Anno Accademico 2007-2008

Corso di Tecnologie Web
Web Application: Java Server Faces Core e HTML tags

http://www-lia.deis.unibo.it/Courses/TecnologieWeb0708/

JSF: standard Core Tags

Tag	Description
`view`	Creates the top-level view
`subview`	Creates a subview of a view
`facet`	Adds a facet to a component
`attribute`	Adds an attribute (key/value) to a component
`param`	Adds a parameter to a component
`actionListener`	Adds an action listener to a component
`valueChangeListener`	Adds a valuechange listener to a component
`converter`	Adds an arbitrary converter to a component
`convertDateTime`	Adds a datetime converter to a component
`convertNumber`	Adds a number converter to a component
`validator`	Adds a validator to a component
`validateDoubleRange`	Validates a `double` range for a component's value
`validateLength`	Validates the length of a component's value
`validateLongRange`	Validates a `long` range for a component's value
`loadBundle`	Loads a resource bundle, stores properties as a Map
`selectitems`	Specifies items for a select one or select many component
`selectitem`	Specifies an item for a select one or select many component
`verbatim`	Adds markup to a JSF page

JSF: standard HTML Tags

Tag	Description
`form`	HTML form
`inputText`	Single-line text input control
`inputTextarea`	Multiline text input control
`inputSecret`	Password input control
`inputHidden`	Hidden field
`outputLabel`	Label for another component for accessibility
`outputLink`	HTML anchor
`outputFormat`	Like `outputText`, but formats compound messages
`outputText`	Single-line text output
`commandButton`	Button: submit, reset, or pushbutton
`commandLink`	Link that acts like a pushbutton
`message`	Displays the most recent message for a component
`messages`	Displays all messages
`graphicImage`	Displays an image
`selectOneListbox`	Single-select listbox
`selectOneMenu`	Single-select menu

JSF: standard HTML Tags

Tag	Description
`selectOneRadio`	Set of radio buttons
`selectBooleanCheckbox`	Checkbox
`selectManyCheckbox`	Set of checkboxes
`selectManyListbox`	Multiselect listbox
`selectManyMenu`	Multiselect menu
`panelGrid`	HTML table
`panelGroup`	Two or more components that are laid out as one
`dataTable`	A feature-rich table control
`column`	Column in a `dataTable`

> Si tenga presente che i tag JSF rappresentano un componente e delegano la generazione del markup (HTML in questo caso) al renderer. Questo fatto garantisce circa la possibilità di adattare JSF a tecnologie di display dei dati alternative (ciò accade, per esempio, per le wireless JSF application)

JSF: attributi comuni ai tag

> Gli attributi condivisi tra i vari tag HTML vengono classificati tipicamente in 3 tipologie:

- basic
- HTML 4.0
- DHTML events

> Ogni tag, poi, prevede attributi che sono di sua esclusiva pertinenza (non sono cioè condivisi con altri tag – più precisamente, tali attributi non sono classificabili in categorie definite come quelle sopra citate)

JSF: attributi Basic

Attribute	Component Types	Description
`id`	A (25)	Identifier for a component
`binding`	A (25)	Reference to the component that can be used in a backing bean
`rendered`	A (25)	A boolean; `false` suppresses rendering
`styleClass`	A (23)	Cascading stylesheet (CSS) class name
`value`	I, O, C (19)	A component's value, typically a value binding
`valueChangeListener`	I (11)	A method binding to a method that responds to value changes
`converter`	I,O (15)	Converter class name
`validator`	I (11)	Class name of a validator that's created and attached to a component
`required`	I (11)	A boolean; if `TRue`, requires a value to be entered in the associated field

[a] A = all, I = input, O = output, C = commands, (n) = number of tags with attribute

JSF: attributi Basic

> L'attributo **`id`** consente di:

- Accedere a componenti JSF da altri tag

```
<h:inputText id="name".../> <h:message for="name"/>
```

- Ottenere riferimenti a componenti da codice Java

```
UIComponent component =
event.getComponent().findComponent("name");
```

- Accedere a componenti HTML via script

> L'attributo **`binding`** consente di associare un componente alla proprietà di una classe (backing bean):

```
<h:outputText binding="#{form.statePrompt}".../>
```

Ciò permette di manipolare il componente "agganciato" direttamente da codice Java, potendone ottenere il riferimento con una semplice invocazione al metodo getter della proprietà su cui è stato fatto il *binding*

