JAVASERVER FACES

Ненко Табаков
Пламен Танов
Технологическо училище “Електронни системи”
Технически университет – София

14 декември 2008

ЛИТЕРАТУРА
НЕОБХОДИМИ ПРОГРАМИ

- The Java EE 5 Tutorial - http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf
- JSF Tags Reference - http://java.sun.com/javaee/javaserverfaces/1.2_MR1/docs/tlddocs/inde
- Примери - http://www.exadel.com/tutorial/jsf/jsftags-guide.html
- Step-by-step tutorial - http://balusc.blogspot.com/2008/01/jsf-tutorial-with-eclipse-and-tomca
- Случаи на употреба - http://www.coreservlets.com/JSF-Tutorial/
- Допълнителни - http://www.ibm.com/developerworks/views/java/libraryview.jsp?searcl
- Java API документация - http://java.sun.com/javase/6/docs/api/
- Eclipse - www.eclipse.org
- Apache Tomcat - http://tomcat.apache.org/

ВЪВЕДЕНИЕ

- Framework за създаване на WEB приложения
- Работи от страната на сървъра (server-side)
- Улеснява:
  - Навигацията между отделните страници
  - Извеждането на динамично съдържание
  - Валидация и преобразуването на въведените данни
  - Обработка на събития
  - Работата с **JavaBeans**
- Предимства:
  - Възможност за генериране на съдържание не само в **HTML**
  - Поддържа **MVC** архитектура
  - Използва **EL**
  - Лесна конфигурация
  - Възможност за създаване на нови компоненти

MODEL-VIEW-CONTROLLER (MVC)

Архитектурен модел, който предоставя набор от шаблони за дизайн (Design patterns), в които основното е логическото разделяне на едно приложение на следните части:

- Модел (Model)
  - Представлява моделът на данните, с които работи приложението
  - Отделните класове/таблици и връзките между тях
  - Целта е да не е обвързан с графичното представяне

MODEL-VIEW-CONTROLLER (MVC)

- Изглед (View)
  - Частта, чрез която потребителят взаимодейства с приложението
  - Има за цел да визуализира обработените данни от приложението (посредством **JSP**, **AWT**, **Swing**, ...) и да предостави възможност на потребителя да изпълнява команди (да променя данните)
  - В него не трябва да има логика, определяща поведението на приложението

MODEL-VIEW-CONTROLLER (MVC)

- Контролер (Controller)
  - Осъществява връзката между модела (model) и изгледа (view)
  - Контролерът е мястото където се намира логиката на приложението
  - Извлича необходимата информация от модела (например, от база данни) и предава резултата към изгледа, който се грижи за външното му оформление
  - Получава команди от потребителя, посредством изгледа, и обработва данните
  - Избира следващия "екран" на приложението (страница, диалогов прозорец, т.н.)

СТРУКТУРА

Едно **JSF** приложение може да се състои от:

- Набор от страници за визуализация на съдържание (**JSP**, **Facelets**, ...)
- Набор от **JavaBeans**, които дефинират свойства (property) и методи, използвани от компонентите на страницата
- Конфигурационен файл, който описва правилата за навигация между страниците, използваните **JavaBeans** обекти, потребителски компоненти и други
- Дескриптор за внедряване (deployment descriptor), чрез който приложението се представя пред **WEB** сървъра
- Набор от компоненти, създадени от потребителя – валидатори, конвертори, тагове и т.н.

