JSF 2 Composite Components
Part II – Input Components

JSF 2.2 Version

Originals of slides and source code for examples: http://www.coreservlets.com/JSF-Tutorial/jsf2/
Also see the PrimeFaces tutorial – http://www.coreservlets.com/JSF-Tutorial/primefaces/
and customized JSF2 and PrimeFaces training courses – http://courses.coreservlets.com/jsf-training.html

Customized Java EE Training: http://courses.coreservlets.com/
Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at your location.

For live training on JSF 2, PrimeFaces, or other Java EE topics, email hall@coreservlets.com
Marty is also available for consulting and development support

Taught by the author of Core Servlets and JSP, this tutorial, and JSF 2.2 version of Core JSF. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - JSF 2, PrimeFaces, Ajax, jQuery, Spring MVC, JSP, Android, general Java, Java 8 lambdas/streams, GWT, custom topic mix
    - Courses available in any location worldwide. Maryland/DC area companies can also choose afternoon/evening courses.
  - Courses developed and taught by coreservlets.com experts (edited by Marty)
    - Hadoop, Spring, Hibernate/JPA, RESTful Web Services

Contact hall@coreservlets.com for details
Topics in This Section

- Idea
- Making basic input components
- Using the “type” attribute
- Giving components their own CSS file

Overview: Input Components
Input Components

**Idea**
- A composite component that will go inside h:form and be used to gather input from the user

**Features**
- Must be associated with a bean value
  - Use “type” attribute to enforce the type of the value
- Often has its own style sheet
  - Load with h:outputStylesheet
- May need to assemble final value out of submitted pieces
- May need to call component setter methods based on attribute values
- May have embedded Ajax behavior
- May be built out of a rich JavaScript GUI element
Input Components: Basic Steps

- **Use JSF input element(s) in cc:implementation**
  - E.g., h:inputText, h:selectOneMenu, etc.
- **Do not include h:form**
  - Otherwise your component cannot be combined with other input components in a single form
- **In most cases, name the main attribute “value”**
  - To be consistent with standard input elements
  - Similarly, mimic other standard attribute names (styleClass, required, etc.)
- **Use the “type” attribute of cc:attribute to hook the element(s) to bean properties. Use fully qualified name.**
  - `<cc:attribute name="value" required="true" type="coreservlets.Nameable"/>`
- **Except for builtin types, have “type” refer to an interface**
  - For flexibility, so component can be used in many places

Example: inputName1

- **Idea**
  - Collect a first and last name
- **Code summary**
  - Use two h:inputText elements
  - Make a Nameable interface that defines getFirstName, setFirstName, getLastName, and setLastName (i.e., firstName and lastName bean properties). Have “type” refer to coreservlets.Nameable
  - Add optional properties with defaults to specify the prompts in front of the two textfields
package coreservlets;

public interface Nameable {
    public String getFirstName();
    public void setFirstName(String firstName);
    public String getLastName();
    public void setLastName(String lastName);
}

• Filename: inputName1.xhtml
  – In WebContent/resources/utils

• Basic code
  <!DOCTYPE html ...>
  <html xmlns="http://www.w3.org/1999/xhtml"
       xmlns:h="http://xmlns.jcp.org/jsf/html"
       xmlns:cc="http://xmlns.jcp.org/jsf/composite">
    <cc:interface>
      Declare value, firstNamePrompt, lastNamePrompt, required, requiredMessage
    </cc:interface>
    <cc:implementation>
      Make two input fields, one for first name, one for last name.
    </cc:implementation>
  </html>
Component: Interface Section

```xml
<cc:interface>
    <cc:attribute name="value" required="true" type="coreservlets.Nameable"/>
    <cc:attribute name="firstNamePrompt" required="false" default="First Name"/>
    <cc:attribute name="lastNamePrompt" required="false" default="Last Name"/>
    <cc:attribute name="required" required="false" default="true"/>
    <cc:attribute name="requiredMessage" required="false" default="Required!"/>
</cc:interface>
```

