Zastosowanie technologii Ajax w ramach technologii JavaServer Faces

wg

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

rozdział 13

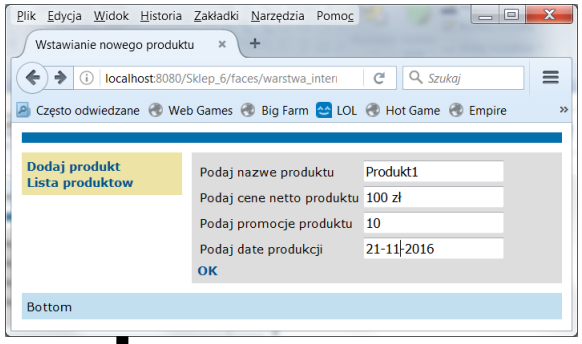
http://www.coreservlets.com

Technologie internetowe 8

1. Wprowadzenie

- **Ajax - Asynchronous JavaScript and XML**
- **Główne zalety:**
  - walidacja danych w czasie rzeczywistym, nie wymagająca ponownego załadowania formularza
  - poprawa funkcjonalności stron internetowych takich jak podpowiedzi nazwy i hasła użytkownika
  - częściowa aktualizacja strony, co poprawia wydajność aplikacji

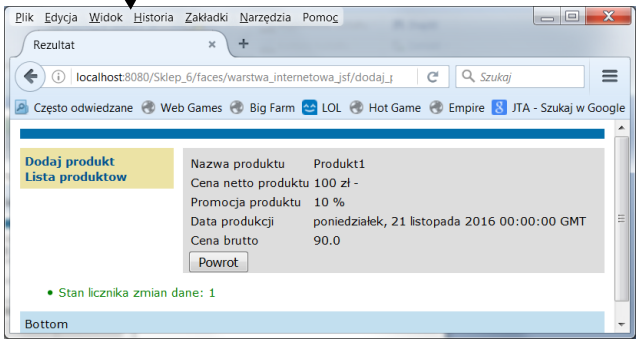
Aktualizacja strony: tradycyjna i z wykorzystaniem Ajax

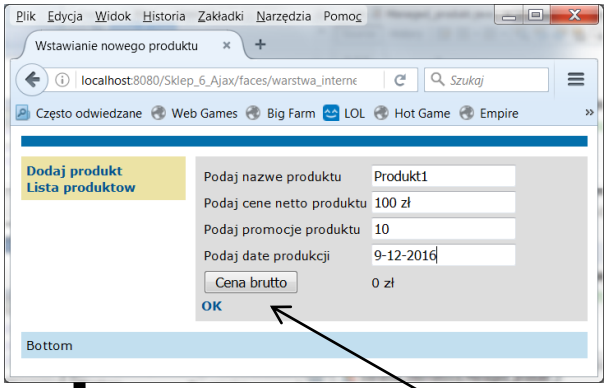


Wysłanie **żądania (HTTP Request)** po naciśnięciu **OK** na stronie **dodaj_produkt2.xhtml**

Serwer

Wysłanie **odpowiedzi XHTML** w postaci nowej strony **rezultat2.xhtml**

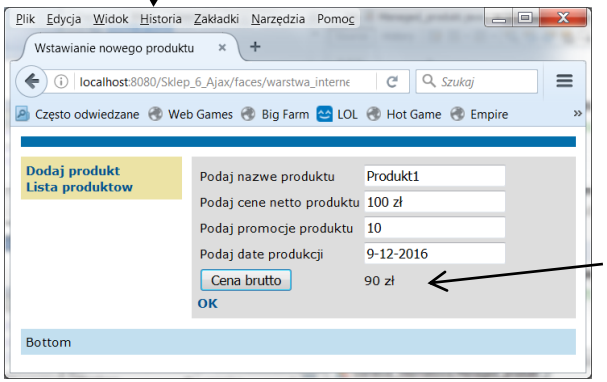




Wysłanie **żądania JavaScript (XML HTTP Request)** po naciśnięciu **Cena brutto** na stronie **dodaj_produkt2.xhtml**

Serwer

Wysłanie **odpowiedzi XML** w postaci aktualizacji pola z prawej strony przycisku **Cena brutto** na stronie **dodaj_produkt2.xhtml**



Przykłady zastosowania technologii Ajax

1. **Wyświetlanie ceny brutto przed przekazaniem wprowadzanych danych do przetwarzania**

1.1. Należy dodać przycisk do strony **dodaj_podukt2.xhtml**, który po naciśnięciu podaje aktualną **cenę brutto** wynikającą z podanej **promocji** i **ceny netto**. **Cena brutto** jest wyświetlana na stronie **dodaj_podukt2.xhtml**, ale bez przeładowania tej strony.

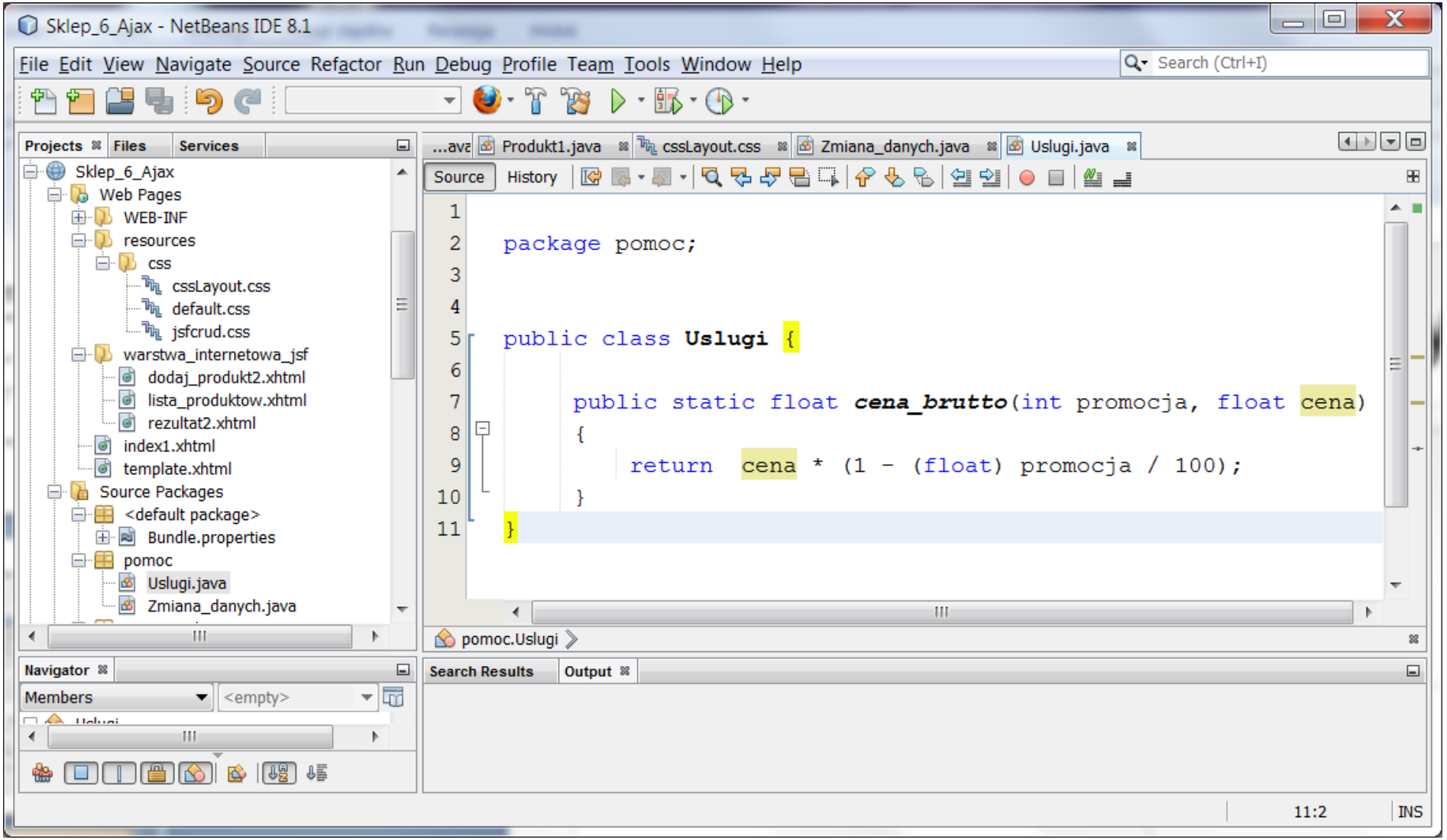
W tym celu należy dodać jako ostatni fragment w znaczniku **panelGrid strony** znaczniki**: h:outputText** oraz **h:CommandButton** z zagnieżdżonym znacznikiem **f:ajax,** który po kliknięciu na przycisk powoduje wysłanie wartości znaczników h:inputText o **id="cena" i id="promocja"** na serwer, gdzie zostanie obliczona cena brutto za pomocą metody **getCena_brutto** w komponencie **managed_produkt i** wysłanie jej do przeglądarki w znaczniku **h:outputText o id="brutto" .**

```
<h:panelGrid columns="2" >
.....
    <h:commandButton value="#{bundle['lista_produktow.cena_brutto']}">
        <f:ajax execute="cena promocja" render="brutto"/>
     </h:commandButton>
    <h:outputText id="brutto" value="#{managed_produkt.cena_brutto_}" >
       <f:convertNumber currencySymbol="z&#322;" type="currency"/>
    </h:outputText>
 </h:panelGrid>
```