ПРИМЕР

```
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<f:view>
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <title>Hello world example</title>
        </head>
        <body>
            <h:outputText value="Hello world" />
        </body>
    </html>
</f:view>
```

Всички **JSF** тагове са в `<f:view>`

Използвани библиотеки

Таг за извеждане на текст

ГРАФИЧНИ КОМПОНЕНТИ

- Създадени са тагове за всеки един стандартен компонент на **HTML** форма (`<input type=...>`):
  - http://java.sun.com/javaee/javaserverfaces/1.2_MR1/docs/tlddocs
- Предоставена е възможност за създаване на допълнителни тагове от потребителя (например, за взимане на дата)
- Има възможност да се закачат **JavaBeans** свойства (property), валидатори, конвертори и други към компонентите
- Графичните компоненти могат да съхраняват стойността си при последователни заявки към съответната страница (stateful)

```
<h:form id="login">
   <h:panelGrid border="1" columns="2">
<h:outputLabel for="username" value="username" />
<h:inputText   id="username" value="#{loginReq.username}" />
<h:inputSecret id="password" value="#{loginReq.password}" />
   </h:panelGrid>
</h:form>
```

НАВИГАЦИЯ МЕЖДУ СТРАНИЦИТЕ

- Всяка една препратка (Hyperlink) или бутон дефинира резултатен низ (outcome) или задава метод, който извършва дадено действие и връща този низ като резултат
- Когато потребител натисне върху препратка или бутон се извиква съответният метод и резултатния низ от него се сравнява с предварително дефинирани правила от вида:
  - Резултатен низ (outcome) – нова страница
  - При което се преминава към съответната нова страница
- Ако резултатният низ е **null** се остава на текущата страница
- Правилата се дефинират във файла `faces-config.xml`

```
<h:commandButton action="#{loginRequest.login}"
                                    value="Login" />
<h:commandLink action="register" value="here" />
```

Извиква се методът `login()` на `loginRequest`

Резултатният низ е константа (не се извиква метод)

ДЕФИНИРАНЕ НА ПРАВИЛА ЗА НАВИГАЦИЯ

```
<navigation-rule>
   <from-view-id>/login.jsp</from-view-id>
   <navigation-case>
      <from-outcome>posts</from-outcome>
      <to-view-id>/posts.jsp</to-view-id>
   </navigation-case>
</navigation-rule>

<navigation-rule>
   <from-view-id>/login.jsp</from-view-id>
   <navigation-case>
      <from-outcome>register</from-outcome>
      <to-view-id>/register.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

От коя страница

При какъв резулатен низ (**outcome**)

Следваща страница

ДЕФИНИРАНЕ НА ПРАВИЛА ЗА НАВИГАЦИЯ

- Правилата могат да се комбинират:

```
<navigation-rule>
   <from-view-id>/login.jsp</from-view-id>
   <navigation-case>
      <from-outcome>posts</from-outcome>
      <to-view-id>/posts.jsp</to-view-id>
   </navigation-case>
   <navigation-case>
      <from-outcome>register</from-outcome>
      <to-view-id>/register.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

От коя страница

При какъв резулатен низ (**outcome**)

