Zastosowanie technologii Ajax w ramach technologii JavaServer Faces

wg

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

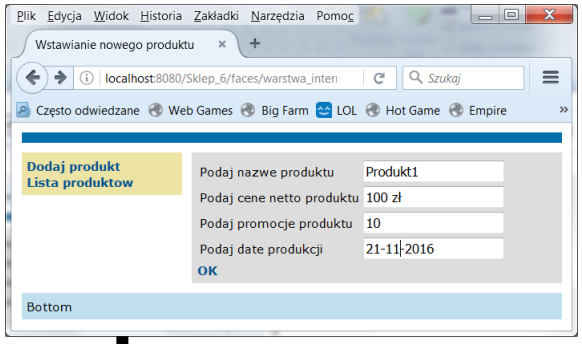
http://www.coreservlets.com

Technologie internetowe 7

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

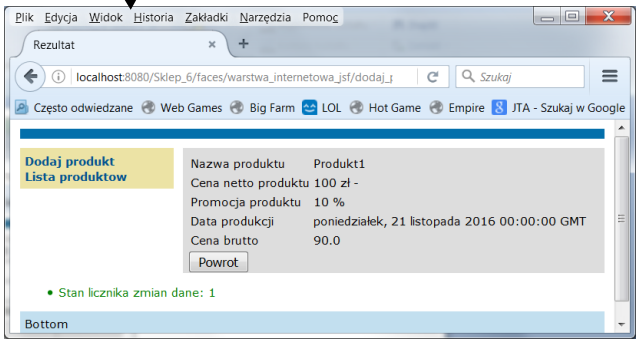
Aktualizacja strony: tradycyjna i z wykorzystaniem Ajax

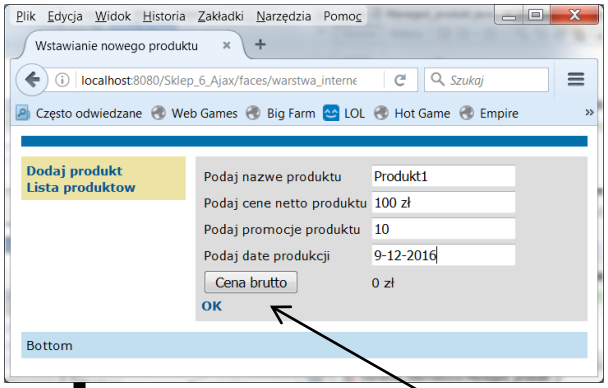


Wysłanie **żądania (HTTP Request)** po naciśnięciu **OK** na stronie **dodaj_produkt2.xhtml**

Serwer

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

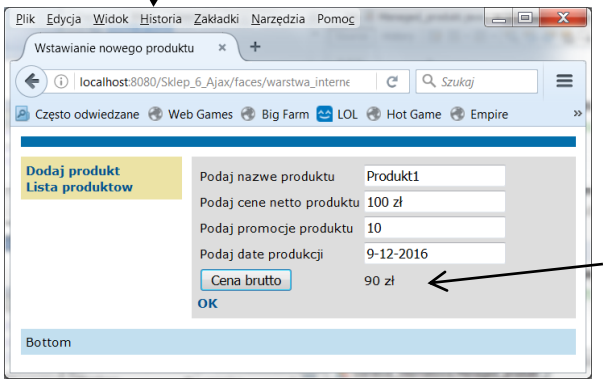




Wysłanie **żądania JavaScript (XML HTTP Request)** po naciśnięciu **Cena brutto** na stronie **dodaj_produkt2.xhtml**

Serwer

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



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

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

   2.3 Rozszerzenie komponentów JSF z wykorzystaniem funkcjonalności Ajax - zastosowanie znacznika f: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>
```

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**) lib 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

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

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

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

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>
```