JSF: attributi HTML

> I tag JSF supportano attributi che sono propri degli elementi HTML 4.0: tali attributi vengono definiti *pass-through*. Tale definizione deriva dal fatto che questi attributi vengono passati *as is* ai corrispondenti elementi HTML generati. Ogni attributo previsto per i diversi elementi HTML ha un omologo server side definito dal tag JSF. Per esempio:

```
<h:inputText value="#{form.name.last}" size="25".../>
```

genera l'elemento HTML seguente:

```
<input type="text" size="25".../>
```

JSF: attributi DHTML

Attribute	Description
`onblur (14)`	Element loses focus
`onchange (11)`	Element's value changes
`onclick (17)`	Mouse button is clicked over the element
`ondblclick (18)`	Mouse button is double-clicked over the element
`onfocus (14)`	Element receives focus
`onkeydown (18)`	Key is pressed
`onkeypress (18)`	Key is pressed and subsequently released
`onkeyup (18)`	Key is released
`onmousedown (18)`	Mouse button is pressed over the element
`onmousemove (18)`	Mouse moves over the element
`onmouseout (18)`	Mouse leaves the element's area
`onmouseover (18)`	Mouse moves onto an element
`onmouseup (18)`	Mouse button is released
`onreset (1)`	Form is reset
`onselect (11)`	Text is selected in an input field
`onsubmit (1)`	Form is submitted

JSF: form

> Il tag JSF form non prevede un attributo action, a differenza dell'omologo tag HTML. Se si utilizza il tag **`h:form`** senza alcun attributo, in una pagina denominata -ad esempio- **`/index.jsp`**, il renderer associato al componente JSF Form genera il seguente HTML:

```
<form id="_id0" method="post“ action="/forms/faces/index.jsp"
enctype="application/x-www-form-urlencoded">
```

Attributes	Description
`binding, id, rendered, styleClass`	Basic attributes[a]
`accept, acceptcharset, dir, enctype, lang, style, target, title`	HTML 4.0[b] attributes
`onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onreset, onsubmit`	DHTML events[c]

JSF: form e javascript

Form JSF

```
<h:form id="registerForm">

    <h:inputText id="password".../>

    <h:inputText id="passwordConfirm".../>

    ...

    <h:commandButton type="button“ onclick ="checkPassword(this.form)"/>


</h:form>
```

JS

```
function checkPassword(form) {
    var password = form["registerForm:password"].value;
    var passwordConfirm =    form["registerForm:passwordConfirm"].value;

    if (password == passwordConfirm)
        form.submit();
    else
        alert("Password and password confirm    fields don't    match");
}
```

HTML generato

```
<form id="registerForm" method="post"
     action="/javascript/faces/index.jsp"
    enctype="application/x-www-form-urlencoded">
    ...
    <input id="registerForm:password"
         type="text" name="registerForm:password"/>
    ...
    <input type="button" name="registerForm:_id5"
        value="Submit Form" onclick="checkPassword(this.form)"/>
    ...
</form>
```

JSF: text field e text area

Attributes	Description
`cols`	For `h:inputTextarea` only—number of columns
`immediate`	Process validation early in the life cycle
`redisplay`	For `h:inputSecret` only—when true, the input field's value is redisplayed when the web page is reloaded
`required`	Require input in the component when the form is submitted
`rows`	For `h:inputTextarea` only—number of rows
`valueChangeListener`	A specified listener that's notified of value changes
`binding, converter, id, rendered, required, styleClass, value, validator`	Basic attributes[a]
`accesskey, alt, dir, disabled, lang, maxlength, readonly, size, style, tabindex, title`	HTML 4.0 pass-through attributes[b]—`alt`, `maxlength`, and `size` do not apply to `h:inputTextarea`
`onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onselect`	DHTML events[c]

JSF: text field e text area

Example	Result
`<h:inputText value="#{form.testString}" readonly="true"/>`	12345678901234567890
`<h:inputSecret value="#{form.passwd}" redisplay="true"/>`	********** (shown after an unsuccessful form submit)
`<h:inputSecret value="#{form.passwd}" redisplay="false"/>`	(shown after an unsuccessful form submit)
`<h:inputText value="inputText" style="color: Yellow; background: Teal;"/>`	inputText
`<h:inputText value="1234567" size="5"/>`	123456
`<h:inputText value="1234567890" maxlength="6" size="10"/>`	123456