Следваща страница

Второро правило от страница `login.jsp`

ДЕФИНИРАНЕ НА ПРАВИЛА ЗА НАВИГАЦИЯ

- Общи правила – без значение на коя страница се намира в момента потребителят при получаване на даден резултатен низ (outcome) се преминава към определена страница:

```
<navigation-rule>
   <from-view-id>*</from-view-id>
   <navigation-case>
      <from-outcome>login</from-outcome>
      <to-view-id>/login.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

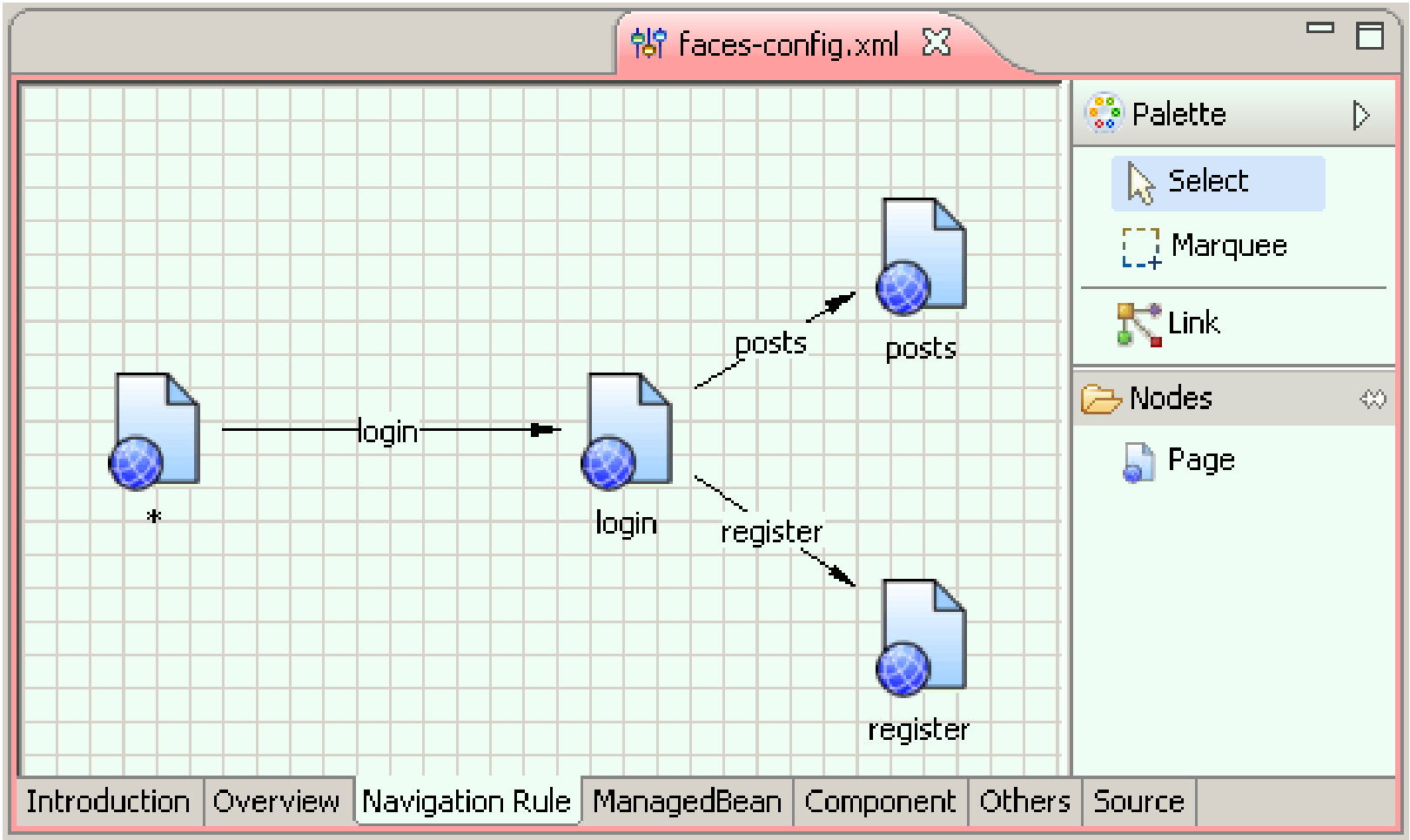
От всяка страница

При какъв резулатен низ (outcome)

Следваща страница

ДЕФИНИРАНЕ НА ПРАВИЛА ЗА НАВИГАЦИЯ

- Правилата могат да се задават и графично посредством Eclipse:



MANAGED BEANS

- **JSF** улеснява работата с **JavaBeans**
- Създават се при първо обръщение към тях
- За целта те трябва да бъдат описани във файла `faces-config.xml`:

```
    <managed-bean>
        <managed-bean-name>addPostRequest</managed-bean-name>

        <managed-bean-class>
org.elsysbg.courses.ip.jsf.simpleForum.request.AddPostRequest
        </managed-bean-class>

