JavaServer Faces

The Next Big Thing for Enterprise Java!

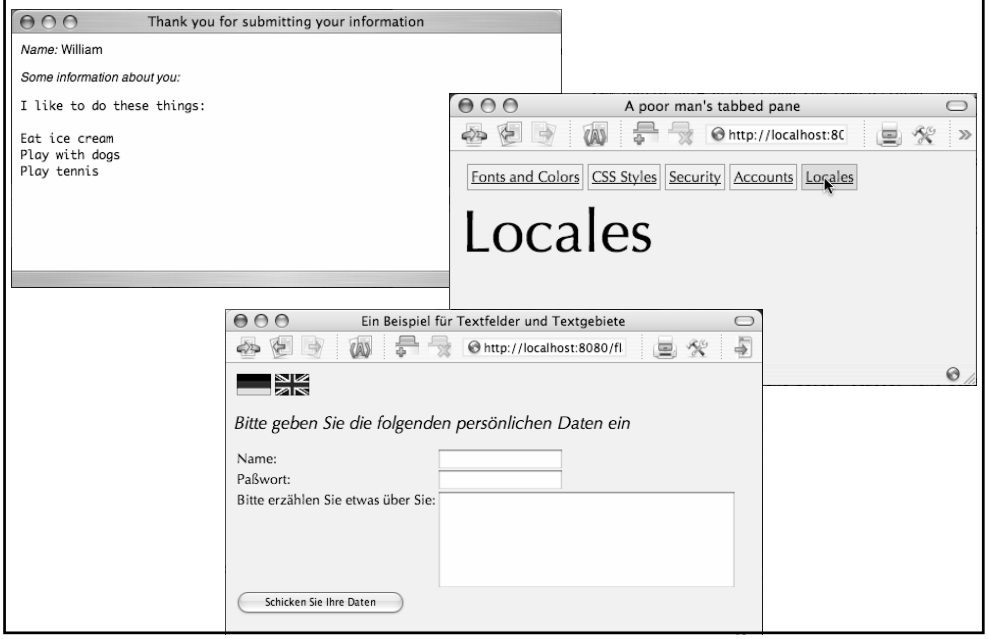
No Fluff, Just Stuff
February 20, 2004

David Geary
www.corejsf.com
www.core-jstl.com
sabreware@earthlink.net

David Geary

- 20 yrs OO development; Sun from 1994-1997
- Member of JSF and JSTL Expert Groups
- Author of 6 Java books; coming soon: Core JavaServer Faces
- 2nd Struts committer; Designed and implemented the Struts Template library
- Wrote test questions for the Web Developer Certification Exam
- JavaWorld Columnist: Java Design Patterns

Demos



This session

- What is JavaServer Faces (JSF)?
- JSF vs. Struts
- Using JSF Tag Libraries
- Converters and Validators
- Wiring JSF HTML tags to Java code
- The JSF lifecycle and event handling

Next...

- Introduction
- Input demo
- JSF Tag Libraries
- Hello Willis
- Tabbed pane demo
- Converters and Validators
- The JSF lifecycle and event model
- Flags demo

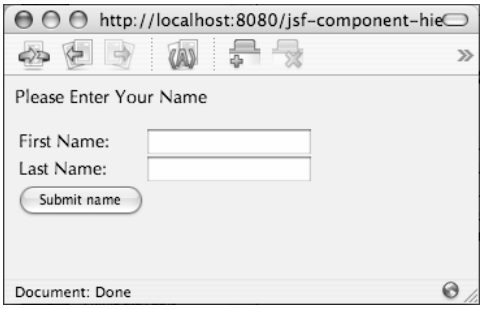
A brief history of server-side Java

- 1999: Servlets
- JavaServer Pages (JSP)
- Struts
- The JSP Standard Tag Library (JSTL)
- JavaServer Faces