JSF: display di testo e immagini

> JSF fornisce 3 modalità per il display di testo e immagini

- `h:outputText`
- `h:outputFormat`
- `h:graphicImage`

> Attributi per **`h:outputText`** e **`h:outputFormat`**

Attributes	Description
`escape`	If set to `true`, escapes <, >, and & characters. Default value is `false`.
`binding, converter, id, rendered, styleClass, value`	Basic attributes[a]
`style, title`	HTML 4.0[b]

JSF: display di testo e immagini

> Attributi per **`h:graphicImage`**

Attributes	Description
`binding, id, rendered, styleClass, value`	Basic attributes[a]
`alt, dir, height, ismap, lang, longdesc, style, title, url, usemap, width`	HTML 4.0[b]
`onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup`	DHTML events[c]

JSF: display di testo e immagini

> Alcuni esempi:

```
<h:outputFormat value="{0} is {1} years old">
    <f:param value="Bill"/>
    <f:param value="38"/>
</h:outputFormat>
```

Example	Result
`<h:outputText value="#{form.testString}"/>`	12345678901234567890
`<h:outputText value="Number #{form.number}"/>`	Number 1000
`<h:outputText value="<input type='text' value='hello'/>"/>`	hello
`<h:outputText escape="true" value="<input type='text' value='hello'/>"/>`	`<input type="text" value="hello">`
`<h:graphicImage value="/tjefferson.jpg"/>`	

JSF: campi hidden

> Il tag JSF che genera un elemento HTML hidden è: **`h:inputHidden`**

Attribute	Description
`binding, converter, id, immediate, required, validator, value, valueChangeListener`	Basic attributes

JSF: bottoni e link

> JSF fornisce i seguenti tag per la gestione di bottoni e link:

- `h:commandButton`
- `h:commandLink`
- `h:outputLink`

> I tag **`h:commandButton`** e **`h:commandLink`** rappresentano componenti JSF di tipo *command* –il framework JSF lancia eventi e invoca azioni quando il bottone o il link è attivato.

> Il tag **`h:outputLink`** genera un **`anchor`** HTML che punta ad una determinata risorsa.

JSF: commandButton e commandLink - attributi

Attribute	Description
`action`	If specified as a string: Directly specifies an outcome used by the navigation handler to determine the JSF page to load next as a result of activating the button or link. If specified as a method binding: The method has this signature: `String methodName()`; the string represents the outcome
`actionListener`	A method binding that refers to a method with this signature: `void methodName(ActionEvent)`
`charset`	For `h:commandLink` only— The character encoding of the linked reference
`image`	For `h:commandButton` only— A context-relative path to an image displayed in a button. If you specify this attribute, the HTML `input`'s type will be `image`.
`immediate`	A boolean. If `false` (the default), actions and action listeners are invoked at the end of the request life cycle; if `true`, actions and action listeners are invoked at the beginning of the life cycle.
`type`	For h:commandButton: The type of the generated `input` element: `button`, `submit`, or `reset`. The default, unless you specify the `image` attribute, is `submit`. For h:commandLink: The content type of the linked resource; for example, `text/html`, `image/gif`, or `audio/basic`
`value`	The label displayed by the button or link. You can specify a string or a value reference expression.

JSF: commandButton e commandLink - attributi

Attribute	Description
accesskey, alt, binding, id, lang, rendered, styleClass, value	Basic attributes
coords (h:commandLink only), dir, disabled, hreflang (h:commandLink only), lang, readonly, rel (h:commandLink only), rev (h:commandLink only), shape (h:commandLink only), style, tabindex, target (h:commandLink only), title, type	HTML 4.0
onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onselect	DHTML events

JSF: commandButton - esempi

```
<h:commandButton value="submit"
type="submit"/>
```

submit

```
<h:commandButton value="reset"

   type="reset"/>
```

reset

```
<h:commandButton value="click this button..."

   onclick="alert('button clicked')"

   type="button"/>
```

click this button to execute JavaScript

```
<h:commandButton value="disabled"

disabled="#{not form.buttonEnabled}"/>
```

disabled

```
<h:commandButton value="#{form.buttonText}"

   type="reset"/>
```

press me

JSF: commandLink - esempi

```
<h:commandLink>
   <h:outputText value="register"/>
</h:commandLink>
```

