na link **Dodaj produkt**

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns: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">
            Dodaj produkt
          </ui:define>

         <ui:define name="content">
          <h:form><h:panelGrid columns="2">
             <h:outputLabel value="Podaj nazwe produktu"  for="nazwa" />
             <h:inputText
                    id="nazwa"
                    title="Podaj nazwe:"
                    value="#{managed_produkt.nazwa}"
                    required="true"
                    requiredMessage="Blad: Podaj nazwe." >
             </h:inputText>
```

Wprowadzenie do strony dodaj_produkt2.xhtml opartej na szablonie, do części oznaczonej jako content treść znacznika h:form z przykładów 8,9 ze strony dodaj_produkt.xhtml

```
<h:outputLabel value="Podaj cene netto produktu" for="cena" />
        <h:inputText
                id="cena"
                title="Podaj cene:"
                value="#{managed_produkt.cena}"
                required="true"
                requiredMessage="Blad: Podaj cene." >
        </h:inputText>
        <h:outputLabel value="Podaj promocje produktu" for="promocja" />
        <h:inputText
                id="promocja"
                title="Podaj promocje:"
                value="#{managed_produkt.promocja}"
                required="true"
                requiredMessage="Blad: Podaj promocje." >
        </h:inputText>
      </h:panelGrid>
      <h:commandLink  action="#{managed_produkt.dodaj_produkt}" value="OK" />

    </h:form>
    </ui:define>
   </ui:composition>
</body>
</html>
```

Widok strony dodaj_produkt2.xhtml po kliknięciu na link Dodaj produkt

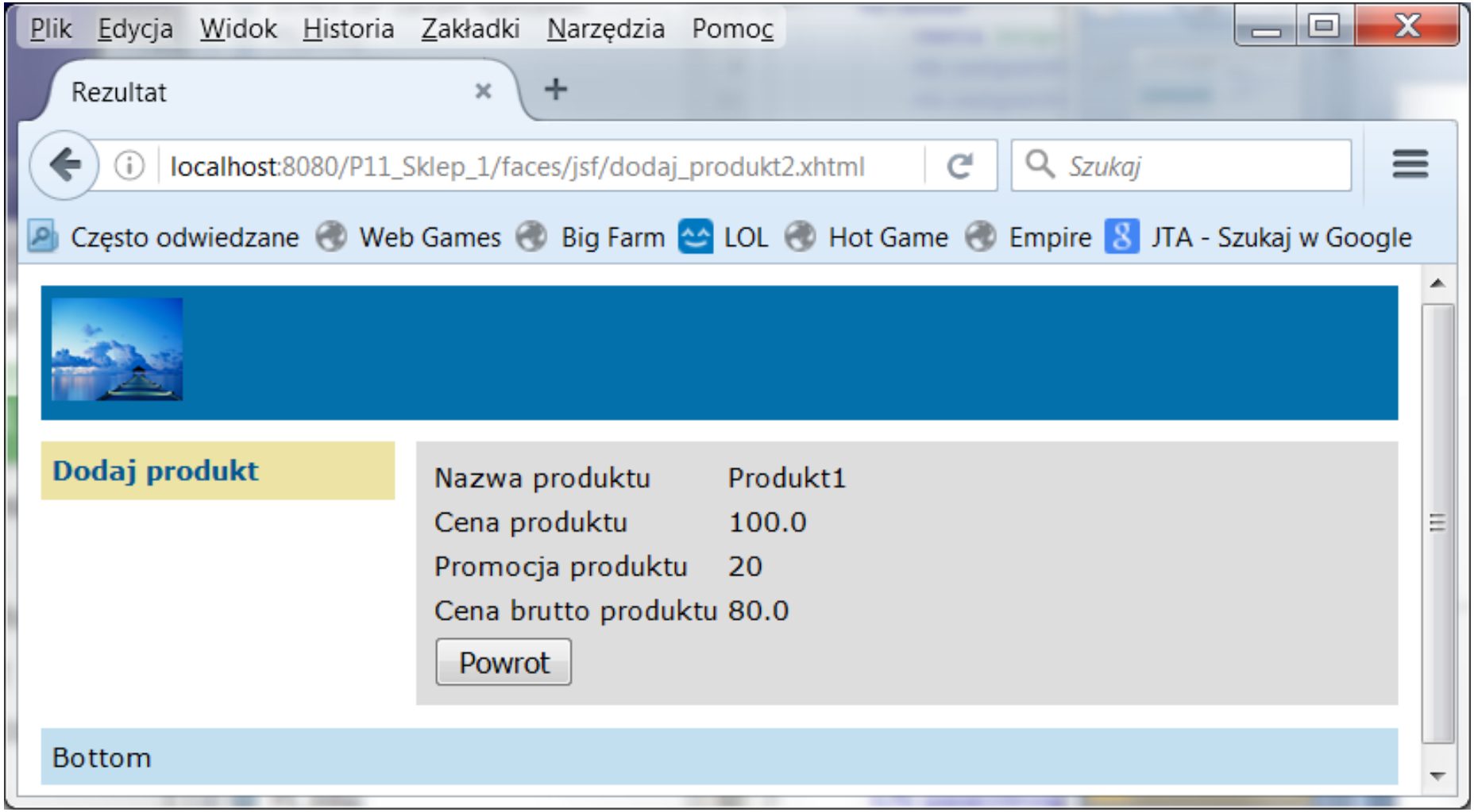


Strona rezultat2.xhtml

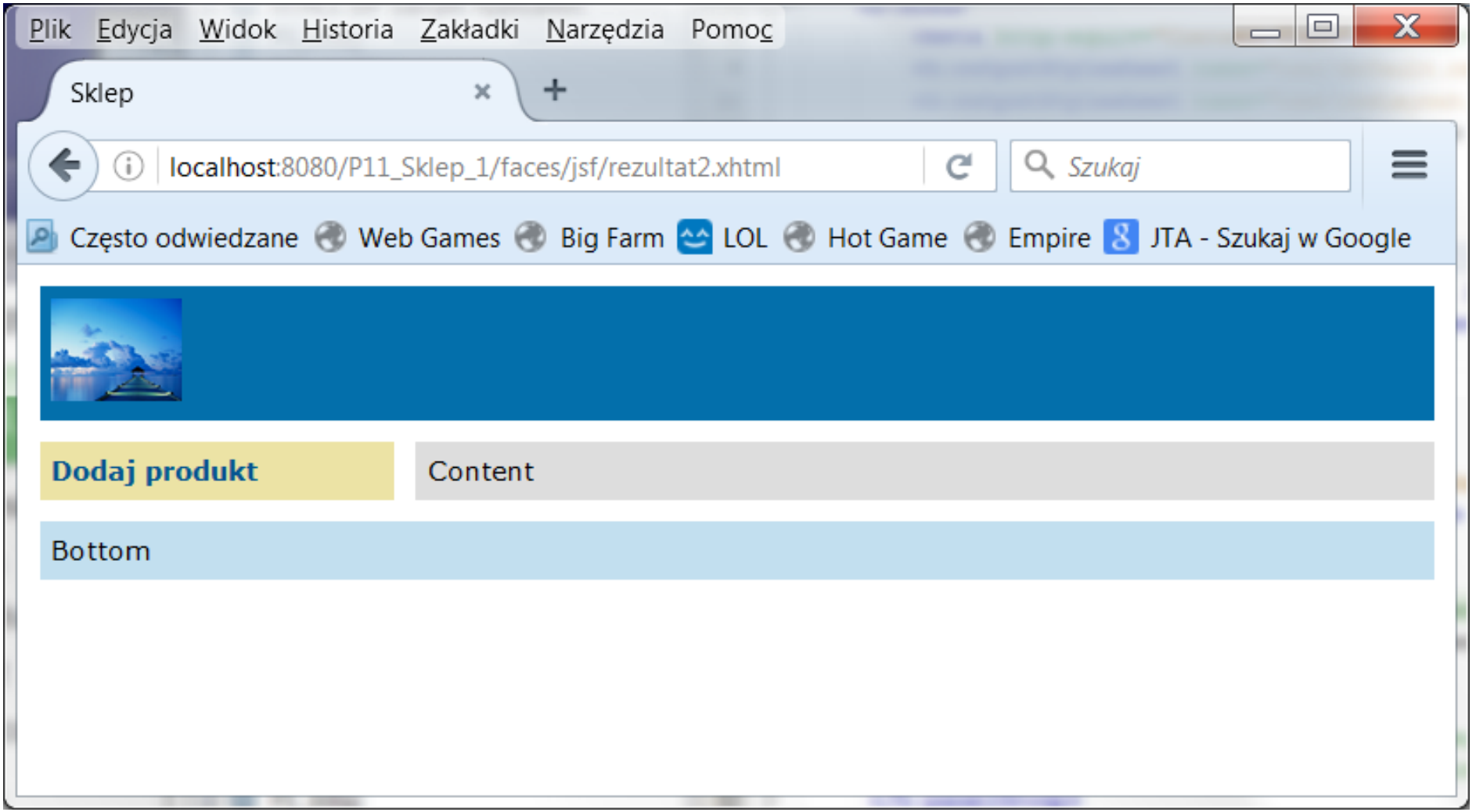
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
     xmlns:h="http://xmlns.jcp.org/jsf/html">
  <body>
 <ui:composition  template="./../template.xhtml">
      <ui:define name="title">
          Rezultat
      </ui:define>
      <ui:define name="content">