Źródło (DOM)

zdarzenie opcjonalne

opcje

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

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")
```

Arkusze stylów CSS
(Cascading Style Sheets)

http://www.w3.org/TR/css3-selectors/

http://courses.coreservlets.com/Course-Materials/pdf/jsf/jsf2/JSF2-CSS-Overview.pdf

1. Kaskadowe arkusze stylów CSS

Kaskadowe arkusze stylów pozwalają projektantowi:

- **kontrolować** wygląd dokumentu
- **oddzielić** tę kontrolę od języka HTML, czyli od struktury dokumentu
- **łączyć** w sposób uporządkowany sekwencję informacji na temat stylu z wielu źródeł – stąd kaskadowe arkusze stylów- wg kolejności priorytetów (najwyższy priorytet ma sposób 1):
  - włączane arkusze stylów (w elementach HTML jako atrybut)
  - wewnętrzne arkusze stylów (w bloku znacznika **<head>**)
  - zewnętrzne arkusze stylów
  - domyślny arkusz użytkownika przeglądarki (p.3 - zależny od autora strony)
  - domyślne style przeglądarki (niezależny od autora strony)

2. Ładowanie zewnętrznych arkuszy stylów

2.1. Standardowy HTML

- **Ładowanie** zewnętrznego arkusza stylów

```
<head>
    <link href="css/styles.css"
          rel="stylesheet" type="text/css"/>
...
</head>
```

- **Lokalizacja** zewnętrznego arkusza stylów

– Znajduje się w normalnym katalogu wskazanym za pomocą względnej ścieżki URL. Wartość **css** jest podkatalogiem bieżącego katalogu

2. Ładowanie zewnętrznych arkuszy stylów:
2.2. Specyfika JavaServer Faces

- **Ładowanie** zewnętrznego arkusza stylów

```
<h:head>
   <h:outputStylesheet name="styles.css" library="css"/>
…
</h:head>
```

- **Lokalizacja** zewnętrznego arkusza stylów

– Plik CSS znajduje się w podkatalogu wskazanym przez "library", odniesionym do katalogu „resources" :

…/resources/css/styles.css

2. Ładowanie zewnętrznych arkuszy stylów:
2.3. Specyfika JavaServer Faces

- **Ładowanie** zewnętrznego arkusza stylów

```
<h:head>
   <h:outputStylesheet name="css/styles.css" />
…
</h:head>
```

- **Lokalizacja** zewnętrznego arkusza stylów

– Plik CSS znajduje się w podkatalogu wskazanym przez „name”, odniesionym do katalogu „resources” :

…/resources/css/styles.css

3. Zagnieżdżanie stylów (HTML i JSF)

- **wewnętrzne arkusze stylów - umieszczane w znaczniku head**

```
<head>
<style type="text/css">
    p { color: blue; }
    .note { font-weight: bold; background-color: red; }
</style>
…
</head>
```

- **włączane arkusze stylów z wykorzystanie atrybutu style**

```
<h1 style="color: red; background-color: blue">…</h1>
```

4. Zastosowanie arkuszy stylów

- **Style przypisane do elementów**

  h2 { color: blue; font-family: sans-serif }

  Wszystkie elementy <h2> są niebieski i napisane czcionką sans:serif

- **Style zdefiniowane jako .nazwa_stylu zastosowane z atrybutem "class" (HTML) lub "styleClass" (JSF)**

  .left_content {

  background-color: #dddddd;

  padding: 5px;

  margin-left: 170px;

  height:150px;

  }

  – elementy HTML

  ** … **

  – JSF elements

  <h:outputText styleClass="left_content" …/>

4.1. Przykład definicji stylów

```
body {
    background-color: #ffffff;
    font-size: 12px;
    font-family: Verdana, "Verdana CE",
        Arial, "Arial CE", "Lucida Grande CE",
        lucida, "Helvetica CE", sans-serif;
    color: #000000;
    margin: 10px;
}
```

```
#top {
    position: relative;
    background-color: #036fab;
    color: white;
    padding: 5px;
    margin: 0px 0px 10px 0px;
}
```

```
#left {
    float: left;
    background-color: #ece3a5;
    padding: 5px;
    width: 150px;
   }
```

```
.left_content {
    background-color: #dddddd;
    padding: 5px;
    margin-left: 170px;
     height:150px;
}
```

4.2. Przykład wykorzystania stylów

```
<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>
      <ui:insert name="title"> Template</ui:insert>
    </title>
  </h:head>
  <h:body>
    <div id="top">
      <h:panelGroup>
        <ui:insert name="top"></ui:insert>
      </h:panelGroup>
    </div>
```

4.3. Przykład wykorzystania stylów

```
<div>
    <div id="left">
        <h:link outcome="/faces/warstwa_internetowa_jsf/dodaj_produkt2"
                value="Dodaj produkt"/>    <br/>
        <h:link outcome="/faces/warstwa_internetowa_jsf/lista_produktow"
                value="Lista produktow"/><br/>
    </div>
     <div id="content" class="left_content">
        <ui:insert name="content">Content</ui:insert>
     </div>
</div>
```

4.4. Przykład wykorzystania stylów