register

```
<h:commandLink style="font-style: italic">
   <h:outputText value="#{msgs.linkText}"/>
</h:commandLink>
```

click here to register

```
<h:commandLink>
   <h:outputText value="#{msgs.linkText}"/>
   <h:graphicImage value="/registration.jpg"/>
</h:commandLink>
```

click here to register

```
<h:commandLink value="welcome"
   actionListener="#{form.useLinkValue}"
   action="#{form.followLink}">
```

welcome

```
<h:commandLink>
   <h:outputText value="welcome"/>
   <f:param name="outcome" value="welcome"/>
</h:commandLink>
```

welcome

Il renderer associato al componente gestito dal tag h:commandLink genera Javascript per avere un comportamento simile al bottone:

<a href="#“ onclick ="document.forms['_id0']['_id0:_id2'].value ='_id0:_id2';

document .forms['_id0'].submit()“ > register

JSF: outputLink - attributi

Attribute	Description
accesskey, binding, converter, id, lang, rendered, styleClass, value	Basic attributes
charset, coords, dir, hreflang, lang, rel, rev, shape, style, tabindex, target, title, type	HTML 4.0
onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup	DHTML events

JSF: outputLink - esempi

```
<h:outputLink value="http://java.net">
   <h:graphicImage value="java-dot-net.jpg"/>
   <h:outputText value="java.net"/>
</h:outputLink>
```

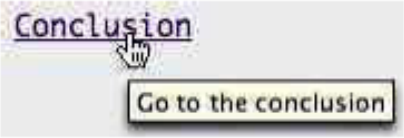

```
<h:outputLink value="#{form.welcomeURL}">
   <h:outputText value="#{form.welcomeLinkText}"/>
</h:outputLink>
```

go to welcome page

```
<h:outputLink value="#introduction">
   <h:outputText value="Introduction"
      style="font-style: italic"/>
</h:outputLink>
```

Introduction

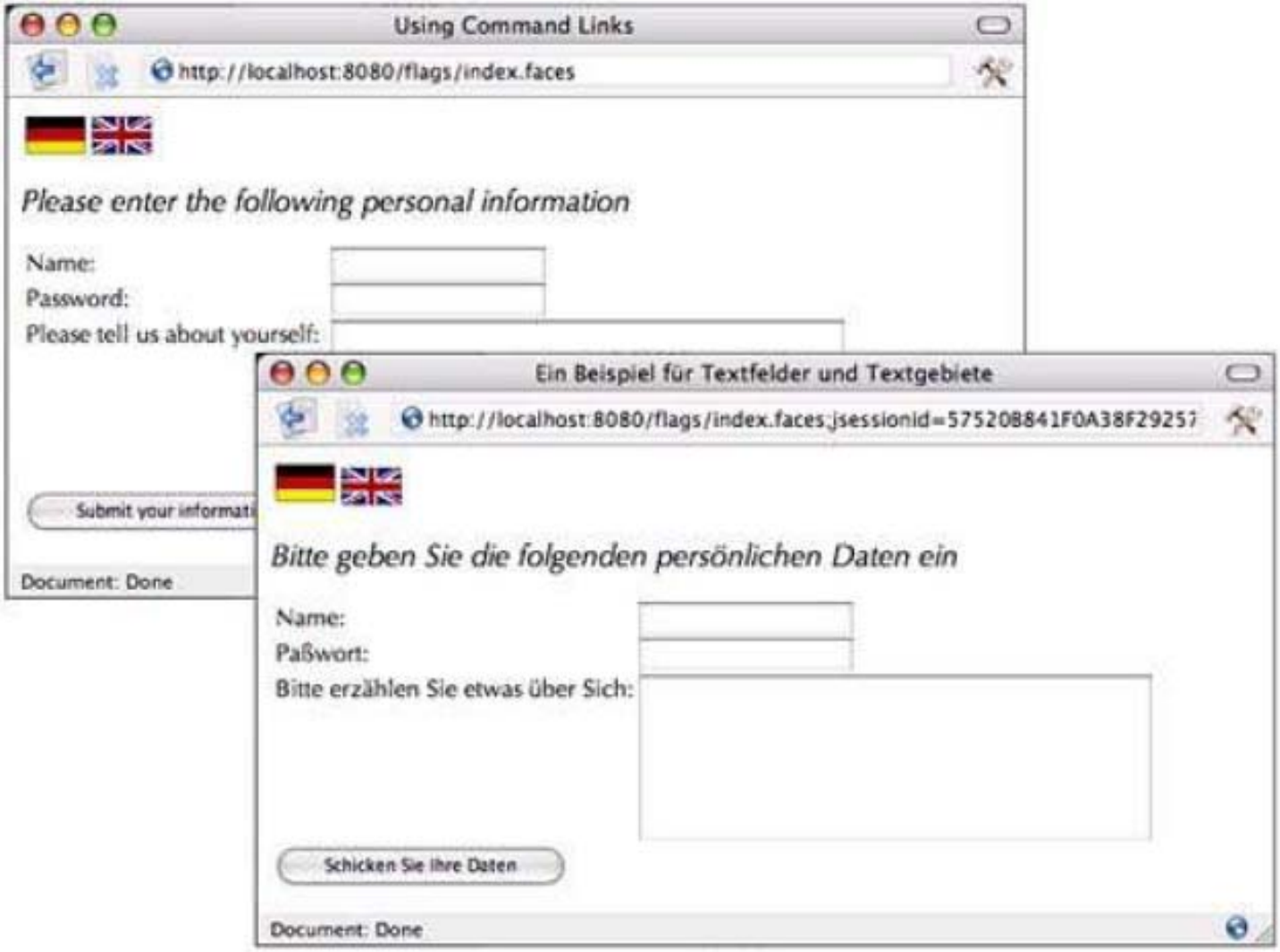
```
<h:outputLink value="#conclusion"
   title="Go to the conclusion">
   <h:outputText value="Conclusion"/>
</h:outputLink>
```




```
<h:outputLink value="#toc"
   title="Go to the table of contents">
   <f:verbatim>
      <h2>Table of Contents</h2>
   </f:verbatim>
</h:outputLink>
```