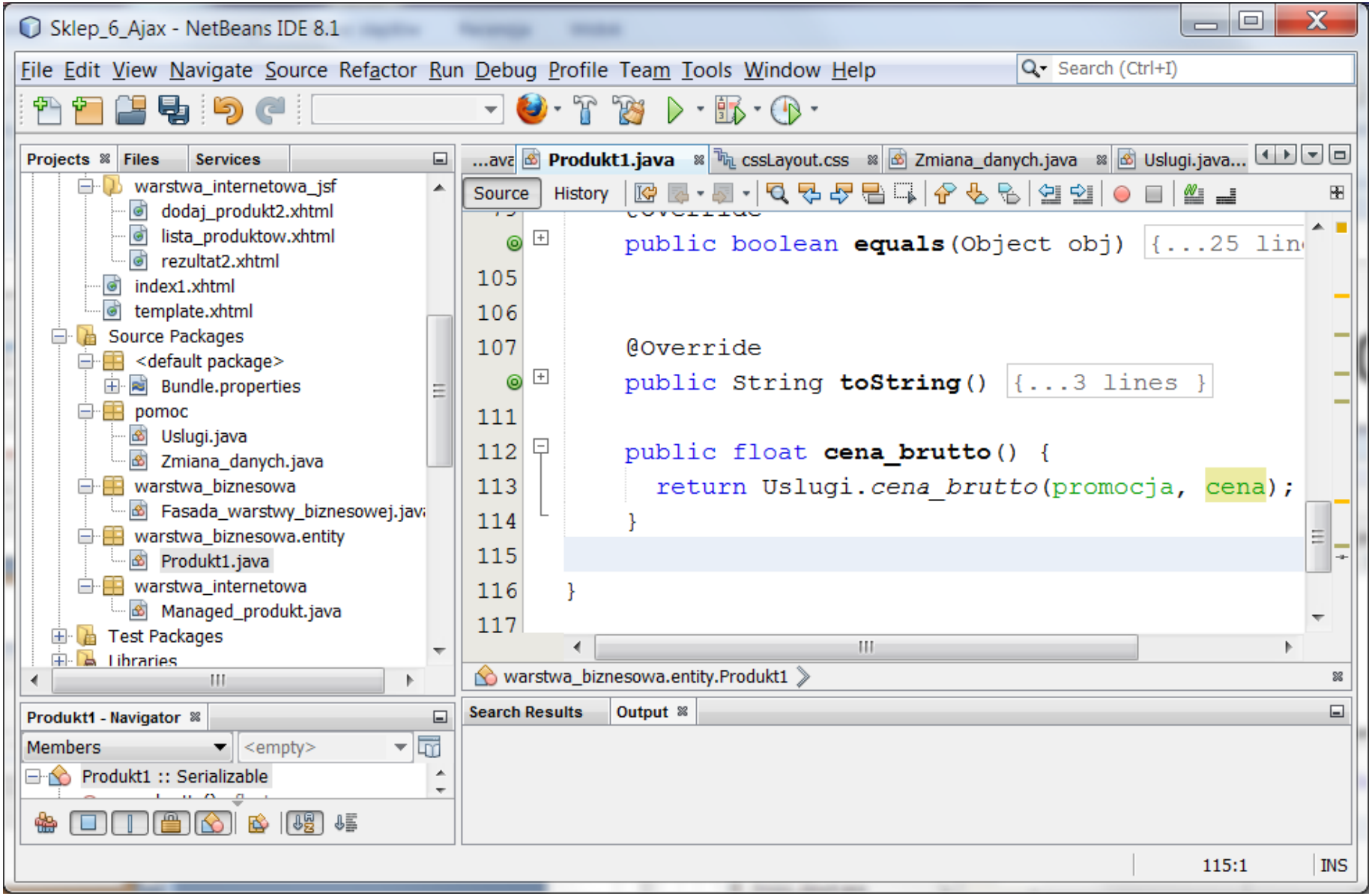
W atrybucie **execute** znacznika **f:ajax** znajduje się łańcuch nazw jako **id** poszczególnych znaczników **h:inputText,** służących do wprowadzania danych na stronie **dodaj_produkt2.xhtml:**

```
<h:inputText   id="cena"
<h:inputText   id="promocja"
```

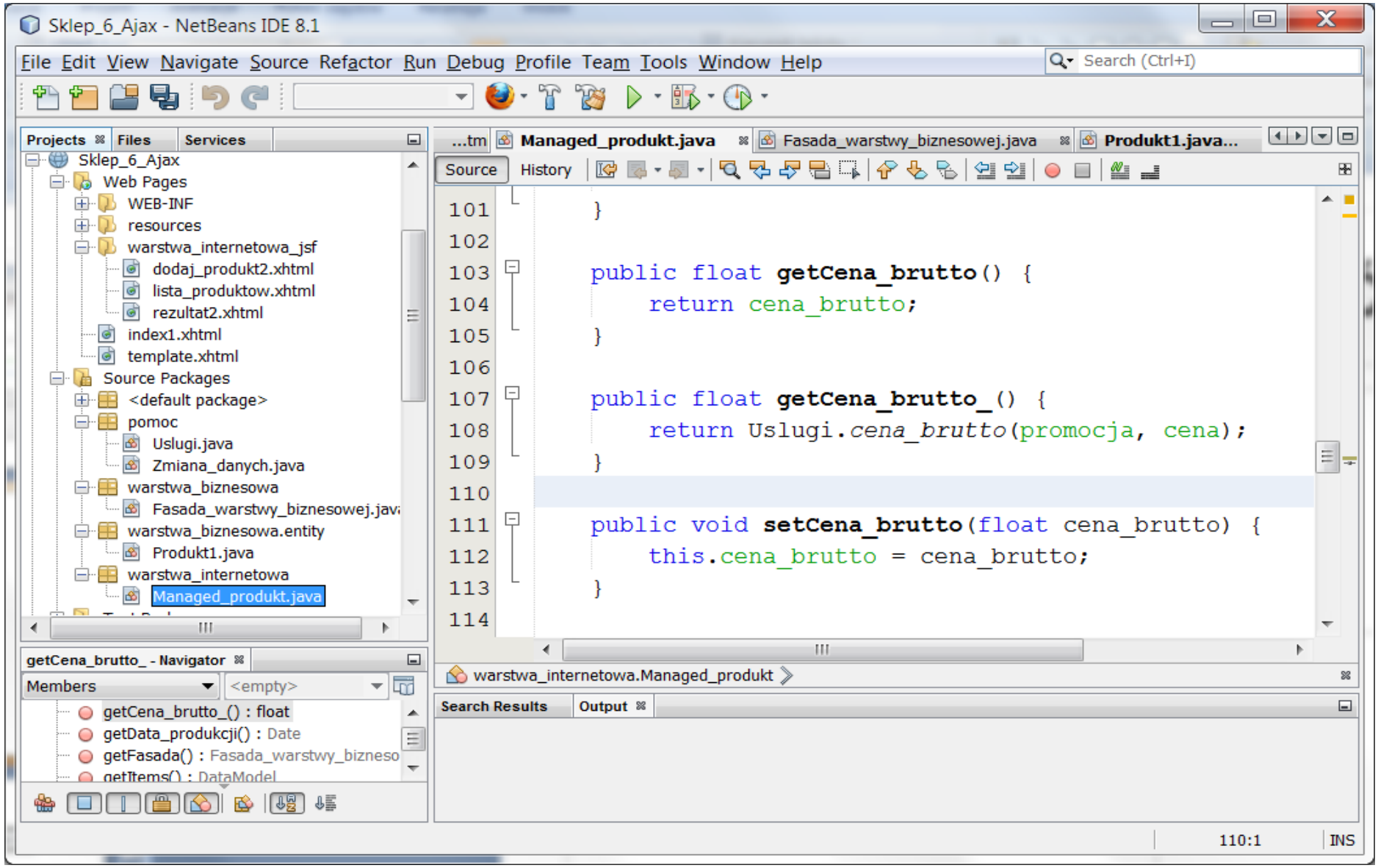
1.2. Zdefiniowanie uniwersalnej metody cena_brutto w klasie Uslugi