        <managed-bean-scope>request</managed-bean-scope>
        <managed-property>
            <property-name>text</property-name>
            <property-class>java.lang.String</property-class>
            <value>empty post</value>
        </managed-property>
    </managed-bean>
```

Името, с което ще бъде виждан през **EL**

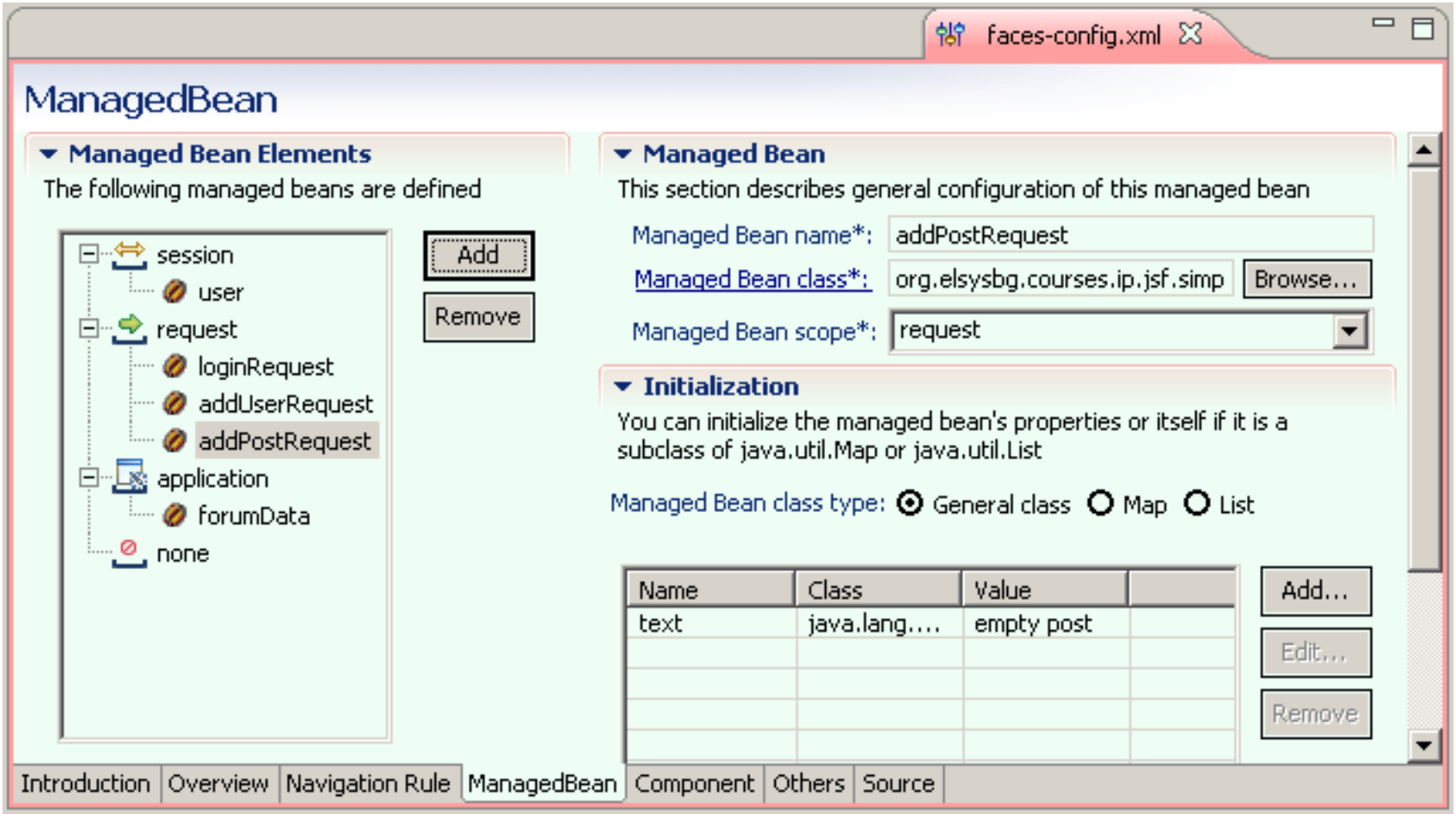
Класът на обекта

Обхват (scope) на **JavaBeans**

Инициализация на свойство (property)

MANAGED BEANS

- Могат да се описват и графично посредством Eclipse:



MANAGED BEANS

- Достъпа до тях се осъществява посредством **EL**
  - Достъп до свойство
    - Стойността се прочита и се обработва в зависимост от тага (например, визуализира)
      - Трябва да съществува съответен `getXXX()` метод
    - Ако бъде променена от потребителя тя автоматично се променя и в обекта (след като формата бъде изпратена)
      - Трябва да съществува съответен `setXXX()` метод