Component: Implementation Section

```xml
<cc:implementation>
    <h:panelGrid columns="3">
        #{cc.attrs.firstNamePrompt}:
        <h:inputText value="#{cc.attrs.value.firstName}" id="firstName"
                      required="#{cc.attrs.required}"
                      requiredMessage="#{cc.attrs.requiredMessage}"/>
        <h:message for="firstName" style="font-weight: bold; color: red"/>
        #{cc.attrs.lastNamePrompt}:
        <h:inputText value="#{cc.attrs.value.lastName}" id="lastName"
                      required="#{cc.attrs.required}"
                      requiredMessage="#{cc.attrs.requiredMessage}"/>
        <h:message for="lastName" style="font-weight: bold; color: red"/>
    </h:panelGrid>
</cc:implementation>
```
Example Usage 1: Employee of Month

- **Idea**
  - Nominate employee of the month for the corporation

- **Managed Bean**
  - Implements Nameable
  - Also stores a nomination reason

- **Page that uses component**
  - Collects name and nomination reason

- **Results page**
  - Shows confirmation with first name, last name, and nomination reason

Managed Bean (First Half)

```java
@ManagedBean
class EmployeeOfMonth implements Nameable {
    private String firstName, lastName;
    private String nominationReason;

    public String getFirstName() {
        return(firstName);
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return(lastName);
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
Managed Bean (Second Half)

```java
public String getNominationReason() {
    return(nominationReason);
}

public void setNominationReason(String nominationReason) {
    this.nominationReason = nominationReason;
}

public String recordVote() {
    storeVoteSomehow();
    return("show-employee-vote");
}

private void storeVoteSomehow() {
    // Nothing to see here. Move along.
}
```

Facelets Page that Uses Component

```html
...<fieldset>
<legend>Employee of the Month</legend>
<h2>Nominate this Month's Top Employee!</h2>
<h:form>
    <utils:inputName1 value="#{employeeOfMonth}"/>
    Reason:
    <h:inputText value="#{employeeOfMonth.nominationReason}" size="75"/><br/>
    <h:commandButton action="#{employeeOfMonth.recordVote}" value="Record Vote"/>
</h:form>
</fieldset>
...```
Results Page

...<fieldset><legend>Employee of Month</legend><ul><li>Name: #{employeeOfMonth.firstName} #{employeeOfMonth.lastName}</li><li>Reason: #{employeeOfMonth.nominationReason}</li></ul></fieldset>...

Result: Page that Uses Component

Original (blank) form

After submitting with missing first/last names
Usage 2: Write-in Ballot for US General Election (Maryland)

- **Idea**
  - Lets voters enter a name for president and vice president, and also vote on two local (state) initiatives.

- **Managed Bean**
  - Parent class: uses Person, which implements Nameable, has getters/setters for the two state propositions, has getters for the descriptions of the propositions
  - Subclass: provides descriptions of the propositions, has action controller method that verifies that president and vice president are different people

- **Page that uses component**
  - Collects president’s name, vice president’s name, and yes/no votes on the two propositions

- **Results page**
  - Shows confirmation for president, VP, and propositions
public class WriteinBallot {
    private final Person president, vicePresident;
    private boolean forProp1, forProp2;
    private final String prop1Description, prop2Description;

    public WriteinBallot(String prop1Description,
                          String prop2Description) {
        this.prop1Description = prop1Description;
        this.prop2Description = prop2Description;
        forProp1 = true;
        forProp2 = true;
        president = new Person();
        vicePresident = new Person();
    }

    public Person getPresident() {
        return (president);
    }

    public Person getVicePresident() {
        return (vicePresident);
    }

    public boolean isForProp1() {
        return (forProp1);
    }

    public void setForProp1(boolean forProp1) {
        this.forProp1 = forProp1;
    }

    public String getProp1Description() {
        return (prop1Description);
    }
}
Parent Class of Managed Bean (Continued)

```java
public boolean isForProp2() {
    return (forProp2);
}

public void setForProp2(boolean forProp2) {
    this.forProp2 = forProp2;
}

public String getProp2Description() {
    return (prop2Description);
}
```
Person (Continued)

    public String getFirstName() {
        return(firstName);
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return(lastName);
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getFullName() {
        return(firstName + " " + lastName);
    }

Managed Bean (First Half)