```
<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

5. Podstawowe selektory

https://www.w3.org/TR/css3-selectors/#selectors
http://courses.coreservlets.com/Course-Materials/pdf/jsf/jsf2/JSF2-CSS-Overview.pdf

Selektor	Przykład definicji selektora	Przykład zastosowania selektora
element	body{…} h1 {….}	Użycie znaczników bez jawnego odwołania do stylu
#id	#bottom {…..} #left {…} #content{…}	** ** ** **
.class	.left_content {…..}	** **
element.class	form.left_content{..}	**<form class="left_content">**
element#id	form#bottom{…}	**<form id="bottom">**
*	*{…..} div *{…..}	Dotyczy wszystkich elementów na stronie Dotyczy wszystkich elementów zawartych w znaczniku **div**

5.1. Hierarchia selektorów

Selektor	Przykład definicji	Znaczenie
s1 s2	div.foo span.bar{…}	Styl wszystkich elementów ****, które mogą znajdować się wewnątrz ** **
s1>s2	div.foo > span.bar {…}	Styl wszystkich elementów ****, które są zawarte wewnątrz ** **
s1, s2	ul,ol,div.foo {…}	Oznacza definicję stylu wszystkich elementów **ul, ol, **
s1+ s2	label + input {…}	Oznacza styl wszystkich elementów **input** znajdujących się za elementami **label**
s1~s2	label ~ input {…}	Oznacza styl wszystkich elementów **input**, które mają element **label** znajdujący przed elementami **input** na tym samym poziomie zagnieżdżenia

5.2. Atrybuty selektorów

Selektor	Przykład definicji	Znaczenie
s[att]	div.blah a[name] {…}	Styl wszystkich elementów **<a name="…">,** które są wewnątrz ** **
s[att=val]	**a[href=#sect2] {…}**	Oznacza styl wszystkich elementów **<a href="#sect2">**
s[att^=val]	a[href^=#] {…}	Oznacza styl wszystkich wewnętrznych linków
s[att$=val]	a[href$=jquery.com] {…}	Oznacza styl wszystkich linków do strony **blah.jquery.com** (bez podstron)
s[att*=val]	a[href*=jquery.com] {…}	Oznacza styl wszystkich linków do każdej strony, na stronie dostępnej z **blah.jquery.com**
s[att!=val]	a[href!=#sect2] {…}	Oznacza styl wszystkich linków oprócz linków **<a href="#sect2">**
s:not([….])	a:not([href^=http]) {…}	Oznacza styl wszystkich linków, które nie zaczynają się od **http**…

5.3. Selektory pozycyjne

Selektor	Przykład definicji	Znaczenie
s:first s:last	ul.foo li:first {…}	Oznacza styl pierwszego elementu, który jest wewnątrz <ul class="foo">
s:eq(*n)*	p:eq(3) {…}	Styl czwartego elementu p na stronie
s:gt(*n), s:lt(n)*	p:gt(3) {…}	Styl 5-tego i kolejnych elementów
s:even s:odd	tr:even {…}	Znajduje wszystkie rowki tabeli, które mają numery parzyste
s:first-child s:last-child s:only-child	tr:first-child {…}	Oznacza styl pierwszego elementu tabeli
s:nth-child(*n)*	tr:nth-child(3) {…}	Oznacza styl trzeciego rowka każdej tabeli
s:nth-child(even) s:nth-child(odd)	tr:nth-child(even) {…}	Oznacza styl rowków własnej tabeli o numerach parzystych
s:nth-child(*xn+y)*	tr:nth-child(4n+2) {…}	Oznacza styl rowków 6, 10, 14,… każdej tabeli

5.4. Selektory filtrowania zawartości

Selektor	Przykład definicji	Znaczenie
s:contains (text)	.foo li:contains(wow) {…}	Oznacza styl elementów **li**, które mają tekst **„wow”** w tekście strony i znajdują się wewnątrz **<…class="foo">**
s:empty	div:empty {…}	Oznaczają styl pustych elementów **div**
s:parent	div:parent {…}	Styl niepustych elementów **div**
s1:has(s2)	table:has(th) {…}	Styl wszystkich tabel (**table**), które mają przynajmniej jeden element **th**