```
<h:inputText id="text" value="#{addPostRequest.text}" />
```

Достъп до свойството `text` на `addPostRequest`

MANAGED BEANS

- Достъпа до тях се осъществява посредством **EL**
  - Достъп до метод
    - След изпращането на формата и задаването на свързаните с нея свойства се извиква дадения метод
    - Трябва да е публичен, да връща `String` и да е без аргументи:
      - `public String doSomething() { ... }`
    - Изпълнява определена логика
    - Връща резултатен низ (outcome), чрез който **JSF** избира следващата страница

```
<h:commandButton value="Add Post"
                 action="#{addPostRequest.addPost}" />
```

Извикване на методът `addPost()` на `addPostRequest`

ОБХВАТ (SCOPE)

- Определя времето на живот на даден **managed bean**
- Възможни са четири стойности:
  - request
    - Съществуват в рамките на заявката: изпращане на заявка и генериране на отговор
  - session
    - Съществуват в рамките на сесията на потребителя (между различните заявки от един и същи потребител)
    - Докато тя не бъде преустановена (invalidate)
  - application
    - В рамките на приложението, достъпни до всички
  - none
    - Създава се всеки път, когато се направи обръщение към него, не се записва

MANAGED BEANS

- Създадените **managed bean** се съхраняват в асоциативни контейнери
- Могат да бъдат достигнати чрез:
  - **EL**
    - От изгледа (view)

```
<h:inputText id="text" value="#{addPostRequest.text}" />
<h:inputText id="text"
                value="#{requestScope.addPostRequest.text}" />

<h:dataTable var="post" value="#{forumData.posts}" > ...
<h:dataTable var="post"
             value="#{applicationScope.forumData.posts}" > ...
```

MANAGED BEANS

- Създадените **managed bean** се съхраняват в асоциативни контейнери
- Могат да бъдат достигнати чрез:
  - **EL**
  - Чрез контекста на **JSF** (**`FacesContext`**)
    - От контролер, **java** код

```
ExternalContext externalContext =
      FacesContext.getCurrentInstance().getExternalContext();

externalContext.getSessionMap().get("user");
externalContext.getRequestMap().get("addPostRequest");
externalContext.getApplicationMap().put("forumData",instance);
```

MANAGED BEANS

- Създадените **managed bean** се съхраняват в асоциативни контейнери
- Могат да бъдат достигнати чрез:
  - **EL**
  - Чрез контекста на **JSF** (**`FacesContext`**)
  - Чрез заявката (**`HttpServletRequest`**)
    - От филтри, сървлети, т.н., **java** код

```
request.getSession(true).getAttribute("user");
request.getSession(true).setAttribute("user", loggedUser);