@ManagedBean
public class MarylandBallot extends WriteinBallot {
    public MarylandBallot() {
        super("To require all Maryland elementary schools " +
             "to teach Java programming",
             "To have Java programming replace lacrosse " +
             "as the official state team sport");
    }
}
Managed Bean (Second Half)

```java
public String checkBallot() {
    Person pres = getPresident();
    Person vp = getVicePresident();
    if (pres.getFirstName().equals(vp.getFirstName()) &&
        pres.getLastName().equals(vp.getLastName())) {
        FacesContext context = FacesContext.getCurrentInstance();
        FacesMessage errorMessage =
            new FacesMessage("Cannot vote for same person for " +
                            "President and Vice President");
        context.addMessage(null, errorMessage);
        return(null);
    } else {
        return("show-ballot-vote");
    }
}
```

Facelets Page that Uses Component

```html
...<h:form>
    <b>President</b><br/>
    <utils:inputName1 value="#{marylandBallot.president}"/><br/>
    <b>Vice President:</b><br/>
    <utils:inputName1 value="#{marylandBallot.vicePresident}"/><br/>
    <b>#{marylandBallot.prop1Description}</b>
    <h:selectOneRadio value="#{marylandBallot.forProp1}"
                      itemValue="true" itemLabel="For"/>
    <f:selectItem itemValue="false" itemLabel="Against"/>
    </h:selectOneRadio>
    <b>#{marylandBallot.prop2Description}</b>
    <h:selectOneRadio value="#{marylandBallot.forProp2}"
                      itemValue="true" itemLabel="For"/>
    <f:selectItem itemValue="false" itemLabel="Against"/>
    </h:selectOneRadio>
    <br/>
    <h:commandButton action="#{marylandBallot.checkBallot}"
                      value="Submit Vote"/>
</h:form>...
```
<fieldset>
<legend>Maryland Write-in Ballot</legend>
<ul>
<li>President: 
  #{marylandBallot.president.fullName}</li>
<li>Vice President: 
  #{marylandBallot.vicePresident.fullName}</li>
<li>Voted for Proposition 1 
  (#{marylandBallot.prop1Description}): 
  #{marylandBallot.forProp1}</li>
<li>Voted for Proposition 2 
  (#{marylandBallot.prop2Description}): 
  #{marylandBallot.forProp2}</li>
</ul>
</fieldset>

Result: Page that Uses Component

Original (blank) form

After submitting with missing first/last names
Giving Stylesheets to Components
Overview

**Motivation**
- CSS-based alignment is often needed by components that consist of multiple pieces
  - E.g., for name component, make table cell that holds the prompts be right-aligned, and table cell that holds the textfields be left-aligned
- Fonts and colors should be overridable by page author

**Basic steps**
- Create a “css” (or similar) subfolder within your component folder (which, in turn, is inside “resources”)
- Put style sheet in that folder
- Load the style sheet with
  `<h:outputStylesheet name="stylesheet-name.css" library="component-folder/css"/>
  - Even though this is not in “head” section of original file, the HTML tag to load style sheet will be in head of rendered page

Component Outline

**Filename: inputName2.xhtml**
- In WebContent/resources/utils

**Basic code**
```xml
<!DOCTYPE html ...>
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html"
     xmlns:cc="http://xmlns.jcp.org/jsf/composite">

<cc:interface>
    Declare value, firstNamePrompt, lastNamePrompt, required, requiredMessage, errorStyleClass
</cc:interface>
<cc:implementation>
    Make two input fields, one for first name, one for last name. Align with CSS. CSS for errors.
</cc:implementation>
</html>
```
Component: Interface Section

```xml
<cc:interface>
  <cc:attribute name="value" required="true"
    type="coreservlets.Nameable"/>
  <cc:attribute name="firstNamePrompt" required="false"
    default="First Name"/>
  <cc:attribute name="lastNamePrompt" required="false"
    default="Last Name"/>
  <cc:attribute name="required" required="false"
    default="true"/>
  <cc:attribute name="requiredMessage" required="false"
    default="Required!"/>
  <cc:attribute name="errorStyleClass" required="false"
    default="name-error"/>
</cc:interface>
```