1.3. Przedefiniowanie metody cena_brutto w klasie Produkt1 (wprowadzenie wielużywalności kodu)

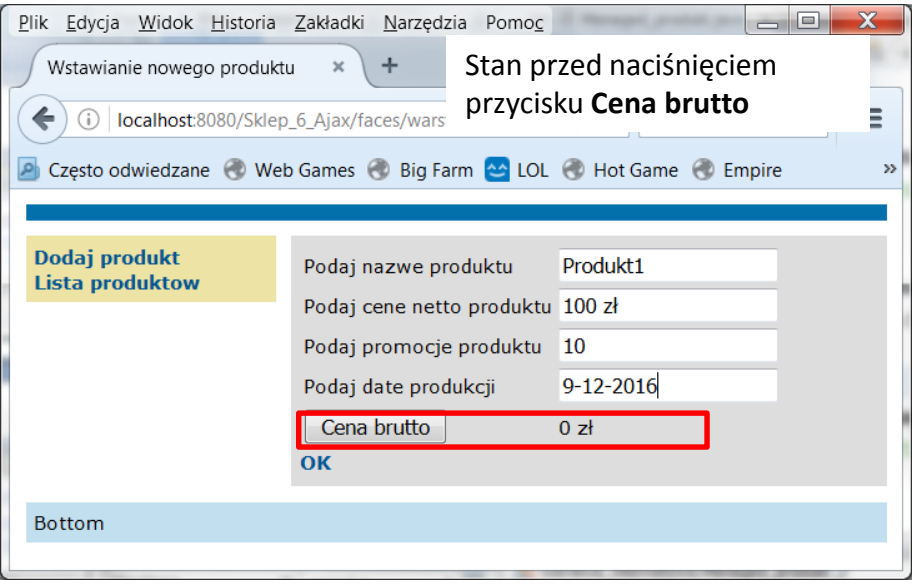


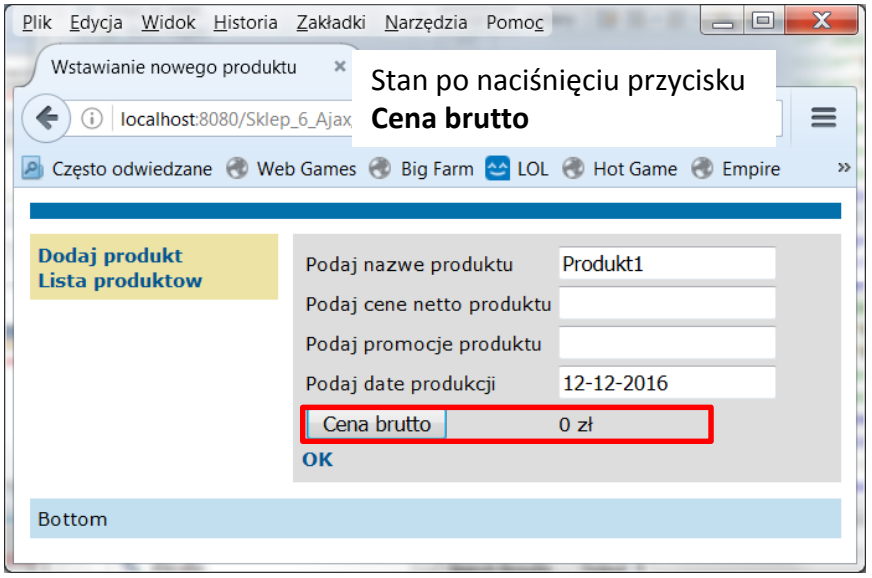
1.4. Dodanie metody **getCena_brutto_** w klasie **Managed_produkt (wprowadzenie wielużywalności kodu),** wywołanej w atrybucie **value:**
<h:outputText id="brutto" **value="#{managed_produkt.cena_brutto_}"** >

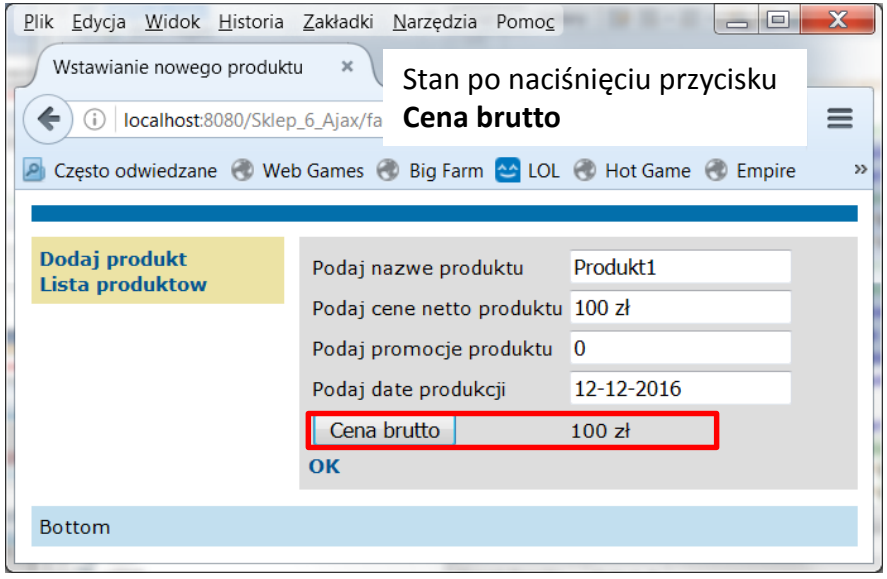


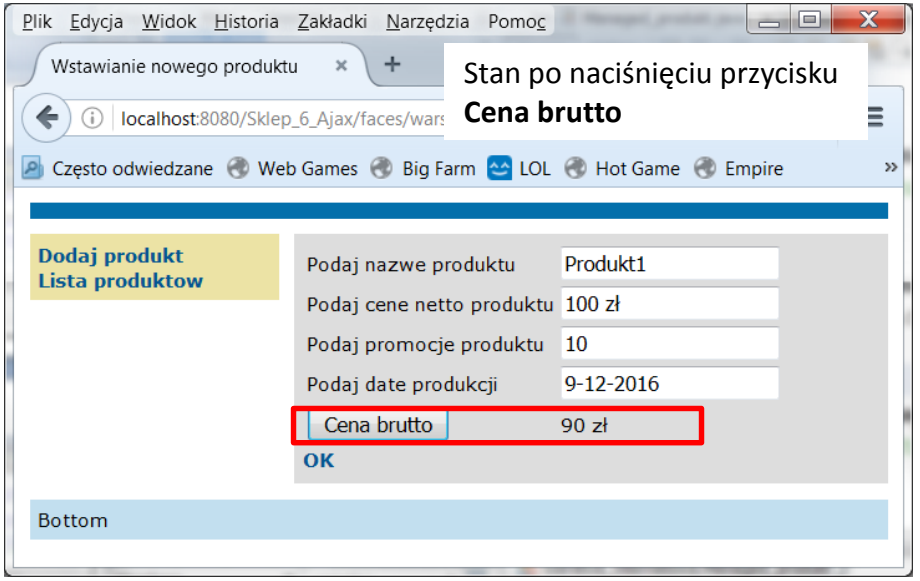
1.5. Wynik działania znacznika <f:ajax execute="cena promocja" render="brutto"/>