Table of Contents

JSF: esempio: la gestione del multilingua



JSF: esempio: la gestione del multilingua (index.jsp)

```
 1. <html>
 2.    <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
 3.    <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
 4.    <f:view>
 5.       <f:loadBundle basename="messages" var="msgs"/>
 6.       <head>
 7.          <link href="styles.css" rel="stylesheet" type="text/css"/>
 8.          <title>
 9.             <h:outputText value="#{msgs.indexWindowTitle}"/>
10.          </title>
11.       </head>
12.       <body>
13.          <h:form>
14.             <table>
15.                <tr>
16.                   <td>
17.                      <h:commandLink immediate="true"
18.                         action="#{localeChanger.germanAction}">
19.                         <h:graphicImage value="/german_flag.gif"
20.                            style="border: 0px"/>
21.                      </h:commandLink>
22.                   </td
23.                   <td>
24.                      <h:commandLink immediate="true"
25.                         action="#{localeChanger.englishAction}">
26.                         <h:graphicImage value="/britain_flag.gif"
27.                            style="border: 0px"/>
28.                      </h:commandLink>
29.                   </td>
30.                </tr>
31.             </table>
32.             <p>
33.                <h:outputText value="#{msgs.indexPageTitle}"
34.                   style="font-style: italic; font-size: 1.3em"/>
35.             </p>
36.             <table>
```

JSF: esempio: la gestione del multilingua (index.jsp)

```
37.                 <tr>
38.                    <td>
39.                       <h:outputText value="#{msgs.namePrompt}"/>
40.                    </td>
41.                    <td>
42.                       <h:inputText value="#{user.name}"/>
43.                    </td>
44.                 </tr>
45.                 <tr>
46.                    <td>
47.                       <h:outputText value="#{msgs.passwordPrompt}"/>
48.                    </td>
49.                    <td>
50.                       <h:inputSecret value="#{user.password}"/>
51.                    </td>
52.                 </tr>
53.                 <tr>
54.                    <td style="vertical-align: top">
55.                       <h:outputText value="#{msgs.tellUsPrompt}"/>
56.                    </td>
57.                    <td>
58.                       <h:inputTextarea value="#{user.aboutYourself}" rows="5"
59.                          cols="35"/>
60.                    </td>
61.                 </tr>
62.                 <tr>
63.                    <td>
64.                       <h:commandButton value="#{msgs.submitPrompt}"
65.                          action="thankYou"/>
66.                    </td>
67.                 </tr>
68.              </table>
69.           </h:form>
70.        </body>
71.     </f:view>
72. </html>
```