        <h:form> <h:panelGrid columns="2">
          <h:outputLabel value="Nazwa produktu"  for="nazwa" />
          <h:outputText id="nazwa" value="#{managed_produkt.nazwa}"/>
          <h:outputLabel value="Cena produktu"  for="cena" />
          <h:outputText id="cena" value="#{managed_produkt.cena}"/>
          <h:outputLabel value="Promocja produktu"  for="promocja" />
          <h:outputText id="promocja" value="#{managed_produkt.promocja}"/>
          <h:outputLabel value="Cena brutto produktu"  for="brutto" />
          <h:outputText id="brutto" value="#{managed_produkt.cena_brutto}" />
          <h: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

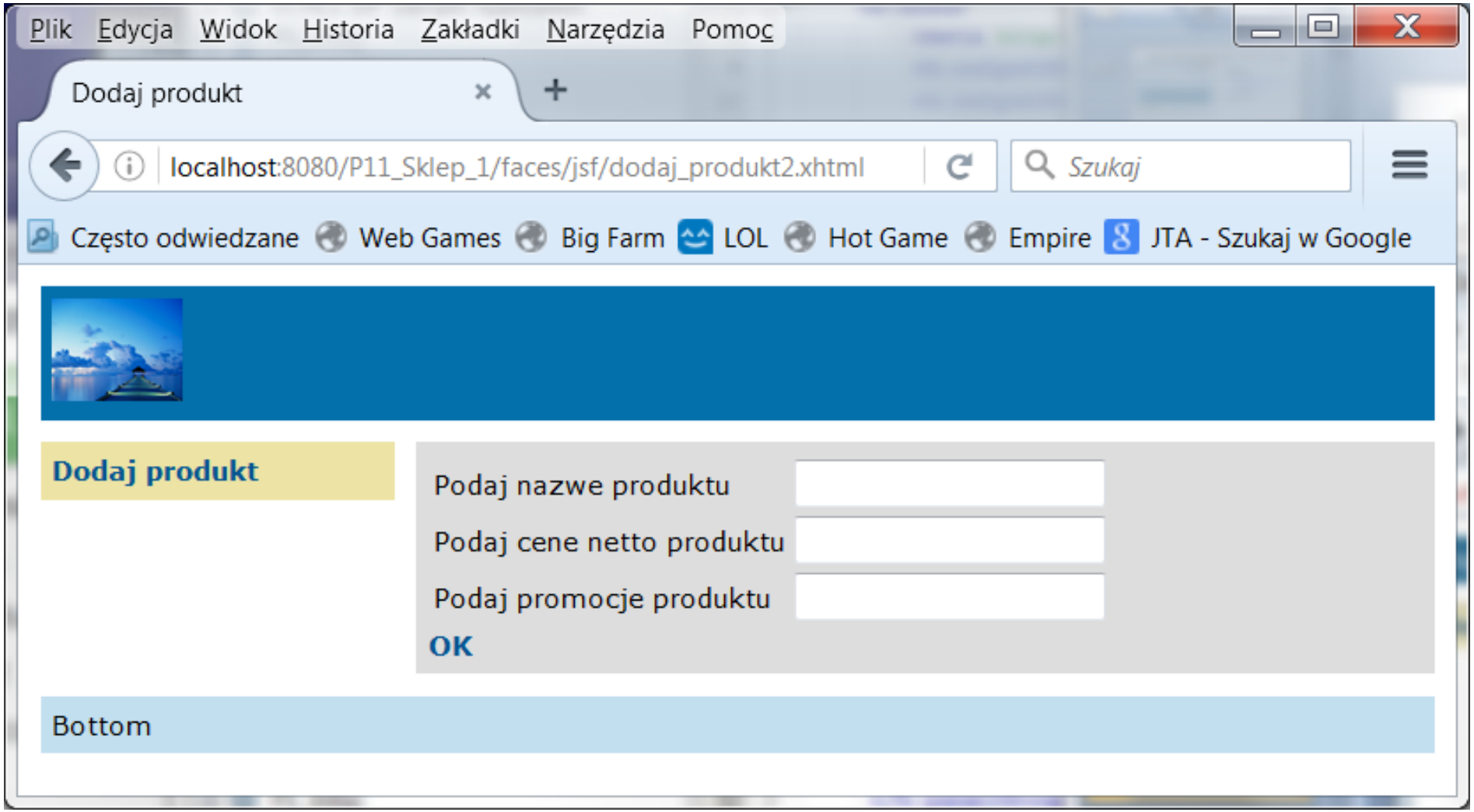
Widok strony rezultat2.xhtml po kliknięciu na przycisk OK



Widok strony index2.xhtml po kliknięciu na przycisk Powrot



Widok strony dodaj_produkt2.xhtml po kliknięciu na link Dodaj produkt



Widok strony dodaj_produkt2.xhtml po kliknięciu na przycisk OK., gdy formularz nie został wypełniony