Uwaga: w celu uzyskania wyniku należy kliknąć na przycisk **Cena brutto**





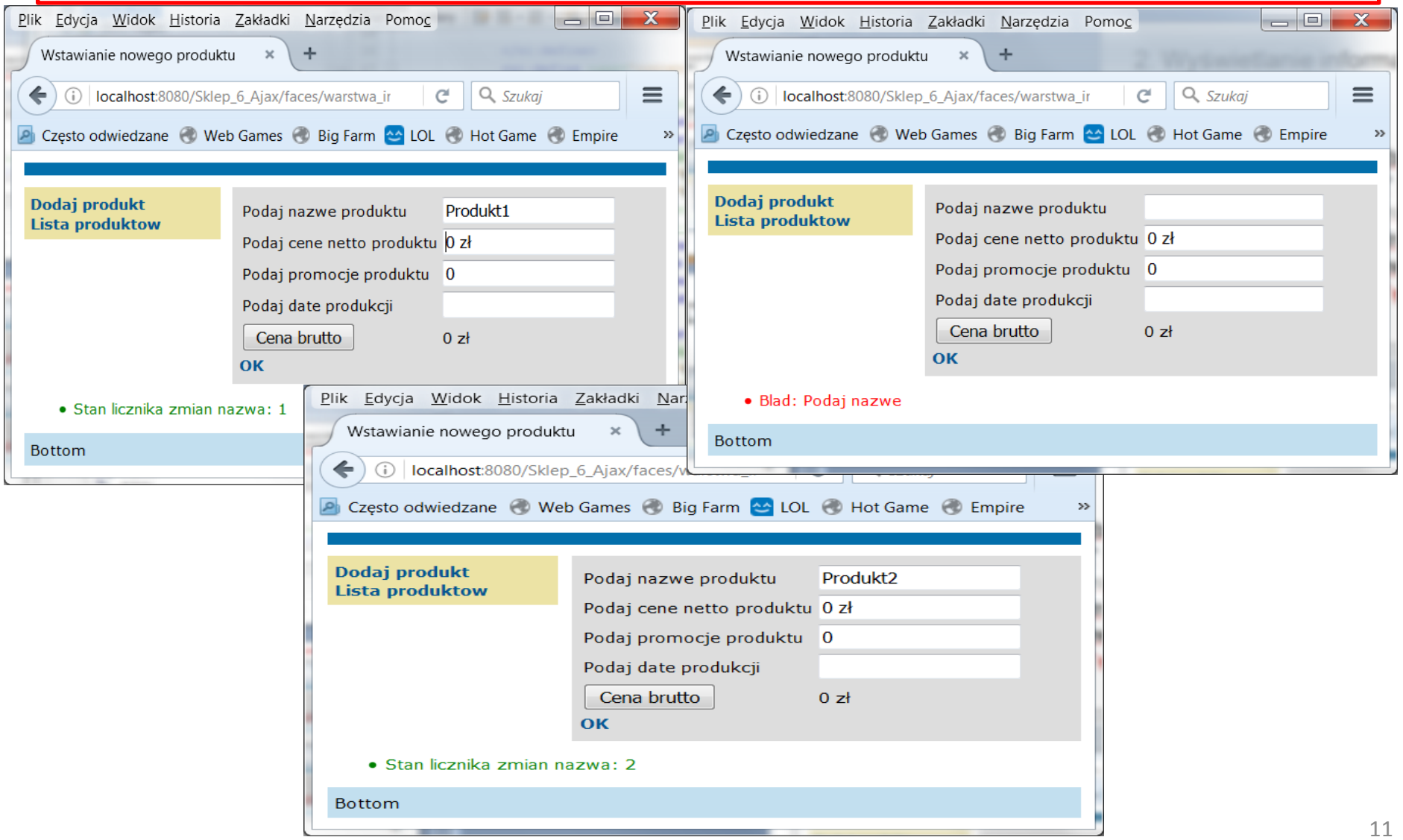




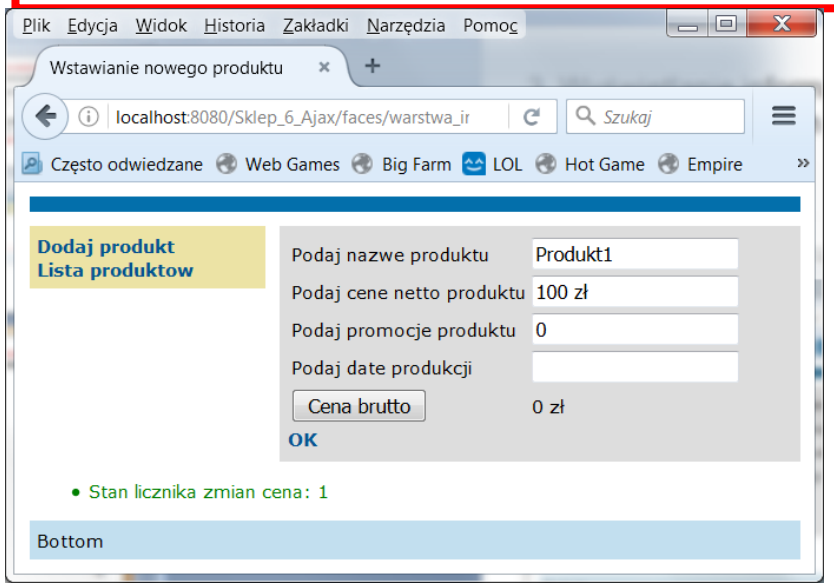
2. Wyświetlanie informacji o liczbie zmian lub braku wprowadzonych danych w polu wejściowym o etykiecie **Podaj nazwe produktu** na stronie **podaj_produkt2.xhtml** bez przeładowania całej strony (za pomocą znacznika **f:ajax**) – należy zmodyfikować kod **h:inputText** podanej poniżej.

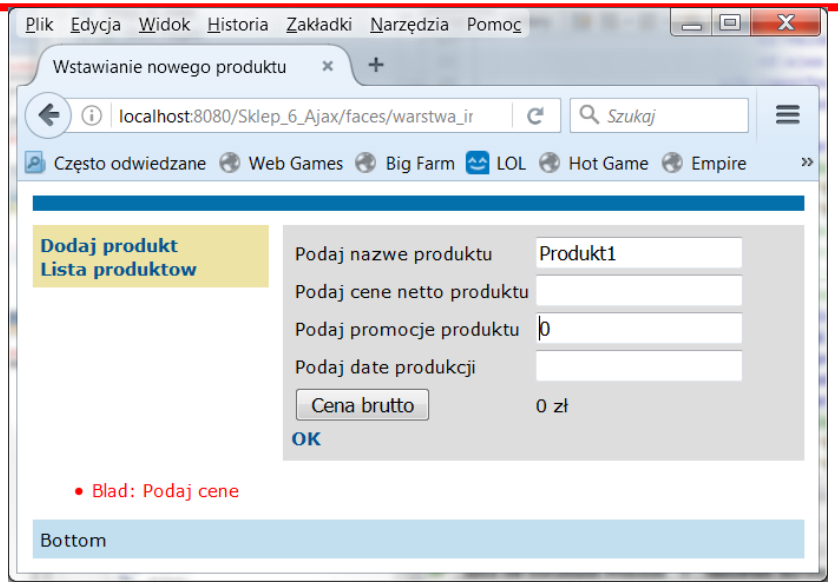
```
<h:outputLabel value="#{bundle['dodaj_produkt2.nazwa']}"  for="nazwa" />
<h:inputText   id="nazwa"  title="#{bundle['dodaj_produkt2.nazwa1']}"
                    value="#{managed_produkt.nazwa}"
                    required="true”
    requiredMessage="#{bundle['dodaj_produkt2.blad_nazwa']}" >
    <f:valueChangeListener binding="#{managed_produkt.zmiana1}"/>
    <f:ajax event="blur" execute="nazwa" render="messagePanel"/>
</h:inputText>
```

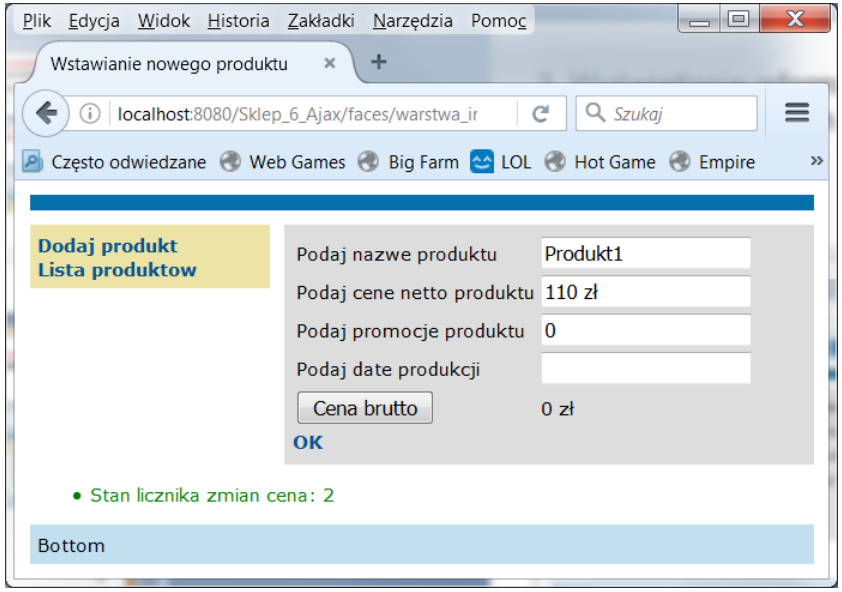
Uwaga: w celu uzyskania komunikatów należy kliknąć np. na tło formularza (dotyczy to p. 2-6). Zdarzenie **event="blur"** występuje w momencie utraty ogniskowania elementu **h:inputText** o **id ="nazwa". Wynik zostanie wyświetlony w polu o id="messagePanel" –** zdefiniowany w pliku **template.xhtml.**