JSF: esempio: la gestione del multilingua (ChangeLocale)

```
public class ChangeLocaleBean {
   public String germanAction() {
      FacesContext context = FacesContext.getCurrentInstance();
      context.getViewRoot().setLocale(Locale.GERMAN);
      return null;
   }
   public String englishAction() {
      FacesContext context = FacesContext.getCurrentInstance();
      context.getViewRoot().setLocale(Locale.ENGLISH);
      return null;
   }
}
```

JSF: esempio: la gestione del multilingua (faces-config)

```
 1. <?xml version="1.0"?>
 2.
 3. <!DOCTYPE faces-config PUBLIC
 4. "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
 5. "http://java.sun.com/dtd/web-facesconfig_1_0.dtd">
 6.
 7. <faces-config>
 8.
 9.    <navigation-rule>
10.      <from-view-id>/index.jsp</from-view-id>
11.      <navigation-case>
12.         <from-outcome>thankYou</from-outcome>
13.         <to-view-id>/thankYou.jsp</to-view-id>
14.      </navigation-case>
15.   </navigation-rule>
16.
17.   <managed-bean>
18.      <managed-bean-name>localeChanger</managed-bean-name>
19.      <managed-bean-class>it.unibo.deis.ChangeLocaleBean</managed-bean-class>
20.      <managed-bean-scope>session</managed-bean-scope>
21.   </managed-bean>
22.
23.   <managed-bean>
24.      <managed-bean-name>user</managed-bean-name>
25.      <managed-bean-class>it.unibo.deis.UserBean</managed-bean-class>
26.      <managed-bean-scope>session</managed-bean-scope>
27.   </managed-bean>
28.
29. </faces-config>
```

JSF: tag di selezione

> JSF fornisce i seguenti tag per la selezione di elementi:

- `h:selectBooleanCheckbox`
- `h:selectManyCheckbox`
- `h:selectOneRadio`
- `h:selectOneListbox`
- `h:selectManyListbox`
- `h:selectOneMenu`
- `h:selectManyMenu`

> Con questi tag si generano i corrispondenti elementi HTML che rappresentano i classici controlli a scelta predefinita del form.

JSF: tag di selezione - attributi

Attributes	Description
disabledClass	CSS class for disabled elements—for h:selectOneRadio and h:selectManyCheckbox only
enabledClass	CSS class for enabled elements—for h:selectOneRadio and h:selectManyCheckbox only
layout	Specification for how elements are laid out: LINE_DIRECTION (horizontal) or PAGE_DIRECTION (vertical)—for h:selectOneRadio and h:selectManyCheckbox only
binding, converter, id, immediate, styleClass, required, rendered, validator, value, valueChangeListener	Basic attributes
accesskey, border, dir, disabled, lang, readonly, style, size, tabindex, title	HTML 4.0—border is applicable to h:selectOneRadio and h:selectManyCheckbox only. size is applicable to h:selectOneListbox and h:selectManyListbox only.
onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onselect	DHTML events

JSF: selectBooleanCheckbox

> Rappresenta un singolo checkbox che può essere legato ad una proprietà di tipo booleano

```
<h:selectBooleanCheckbox value="#{form.contactMe}"/>
```

> Metodi corrispondenti nel bean:

```
private boolean contactMe;

public void setContactMe(boolean newValue) {

    contactMe = newValue;

}

public boolean getContactMe() {

    return contactMe;

}
```

JSF: selectItem

> Il tag **`f:selectItem`** rappresenta un'opzione in un selettore (tutti i tag di selezione, eccetto **`selectBooleanCheckbox`**, utilizza **`f:selectItem`** per indicare gli item)

Attributes	Description
binding	Component binding for more information about component bindings.
id	Component ID
itemDescription	Description used by tools only
itemDisabled	Boolean value that sets the item's disabled property
itemLabel	Text shown by the item
itemValue	Item's value, which is passed to the server as a request parameter
value	Value binding expression that points to a SelectItem instance

JSF: selectManyCheckbox

> Rappresenta il checkbox a selezione multipla

```
<h:selectManyCheckbox value="#{form.colors}">
    <f:selectItem itemValue="Red"/>
    <f:selectItem itemValue="Blue"/>
    <f:selectItem itemValue="Yellow"/>
</h:selectManyCheckbox>
```