Component: Implementation Section (First Half)

```xml
<cc:implementation>
  <h:outputStylesheet name="utils-styles.css"
    library="utils/css"/>
  <h:panelGrid columns="3" styleClass="formTable">
    #{cc.attrs.firstNamePrompt}:
    <h:inputText value="#{cc.attrs.value.firstName}"
      id="firstName"
      required="#{cc.attrs.required}"
      requiredMessage="#{cc.attrs.requiredMessage}"/>
    <h:message for="firstName" styleClass="#{cc.attrs.errorStyleClass}"/>
  </h:panelGrid>
</cc:implementation>
```
Component: Implementation Section (Second Half)

```java
#{cc.attrs.lastNamePrompt}:
<h:inputText value="#{cc.attrs.value.lastName}"
    id="lastName"
    required="#{cc.attrs.required}"
    requiredMessage="#{cc.attrs.requiredMessage}"/>
<h:message for="lastName"
    styleClass="#{cc.attrs.errorStyleClass}"/>
</h:panelGrid></cc:implementation>
```

Examples: Employee of Month and Maryland Ballot

- **CSS**
  - R-aligns the prompts
  - Makes error messages bold and red by default, but can be overridden by passing in errorStyleClass
- **Managed beans**
  - Reused from previous examples without changes
- **Page that uses component**
  - Mostly same as previous examples, but optionally supplies an errorStyleClass
- **Results pages**
  - Reused from previous examples without changes
Employee of Month: Facelets Page that Uses Component

...<h2>Nominate this Month's Top Employee!</h2>
<h:form>
  <util:inputName2 value="#{employeeOfMonth}"/>
  Reason:
  <h:inputText value="#{employeeOfMonth.nominationReason}"
size="75"/><br/>
  <h:commandButton action="#{employeeOfMonth.recordVote}"
value="Record Vote"/>
</h:form>
...

Result: Employee of Month (Input Form)
Result: Employee of Month (Results Page)

Vote Confirmation

Employee of Month

- Name: Julia Schippers
- Reason: Avoiding whooping cough

Write-in Ballot: Facelets Page that Uses Component (1st Half)

...<h:form>
<b>President</b><br/>
<utils:inputName2 value="#{marylandBallot.president}"
    firstNamePrompt="First (Given) Name"
    lastNamePrompt="Last (Family) Name"
    errorStyleClass="blue-error"/><br/>
<b>Vice President:</b><br/>
<utils:inputName2 value="#{marylandBallot.vicePresident}"
    firstNamePrompt="First (Given) Name"
    lastNamePrompt="Last (Family) Name"
    errorStyleClass="blue-error"/><br/>
<b>&{marylandBallot.prop1Description}</b>
<h:selectOneRadio value="&{marylandBallot.forProp1}"
  <f:selectItem itemValue="true" itemLabel="For"/>
  <f:selectItem itemValue="false" itemLabel="Against"/>
</h:selectOneRadio>

<b>&{marylandBallot.prop2Description}</b>
<h:selectOneRadio value="&{marylandBallot.forProp2}"
  <f:selectItem itemValue="true" itemLabel="For"/>
  <f:selectItem itemValue="false" itemLabel="Against"/>
</h:selectOneRadio>
<br/>
<h:commandButton action="&{marylandBallot.checkBallot}" value="Submit Vote"/>

Result: Write-in Ballot (Input Form)

Original (blank) form. Same as previous except for prompt alignment.

After submitting with missing first/last names. Same as previous except for style of error messages.
Result: Write-in Ballot
(Results Page)

Vote Confirmation

Maryland Write-in Ballot

- President: Bill Gates
- Vice President: Steve Ballmer
- Voted for Proposition 1 (To require all Maryland elementary schools to teach Java programming): false
- Voted for Proposition 2 (To have Java programming replace lacrosse as the official state team sport): false

Wrap-Up
Summary

- **Associate input element with bean value**
  - Use “type” attribute to enforce the type of the value
  - Except for built-in types, use interface, not class, for the value

- **Use style sheets**
  - Load with h:outputStylesheet
  - Often used for alignment
  - For fonts and colors, let page author override your styles