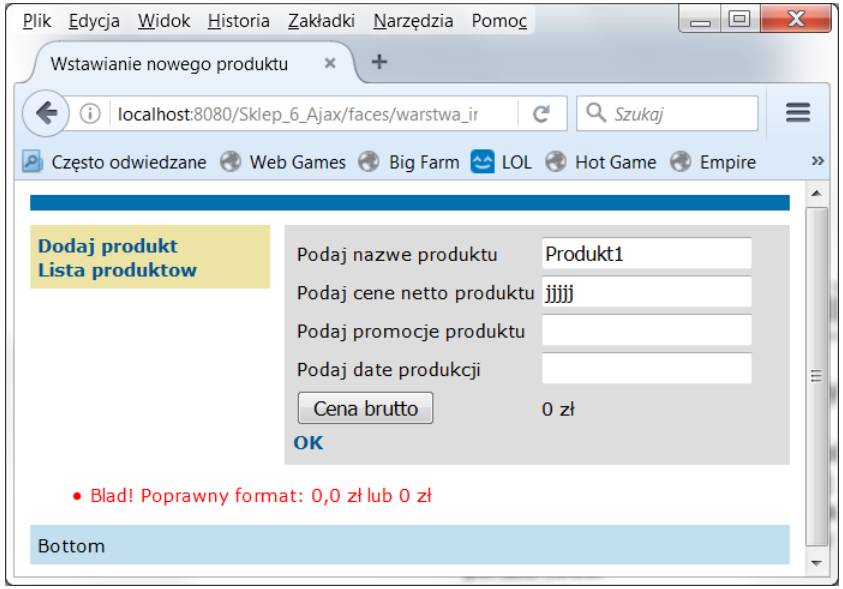


3. Wyświetlanie informacji o liczbie zmian, braku wprowadzonych danych oraz błędy formatu danych w polu wejściowym o etykiecie **Podaj cene netto produktu** na stronie **podaj_produkt2.xhtml** bez przeładowania strony

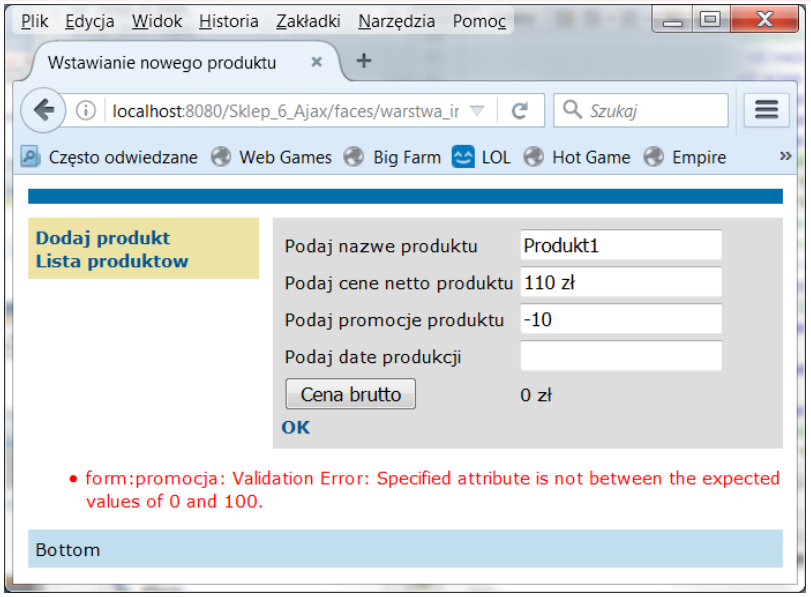


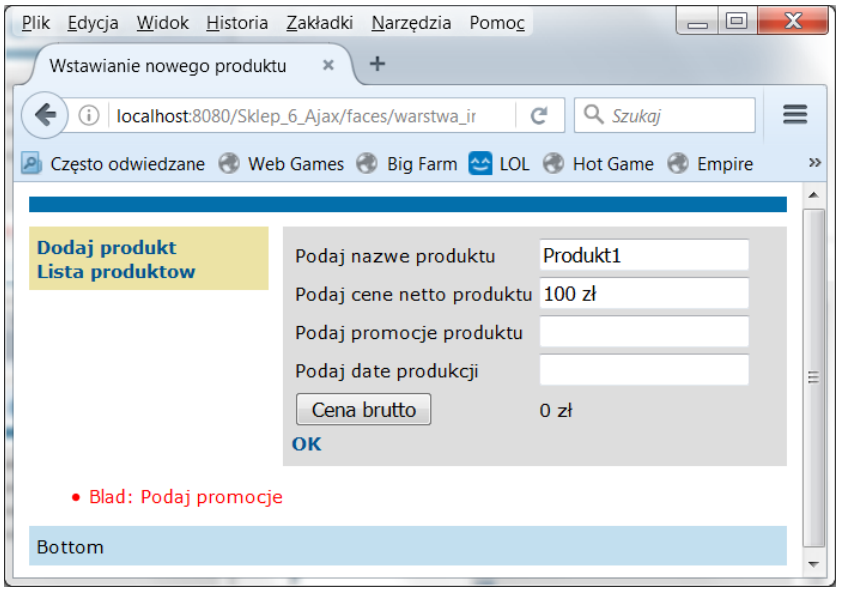


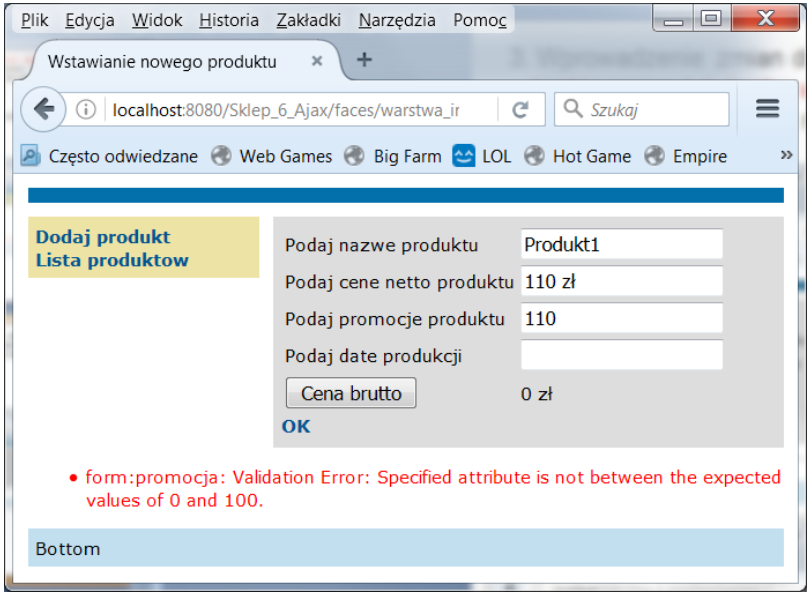


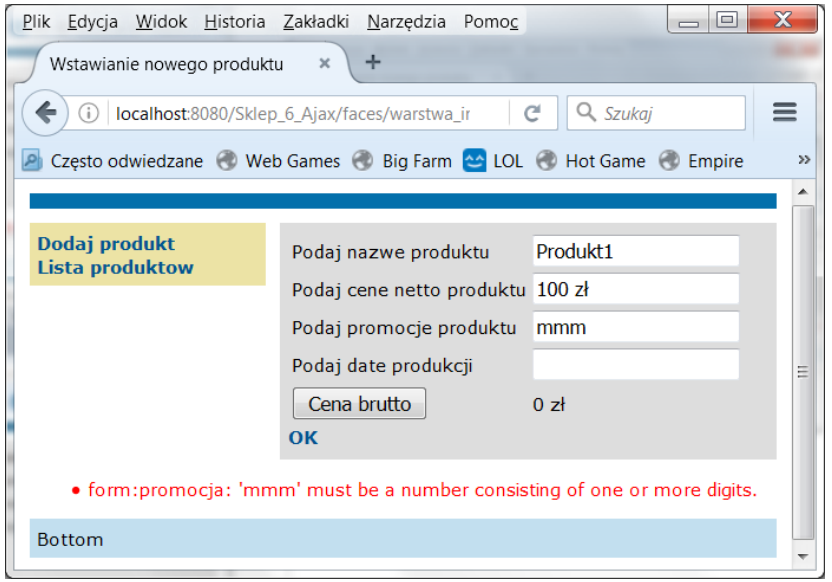


4. Wyświetlanie informacji o błędach przekroczenia wartości minimalne i maksymalnej, braku wprowadzonych danych oraz błędu formatu w polu wejściowym o etykiecie **Podaj promocje produktu** na stronie **podaj_produkt2.xhtml** bez przeładowania strony