> HTML generato:

```
<table>
    <tr>
        <td>
            <label for="_id2:_id14">
                <input name="_id2:_id14" value="Red" type="checkbox">Red</input>
            </label>
        </td>
    </tr>
    ...
</table>
```

JSF: selectOneRadio

> Rappresenta il radio button:

```
<h:selectOneRadio value="#{form.education}">
    <f:selectItem itemValue="High School"/>
    <f:selectItem itemValue="Bachelor's"/>
    <f:selectItem itemValue="Master's"/>
    <f:selectItem itemValue="Doctorate"/>
</h:selectOneRadio>
```

> HTML generato:

```
<table>
  <tr>
    <td>
      <label for="_id2:_id14">
        <input name="_id2:_id14" value="High School" type="radio">High School</input>
      </label>
    </td>
  </tr>
  ...
</table>
```

JSF: menu e listbox

> Rappresentano le select list a selezione singola e multipla:

```
<h:selectManyListbox value="#{form.languages}">
    <f:selectItem itemValue="English"/>
    <f:selectItem itemValue="French"/>
    <f:selectItem itemValue="Italian"/>
    <f:selectItem itemValue="Spanish"/>
    <f:selectItem itemValue="Russian"/>
</h:selectManyListbox>
```

```
<h:selectOneMenu value="#{form.day}">
    <f:selectItem itemValue="Sunday"/>
    <f:selectItem itemValue="Monday"/>
    <f:selectItem itemValue="Tuesday"/>
    <f:selectItem itemValue="Wednesday"/>
    <f:selectItem itemValue="Thursday"/>
    <f:selectItem itemValue="Friday"/>
    <f:selectItem itemValue="Saturday"/>
</h:selectOneMenu>
```

> HTML generato:

```
<select name="_id2:_id11" multiple>
    <option value="English">English</option>
    <option value="French">French</option>
    ...
</select>
```

```
<select name="_id2:_id17" size="1">
    <option value="Sunday">Sunday</option>
    ...
</select>
```

JSF: javax.faces.model.SelectItem

> Si può utilizzare l'attributo **value** di un qualsiasi selettore per accedere a un'istanza di **SelectItem** associata alla proprietà di un bean (usando, come sempre, una value binding expression):

```
<f:selectItem value="#{form.cheeseItem}"/>
```

```
public SelectItem getCheeseItem()  {
    return new SelectItem("Cheese");
}
```

> **javax.faces.model.SelectItem** prevede i seguenti costruttori:

- **SelectItem (Object value)**
- **SelectItem (Object value, String label)**
- **SelectItem (Object value, String label, String description)**
- **SelectItem (Object value, String label, String description, boolean disabled)**

JSF: selectItems

> Per rappresentare un numero variabile di opzioni in forma compatta si utilizza **f:selectItems:**

```
<h:selectOneRadio>
    <f:selectItems value="#{form.condiments}"/>
</h:selectOneRadio>
```

> La value binding expression **#{form.condiments}** punta ad un array di **SelectItem**:

```
private SelectItem[] condiments = {
    new SelectItem(new Integer(1), "Cheese"),
    new SelectItem(new Integer(2), "Pickle"),
    new SelectItem(new Integer(3), "Mustard"),
    new SelectItem(new Integer(4), "Lettuce"),
    new SelectItem(new Integer(5), "Onions")
};

public SelectItem[] getCondiments() {
    return condiments;
}
```

JSF: selectItems

> **`f:selectItems`** prevede un attributo **`value`** la cui value binding expression punta, in generale, a:

- una istanza singola di **`SelectItem`**
- una collection di **`SelectItem`**
- un array di **`SelectItem`**

> In alternativa si può specificare una **`Map`** come proprietà agganciata dalla value binding expression: in questo caso l'implementazione JSF crea una **`SelectItem`** per ogni entry nella mappa. La chiave dell'entry della mappa è usata come label dell'item, il valore dell'entry della mappa è usato come valore dell'item:

```
private Map condiments = null;

public Map getCondiments() {
    if(condiments == null) {
        condiments = new HashMap();
        condiments.put("Cheese",  new Integer(1)); // key,value
        condiments.put("Pickle",  new Integer(2));
        condiments.put("Mustard", new Integer(3));
    }
    return condiments;
}
```

JSF: item groups

> Si possono raggruppare gli item di un menu o listbox nel modo seguente:

```
<h:selectManyListbox>
    <f:selectItems value="#{form.menuItems}"/>
</h:selectManyListbox>
```