request.getAttribute("addPostRequest");
request.setAttribute("addPostRequest", newPost);
```

МОДЕЛ НА КОМПОНЕНТИТЕ НА ПОТРЕБИТЕЛСКИЯ ИНТЕРФЕЙС

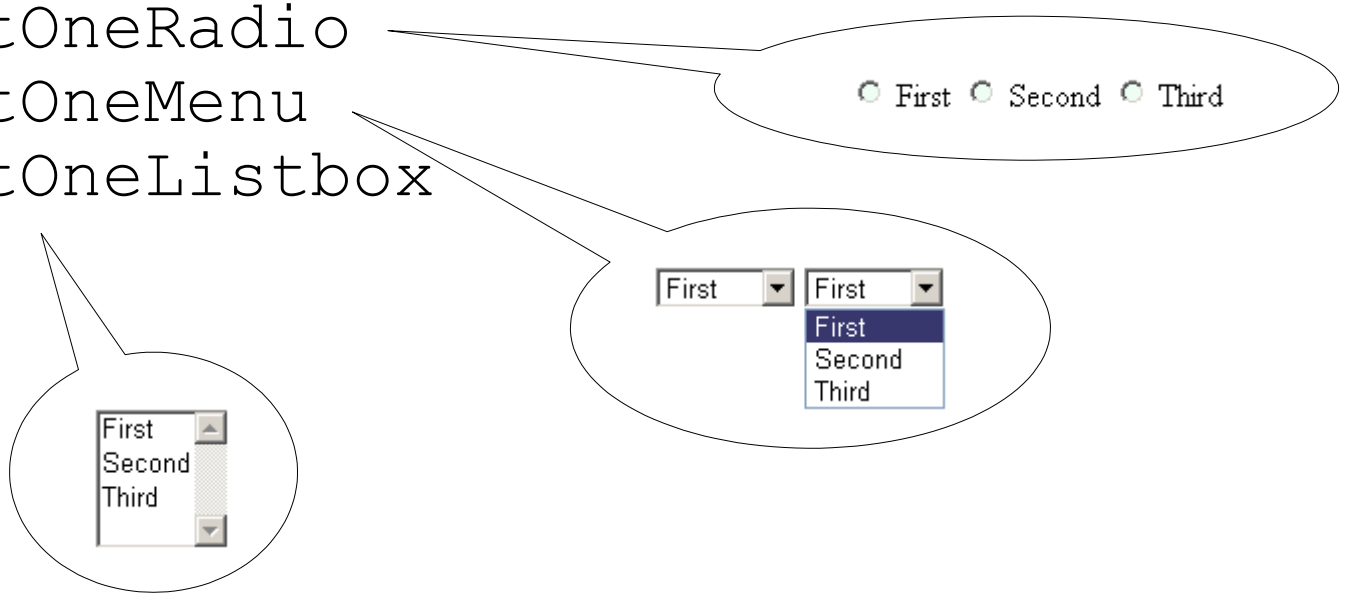
- **JSF** предоставя гъвкав механизъм за създаване на потребителски интерфейс
- Всеки един компонент може да бъде прост (например, бутон) или съставен (например, таблица), състояща се от други компоненти
- Съставен е от:
  - Йерархия от класове, които описват състоянието и поведението на компонентите на графичния интерфейс
    - Базов клас: `UIComponentBase`
  - Механизъм за визуализация на компоненти по различен начин
  - Механизъм за обработка на събития
  - Валидиране и преобразуване на данните на компонентите
- Предоставя възможност да бъде разширяван

ВИЗУАЛИЗАЦИЯ НА КОМПОНЕНТИТЕ

- Отделянето на състоянието и поведението от визуализацията води до следните преимущества:
  - Веднъж написано поведение на компонент може да бъде визуализирано:
    - По различни начини (например, като препратка (Hyperlink) или бутон)
    - За различен клиент (**HTML** клиент, **WML** клиент)
  - Дизайнерите могат да изберат подходящата визуализация в зависимост от дизайна на сайта за достигане до желаното от тях поведение
- Примери:
  - http://www.exadel.com/tutorial/jsf/jsftags-guide.html

RENDER KIT

- Определя как се визуализират компонентите за даден клиент
- **JSF** имплементацията включва вградена поддръжка за визуализация към **HTML** клиент
- Набор от класове, чрез които компонентите се визуализират
- Един компонент може да бъде визуализиран по няколко начина:
  - `UISelectOne` – избор на една стойност от множество
    - `h:selectOneRadio`
    - `h:selectOneMenu`
    - `h:selectOneListbox`



JSF ТАГОВЕ

- Списък на всички стандартни **JSF** тагове:
  - http://java.sun.com/javaee/javaserverfaces/1.2_MR1/docs/tlddocs
- Примери за визуализация:
  - http://www.exadel.com/tutorial/jsf/jsftags-guide.html
- Повечето от тях имат атрибут `rendered`
  - Препоръчително е да се ползва вместо `c:if` конструкция