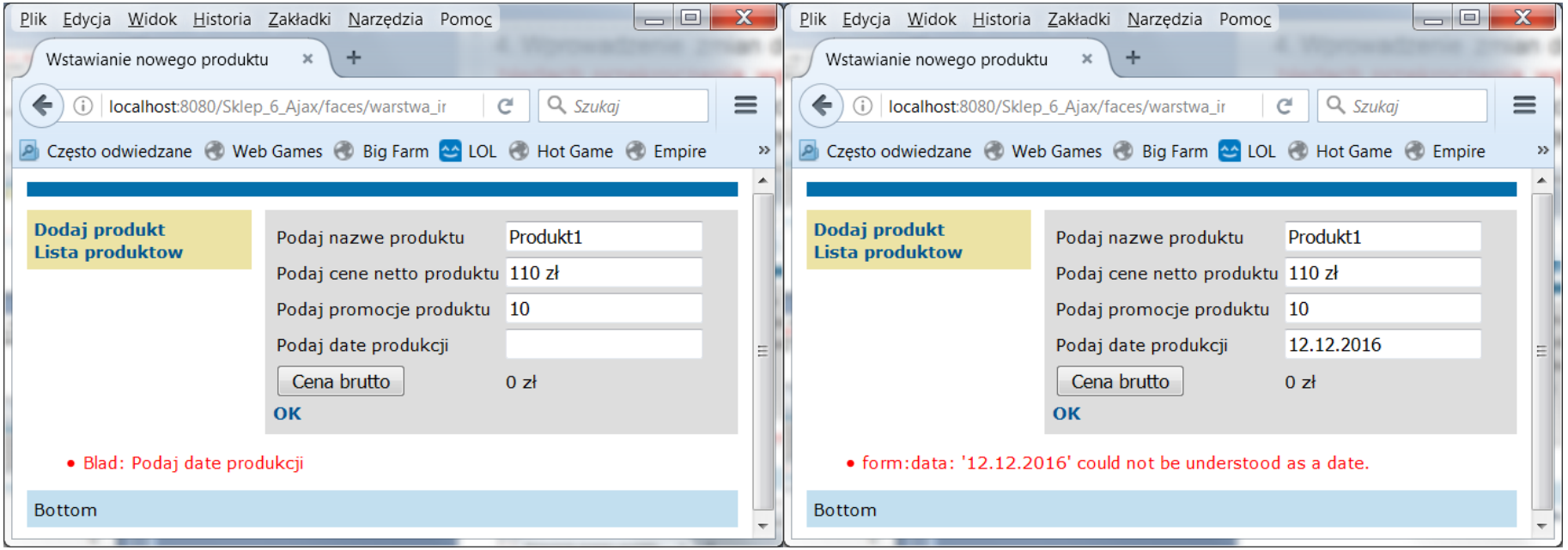








5. Wyświetlanie informacji o błędach formatu daty lub braku wprowadzonych danych daty w polu wejściowym o etykiecie **Podaj date produktu** na stronie **podaj_produkt2.xhtml** bez przeładowania strony



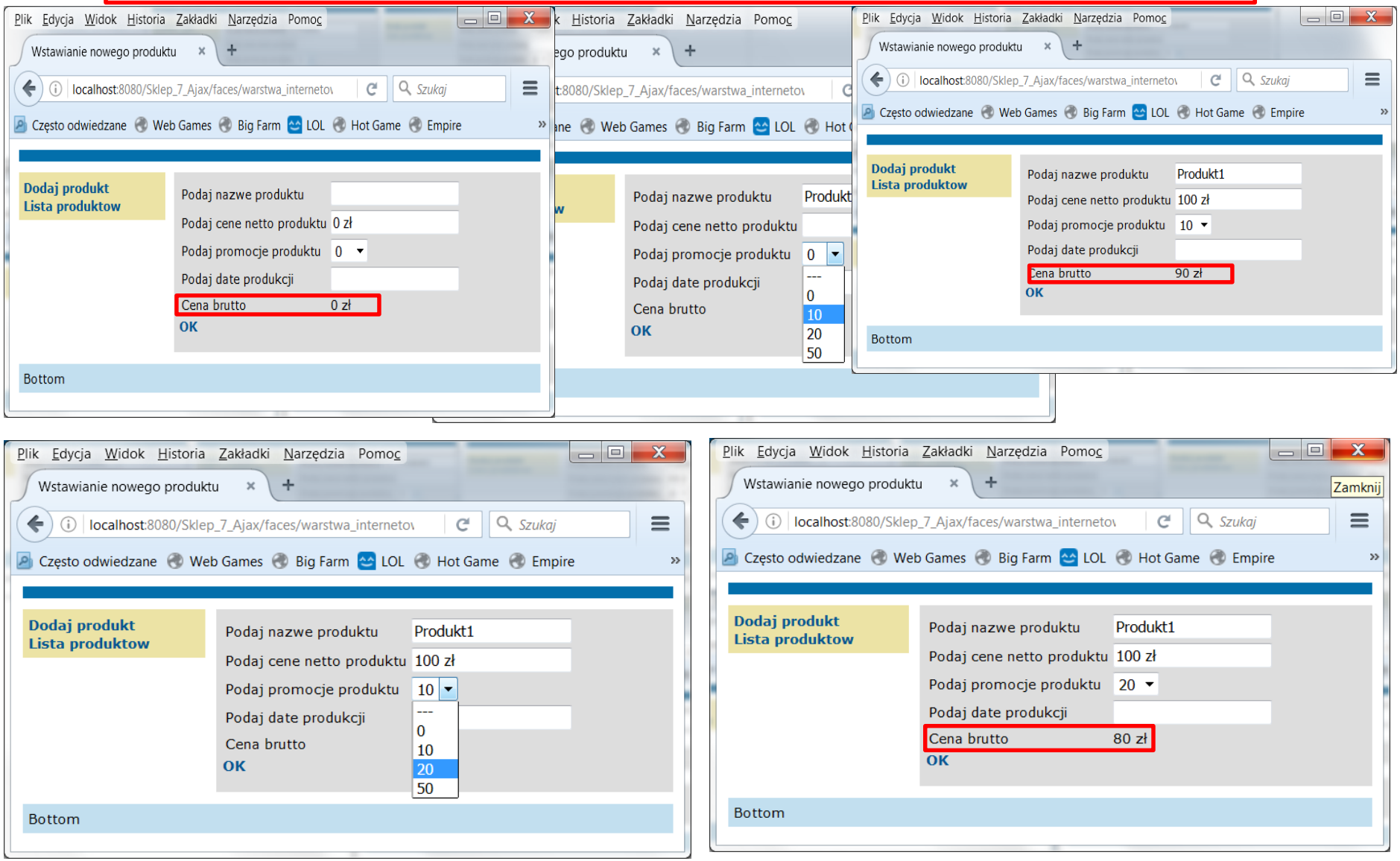
6. Należy wprowadzić zmiany w programie **Sklep_7_Ajax** wykonanym jako kopia **Sklep_7** zrealizowanego w ramach la**b5 –** należy dokonać zmiany podane poniżej w pliku **dodaj_produkt2.xhtml**