In cui la property **`menuItems`** è un array **`SelectItem.`** Si utilizza poi un'istanza di **`javax.faces.model.SelectItemGroup`** per ottenere il raggruppamento. Si noti (esempio slide successiva) come **`SelectItemGroup`** estenda **`SelectItem`** (viene utilizzata, infatti, per popolare un array di **`SelectItem`**)

JSF: item groups

```
public SelectItem[] getMenuItems() { return menuItems; }
private static SelectItem[] menuItems = { burgers,  beverages, condiments };
private SelectItemGroup burgers =
    new  SelectItemGroup("Burgers", // value
                "burgers on the menu", // description
                false, // disabled
                burgerItems); // select items

   private SelectItemGroup beverages =
        new SelectItemGroup("Beverages", // value
                "beverages on the menu", // description
                false, // disabled
                beverageItems); // select items

   private SelectItemGroup condiments =
        new SelectItemGroup("Condiments", // value
                "condiments on the menu", // description
                false, // disabled
                condimentItems); // select items
```

```
private SelectItem[] burgerItems = {
    new SelectItem("Qwarter pounder"),
    new SelectItem("Single"),
    new SelectItem("Veggie"),
};
private SelectItem[] beverageItems = {
    new SelectItem("Coke"),
    new SelectItem("Pepsi"),
    new SelectItem("Water"),
    new SelectItem("Coffee"),
    new SelectItem("Tea"),
};
private SelectItem[] condimentItems = {
    new SelectItem("cheese"),
    new SelectItem("pickle"),
    new SelectItem("mustard"),
    new SelectItem("lettuce"),
    new SelectItem("onions"),
};
```

JSF: il binding dell'attributo value

> Per tenere traccia della selezione dell'utente si utilizza l'attributo **`value`** presente nei diversi selettori:

```
<h:selectOneRadio value="#{form.education}">
   <f:selectItems value="#{form.educationItems}"/>
</h:selectOneRadio>

private Integer education = null;
public Integer getEducation() {
    return education;
}
public void setEducation(Integer newValue) {
    education = newValue;
}
```

> L'espressione **`#{form.education}`** referenzia la property **`education`** del bean denominato **`form.`** Si noti che la property è di tipo **`Integer`** e di tale tipo <u>devono</u> essere anche i valori dei radio button (i tipi devono coincidere: nel caso di tipi custom si specifica un *converter*).

JSF: messaggi

> Durante il lifecycle JSF, ogni oggetto può creare un messaggio e aggiungerlo ad una coda messaggi mantenuta dal contesto. Alla fine del lifecycle -nella fase di *render response*- tali messaggi possono essere visualizzati nella vista. Tipicamente i messaggi sono associati ad errori di validazione relativi ai diversi componenti

> I messaggi sono classificati in 4 categorie:

- **`information`**
- **`warning`**
- **`error`**
- **`fatal`**

JSF: messaggi - attributi

Attributes	Description
errorClass	CSS class applied to error messages
errorStyle	CSS style applied to error messages
fatalClass	CSS class applied to fatal messages
fatalStyle	CSS style applied to fatal messages
globalOnly	Instruction to display only global messages-applicable only to h:messages. Default: false
infoClass	CSS class applied to information messages
infoStyle	CSS style applied to information messages
layout	Specification for message layout: table or list-applicable only to h:messages

JSF: messaggi - attributi

Attributes	Description
showDetail	A boolean that determines whether message details are shown. Defaults are false for h:messages, true for h:message.
showSummary	A boolean that determines whether message summaries are shown. Defaults are true for h:messages, false for h:message.
tooltip	A boolean that determines whether message details are rendered in a tooltip; the tooltip is only rendered if showDetail and showSummary are true
warnClass	CSS class for warning messages
warnStyle	CSS style for warning messages
binding, id, rendered, styleClass	Basic attributes
style, title	HTML 4.0

JSF: messaggi - esempio

> Di seguito viene illustrato un utilizzo dei tag JSF relativi ai messaggi:

```
<h:form>
    <h:messages layout="table" errorClass="errors"/>
    ...
    <h:inputText id="name" required="true"/>
    <h:message for="name" errorClass="errors"/>
    ...
    <h:inputText id="date" value="#{form.date}" required="true"/>
    <h:message for="date" errorClass="errors"/>
    ...
</form>
```