```
<h:outputText value="There is no posts there"
                        rendered="#{empty forumData.posts}" />


<h:dataTable var="post" value="#{forumData.posts}"
                        rendered="#{!empty forumData.posts}">
...
</h:dataTable>
```

JSF ТАГОВЕ

- Списък на всички стандартни **JSF** тагове:
  - http://java.sun.com/javaee/javaserverfaces/1.2_MR1/docs/tlddocs
- Примери за визуализация:
  - http://www.exadel.com/tutorial/jsf/jsftags-guide.html
- Повечето от тях имат атрибут `rendered`
  - Препоръчително е да се ползва вместо `c:if` конструкция

```
<c:if test="${empty forumData.posts}">
   <h:outputText value="There is no posts there" />
</c:if>

<c:if test="${!empty forumData.posts}">
   <h:dataTable var="post" value="#{forumData.posts}" >
...
   </h:dataTable>
</c:if>
```

CSS СТИЛОВЕ И JSF

- Повечето таговете в **JSF** поддържат **CSS** стилове
- Посредством атрибути на таговете: `styleClass`, `style` и др.
- Гъвкав механизъм за промяна на стила на **HTML** елементи

```
<head>
   <link rel="stylesheet" type="text/css"
                                  href="css/dataTable.css" />
   ...
</head>
...
   <h:dataTable var="post" value="#{forumData.posts}"
      styleClass="postsTable"
      headerClass="headerAlignment"
columnClasses="numberColumn,userColumn,textColumn,dateColumn"
      rowClasses="oddRow,evenRow">
...
   </h:dataTable>
```

Използване на css файл в HTML

CSS класове за различни елементи

ПРИМЕР dataTable.css

```
.numberColumn {
    width: 10%;
    text-align: center;
}
.userColumn {
    width: 10%;
   text-align: right;
}
.textColumn {
    width: 50%;
   text-align: center;
}
.dateColumn {
   width: auto;
   text-align: left;
}
```

```
.headerAlignment {
text-align:center ! important;
}
.oddRow {
   background-color: #ddd;
}
.evenRow {
   background-color: #bbb;
}
.postsTable {
   width: 100%;
   padding: 0px;
   border: 1px solid #789DB3;
}
```

ПРИМЕР без CSS стилове

ID	User	Text	Date
1	admin	first	Mon Dec 01 17:27:37 EET 2008
2	user	second post	Mon Dec 01 17:27:52 EET 2008
3	user	third post	Mon Dec 01 17:27:58 EET 2008
4	user	last post	Mon Dec 01 17:28:02 EET 2008

```
<h:dataTable var="post" value="#{forumData.posts}">
...
</h:dataTable>
```

ПРИМЕР с **CSS** стилове

ID	User	Text	Date
1	admin	first	Mon Dec 01 17:24:45 EET 2008
2	user	second post	Mon Dec 01 17:25:11 EET 2008
3	user	third post	Mon Dec 01 17:26:05 EET 2008
4	user	last post	Mon Dec 01 17:26:10 EET 2008

```
<h:dataTable var="post" value="#{forumData.posts}"
    styleClass="postsTable"
    headerClass="headerAlignment"
columnClasses="numberColumn,userColumn,textColumn,dateColumn"
    rowClasses="oddRow,evenRow">
...
</h:dataTable>
```