W znaczniku wyboru promocji dodano znacznik **f:ajax** w celu wybrania wartości wprowadzonych w znacznikach **h:inputText** o **id="cena"** i **id="promocja"** i wyświetlenia wartości ceny brutto w komponencie h:outputText o **id="brutto"** **bez konieczności ponownego załadowania całej strony.**

```
<h:panelGrid columns="2">
……..
   <h:outputLabel value="#{bundle['dodaj_produkt2.promocja']}" for="promocja" />
   <h:selectOneMenu
                id="promocja"  title="#{bundle['dodaj_produkt2.promocja1']}"
                value="#{managed_produkt.promocja}"
                required="true”  requiredMessage="#{bundle['dodaj_produkt2.blad_promocja']}" >
        <f:selectItems value="#{managed_produkt.itemsAvailableSelectOne}"/>
         <f:ajax execute="cena promocja" render="brutto"/>
   </h:selectOneMenu>
……..
    <h:outputLabel for="brutto" value="#{bundle['lista_produktow.cena_brutto']}"/>
    <h:outputText id="brutto" value="#{managed_produkt.cena_brutto_}" >
        <f:convertNumber currencySymbol="z&#322;" type="currency"/>
    </h:outputText>
</h:panelGrid>
```

7.. Wynik zmiany zawartości strony **dodaj_produkt2.xhtml** bez konieczności przeładowania całej strony – zmiana jedynie pola z etykietą **Cena brutto**

Uwaga: w celu uzyskania wyniku należy wybrać promocję z listy rozwijanej



2. Użycie Ajax z technologią JavaServer Faces
Dwa sposoby

1. **Dodanie kodu JavaScript do aplikacji**
2. **Użycie wbudowanej biblioteki Ajax** (od Java EE 7 wbudowany w bibliotekę JavaScript jako część podstawowej biblioteki JSF)

   2.1. Standardowe komponenty JSF (przyciski, etykiety, pola wejściowe) są powiązane z funkcjonalnością Ajax - zastosowanie znacznika f:ajax

   2.2. Możliwość odwołania do kodu Ajax w kodzie komponentów typu Managed Bean

   2.3 Rozszerzenie komponentów JSF z wykorzystaniem funkcjonalności Ajax

3. Drugi sposób
Użycie Ajax w technologii JSF z wykorzystaniem biblioteki Ajax wbudowanej w bibliotekę JavaServer Faces

- Zastosowanie znacznika f:ajax w komponentach typu Facelets bez potrzeby dodania kodu i konfigurowania komponentów
- Za pomocą użycia metody jsf.ajax.request() biblioteki JavaScript API bezpośrednio w aplikacji Facelets odwołanie do metod Ajax, co umożliwia kontrolę zachowania komponentów

3.1. Zastosowanie znacznika f:ajax – dodanie zachowania Ajax do komponentów wejściowych

```
<h:inputText value="#{bean.message}">
    <f:ajax />
</h:inputText>
```

Jeśli zdarzenie nie jest zdefiniowane, wykonywana jest domyślna akcja dla komponentu.

W przypadku składnika inputText, gdy nie określono atrybutu zdarzenia, zdarzeniem domyślnym jest **valueChange**.

Atrybuty znacznika **f:ajax**

Nazwa	Typ	Opis
disabled	javax.el.ValueExpression przekształca do typu Boolean	Wartość typu **Boolean** identyfikująca status znacznika. Wartość **true** oznacza brak renderowania zachowania Ajax, a **false** oznacza renderowanie zachowania Ajax. Domyślna wartość: **false**
event	javax.el.ValueExpression przekształca do typu String	Wartość String identyfikuje typ zdarzenia Ajax. **Powinna być wspierana przez komponent w przypadku specyfikacji zdarzenia**. W przeciwnym wypadku zdarzenie domyślne jest określone przez komponent : **1) action** dla **javax.faces.component.ActionSource (np commandButton)** **2) valueChange** dla **javax.faces.component.EditableValueHolder (np. inputText)**

Atrybuty znacznika **f:ajax** (cd)

Nazwa	Typ	Opis
execute	javax.el.ValueExpression przekształca do typu Object	Typ Collection, który identyfikuje listę komponentów działających na serwerze w postaci identyfikatorów komponentów typu String (ich atrybut **id**) lub słowa kluczowe (slajd 10). Domyślna wartość to **@this**.
imediate	javax.el.ValueExpression przekształca do typu Boolean	Wartość typu **Boolean** identyfikująca czy wejścia powinny być przetwarzane wcześniej w cyklu przetwarzania JSF. Wartość **true** oznacza przetwarzanie zdarzeń podczas fazy **Apply Request Values**, w przeciwnym wypadku podczas fazy **Invoke Application**
listener	javax.el.MethodExpression	Nazwa słuchacza, który jest wywołany, kiedy javax.faces.event.AjaxBehaviorEvent jest przekazany do słuchacza zdarzeń

Atrybuty znacznika **f:ajax** (cd)

Nazwa	Typ	Opis
onevent	javax.el.ValueExpression przekształca do typu String	Nazwa **funkcji JavaScript**, która obsługuje zdarzenia UI
onerror	javax.el.ValueExpression przekształca do typu String	Nazwa **funkcji JavaScript**, która obsługuje błędy Ajax
render	javax.el.ValueExpression przekształca do typu Object	Typ Collection, który zawiera listę komponentów renderowanych po stronie przeglądarki. Zawiera listę identyfikatorów komponentów (ich atrybuty id) o ograniczonej pamięci i/lub słowo kluczowe (slad 10). Jeśli ValueExpression jest wyspecyfikowane, należy powiązać ten atrybut z właściwością komponentu typu Managed Bean, który zwraca instancję typu Collection z elementami typu String, w przeciwnym wypadku jest wartością **@none.**

Słowa kluczowe dotyczące działania i renderowania atrybutów znacznika f:ajax
render i execue

Słowo kluczowe	Opis
@all	Identyfikatory wszystkich komponentów
@form	Formularz zawierający komponent
@none	Brak identyfikatora komponentu
@this	Element, który wywołuje żądanie

4. Wysłanie żądania Ajax

1) **Użycie atrybutu event**

Możliwe wartości: **click, keyup, mouseover, focus, blur.**

W przeciwnym wypadku zdarzenie domyślne jest określone przez komponent :

a) action dla **javax.faces.component.ActionSource (np commandButton)**

b) valueChange dla **javax.faces.component.EditableValueHolder (np. inputText)**

Przykład: W przykładzie obsługiwane jest kliknięcie myszą. Wartość click jest obecnie domyślna – nie ma konieczności jawnego specyfikowania: **event="click"**

```
<h:commandButton id="submit" value="Submit">
        <f:ajax event="click" />
</h:commandButton>
<h:outputText id="result" value="#{userNumberBean.response}" />
```

Wysłanie żądania Ajax (cd)

2) Użycie atrybutu execute.

Może być równy identyfikatorowi komponentu (atrybut **id** komponentu) lub **@all, @none, @this, @form.**

Uczestniczy we wszystkich fazach przetwarzania żądania oprócz fazy **Render Response**.

Przykład: po kliknięciu myszą na przycisk wykonane jest działanie pola typu **inputText** (i związane z nim walidacje, konwersje itp).

```
<h:inputText  id="userNo" title="Type a number from 0 to 10:"
                    value="#{userNumberBean.userNumber}">
..
</h:inputText>
<h:commandButton id="submit" value="Submit">
        <f:ajax event="click" execute="userNo" />
</h:commandButton>
```

Wysłanie żądania Ajax (cd)

3) **Użycie atrybutu listener** – przygotowanie odpowiedzi po stronie serwera na akcję Ajax tzn zawiera odwołanie do metody, która jest wykonana po stronie serwera jako odpowiedź na akcję Ajax po stronie przeglądarki.
   Metoda słuchacza zdarzeń **javax.faces.event.AjaxBehaviorListener .processAjaxBehavior** jest wywołana raz podczas fazy **Invoke Application** cyklu życia.

Przykład: Zawsze, kiedy ulegnie zmiana ceny biletu lub liczba uczestników wycieczki (**event="change"**), metoda **calculateTotal** przeliczy koszt całkowity i wyświetli w komponencie o **id="total"**

```
<f:ajax event="change" render="total"
        listener="#{reservationBean.calculateTotal}"/>
```

Wysłanie żądania Ajax (cd)

4) **Użycie atrybutu immediate** – określa, czy dane przesłane do serwera powinny być przetwarzane wcześniej w cyklu przetwarzania, czy też później.

   Jeśli atrybut ma wartość **true**, zdarzenia generowane prze stronę, są przetwarzane i przesłane z powrotem podczas fazy **Apply Request Values** , w przeciwnym wypadku podczas fazy **Invoke Application.**

   Domyślną wartością jest **false.**

5. Monitorowanie zdarzeń po stronie klienta (przeglądarka)

Właściwości **onevent** typu Data Object do monitorowania żądań Ajax

Właściwość	Opis
responseXML	Odpowiedź dla Ajax w formacie XML
responseText	Odpowiedź dla Ajax w formacie tekstowym
responseCode	Odpowiedź dla Ajax w formacie numerycznym
source	Źródło bieżącego zdarzenia Ajax: DOM (Document Object Model) element
status	Status bieżącego wywołania Ajax: begin, complete lub success
type	Typ wywołania Ajax: event

Przykład: monitorevent jest funkcją **JavaScript**, która monitoruje żądania **Ajax** i ich rozwój, wysłane przez zdarzenie **click. JSF wywołuje tę metodę i wstawia daną typu Data Object do wywołanej funkcji** na każdym etapie żądania Ajax: **begin, complete i success.** Właściwości tych obiektów podano w tabeli.

```
<f:ajax event="click" render="statusmessage"
        onevent="monitormyajaxevent"/>
```

6. Obsługa błędów

Wartością atrybutu **onerror** jest nazwa funkcji **JavaScript.** W przypadku wystąpienia błędu **Ajax,** JSF wywołuje funkcję **JavaScript** i przekazuje data object, który zawiera wszystkie właściwości jak dla atrybutu **onevent** i dodatkowe właściwości:

- description
- errorName
- errorMessage.

Wartości błędów typu Data Object dla właściwości status

Wartość	Opis
emptyResponse	Brak odpowiedzi Ajax z serwera
httpError	Jedna z poprawnych błędów HTTP: request.status==null or request.status==undefined or request.status<200 or request.status>=300.
malformedXML	Odpowiedź Ajax nie jest poprawna
serverError	Odpowiedź Ajax zawiera błędy

<f:ajax event="click" render="errormessage" onerror="handlemyajaxerror"/>

7. Otrzymanie odpowiedzi Ajax – częściowe renderowanie strony

```
<h:commandButton id="submit" value="Submit">
        <f:ajax execute="userNo" render="result" />
</h:commandButton>
<h:outputText id="result" value="#{userNumberBean.response}" />
```

Atrybut **render** określa, **który fragment strony należy zaktualizować.**

Wartością atrybutu **render** może być jeden lub wiele identyfikatorów id komponentów lub jedna z wartości:

@this, @all, @none, @form lub **wyrażenie typu EL.**

Dzięki tym wartościom atrybutu render wyznacza się fragmenty strony wysłane do aktualizacji po stronie przeglądarki

8. Cykl życia obsługi żądania Ajax

Żądanie **Ajax** różni się od żądań **JSF**. Obsługiwane jest za pomocą **javax.faces.context.PartialViewContext**. Metoda **processPartial** obiektu **PartialViewContext** wykorzystuje informację do częściowego przetwarzania **drzewa komponentów i ich renderowania.**

1) Atrybut **execute** określa, **jaki fragment drzewa komponentów** powinien być przetwarzany. Jest to realizowane za pomocą metody **visitTree** obiektu typu **UIComponent** class.
2) Podobnie używany jest atrybut **render**, który pozwala wyszukać **właściwy komponent w drzewie komponentów (na podstawie id tego komponentu) oraz jego „dzieci”.** Te wyszukane komponenty są renderowane wraz z zagnieżdżonymi komponentami („dziećmi”) i wysłane jako odpowiedź. Wtedy nastąpi aktualizacja widoku.

9. Grupowanie komponentów związanych z Ajax

```
<f:ajax >
  <h:form>
    <h:inputText id="input1" value="#{user.name}"/>
    <h:commandButton id="Submit"/>
  </h:form>
</f:ajax>
```

W tym przypadku należy renderować **wszystkie elementy UI zagnieżdżone w znaczniku f:ajax** podczas zdarzenia "**click**„, **generowanym w dowolnym zagnieżdżonym komponencie**.

Skutek - brak możliwosci wprowadzenia danych za pomocą znacznika <h:inputText

9. cd Grupowanie komponentów związanych z Ajax

```
<f:ajax  event="click" render="@all">
  <h:form>
    <h:inputText id="input1" value="#{user.name}"/>
    <h:commandButton id="Submit"/>
  </h:form>
</f:ajax>
```

W tym przypadku należy renderować **wszystkie elementy UI zagnieżdżone w znaczniku f:ajax** podczas zdarzenia "**click„, generowanym w dowolnym zagnieżdżonym komponencie**.

Skutek-możliwość wprowadzenia danych za pomocą znacznika <h:inputText po kliknięciu na dowolny komponent

9.cd Grupowanie komponentów związanych z Ajax

```
<f:ajax  event="click" render="@all">
...
<h:commandButton id="Submit">
      <f:ajax  event="mouseover"/>
</h:commandButton>
...
</f:ajax>
```

Dodatkowo, zdarzenie "mouseover" przesłania zdarzenie "click" w komponencie h:commandButton: **oba typy zdarzeń (click, mouseover) tego komponentu uruchamiają akcję Ajax, czyli renderowanie wszystkich komponentów**

10*. Ładowanie JavaScript jako zasobu

1. Ładowanie plików typu **jsf.js** **(zasób JavaScript zawarty w technologii JavaServer Faces w bibliotece javax.facces)** – automatycznie przekazywany do przeglądarki klienta w znaczniku <h:ajax>
2. Sposoby przesyłania plików typu **js** bezpośrednio do komponentu:

   2.1. **h:outputScript**

   2.2. przez użycie adnotacji **javax.faces.application.ResourceDependency** w klasie Javy typu UIComponent

10.1* Wykorzystanie JavaScript Api w aplikacjach typu Facelets (1)

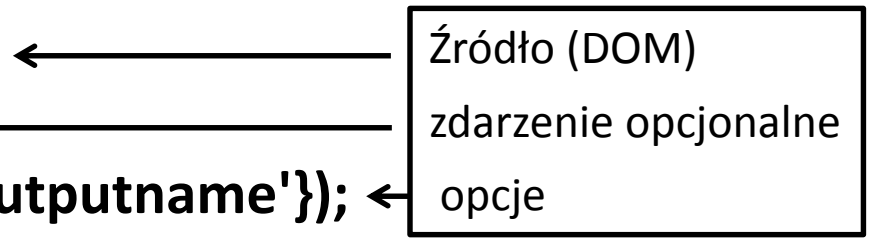
Przykład 1

```
<h:form>
  <h:outputScript name="jsf.js" library="javax.faces" target="head"/>
</h:form>
```

Renderowanie elementu strony HTML typu head wg skryptu **jsf.js**

Przykład 2

```
<h:form>
 <h:outputScript name="jsf.js" library="javax.faces" target="head">
    <h:inputText id="inputname" value="#{userBean.name}"/>
    <h:outputText id="outputname" value="#{userBean.name}"/>
    <h:commandButton id="submit" value="Submit"
      onclick="jsf.ajax.request(this,
       event,
      {execute: 'inputname', render:'outputname'});
      return false;"  />
 </h:outputScript>
</h:form>
```



10.1*. Wykorzystanie JavaScript Api w aplikacjach typu Facelets (2)

Wartość	Opis
execute	Lista identyfikatorów komponentów lub jeden ze słów kluczowych(@all, @form, @none, @this), określających jakie komponenty powinny być przetwarzane podczas fazy **Execute**
render	Lista identyfikatorów elementów strony lub jeden ze słów kluczowych (@all, @form, @none, @this). Te identyfikatory wskazują na komponenty, które są przetwarzane w czasie fazy renderowania strony
onevent	Wartość typu String, oznaczająca nazwę funkcji JavaScript do obsługi zdarzenia
onerror	Wartość typu String, oznaczająca nazwę funkcji do obsługi błędu
params	Wstawienie dodatkowych parametrow do polecenia request

10.2*. Wykorzystanie JavaScript Api w aplikacjach typu Facelets (3)

Wykorzystanie adnotacji
javax.faces.application.ResourceDependency
w celu załadowania biblioteki typu **jsf.js** po stronie serwera i możliwości wykorzystania metody **jsf.ajax.request** w klasie typu Managed Bean.
Ta metoda jest wykorzystana w przypadku tworzenia własnego komponentu lub własnego sposobu renderowania komponentu.

Przykład
Prezentacja ładowania zasobu typu **JavaScript** do klasy typu **Managed Bean**.

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