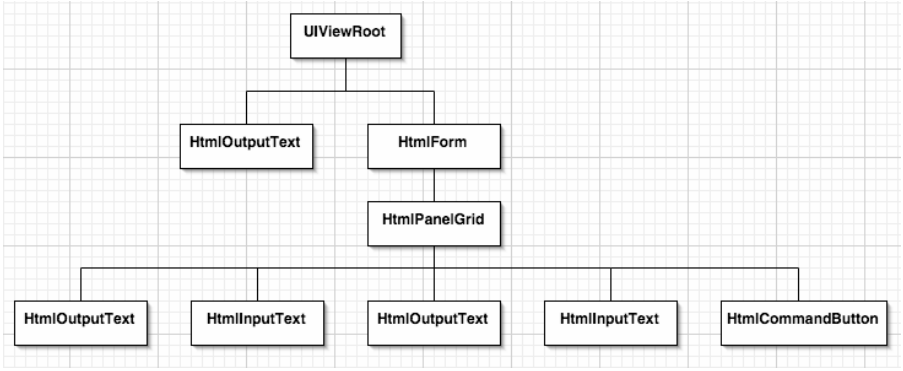
What is JSF?

- A web application framework (like Struts)
- Controller servlet, actions, beans
- Server-side components (like Swing)
- Components, layout managers, event model
- JSP-based; rendering technology independent
- Core and HTML JSP tags
- A standard

A simple form



The component hierarchy



Validation and State

The JSF page

```
<h:output_text value='#{msgs.firstNamePrompt}'/>
<h:input_text  value='#{name.first}'
                    required='true'
                        id='firstName'>

   <f:validate_length minimum='5'/>
</h:input_text>

<h:message for='firstName'/>
```

State Saving

- Happens at </view>
- Server-side: Component hierarchy stored in session scope
- Client-side:
- Components are serialized
- Serialized components are encoded in a hidden field
- State saving is pluggable

JSF Main Players

- JSP custom tags
- Components (independent of rendering)
- Converters (String==>Object; Object==>String)
- Validators (length, double-range, long-range...)
- Facets
- Managed beans

JSF Main Players (cont)

- Event handlers (action and value changed listeners)
- Actions (insert business logic here)
- Renderers (encode component markup here)
- Render kits (markup-specific groups of renderers)
- Pluggable subsystems
- Navigation and view handlers, resolvers, listeners

JSF Plugins

- Default action listener
- Navigation handler
- Variable resolver
- Property resolver
- View Handler

JSF and Struts

- Is JSF a replacement for Struts? Yes!
- Struts is more reliable
- Struts has better tool support

- Validator, WebDoclet, Tiles, StrutsTestCase
- Struts has better documentation (books and articles)

JSF and Struts (cont.)

- JSF is a "better Struts"
- JSF has server-side components
- JSF will be the standard web application framework
- You can use Struts and JSF together

JSF Status

- 1.0 Beta released in December, 2003
- 1.0 FCS, Q1CY2004

JSF Resources

- http://java.sun.com/j2ee/javaserverfaces
- http://www.corejsf and http://www.core-jstl.com
- http://www.theserverside.com/resources/JSFInActionReview.jsp

More Resources

- http://www.jsfcentral.com
- http://forum.java.sun.com/forum.jsp?forum=427
- http://www.javaworld.com/javaworld/jw-11-2002/jw-1129-jsf_p.html
- http://www.javaworld.com/javaworld/jw-12-2002/jw-1227-jsf2_p.html

Next...

- Introduction
- Input demo
- JSF Tag Libraries
- Hello Willis
- Tabbed pane demo
- Converters and Validators
- The JSF lifecycle and event model
- Flags demo

Core Tag Library

- <%@ taglib uri='http://java.sun.com/jsf/core' prefix='f' %>
- <f:use_faces>
- <f:action_listener>, <f:valuechanged_listener>
- <f:attribute>, <f:parameter>
- <f:facet name='header'>...</f:facet>
- <f:validator>, <f:validate_length>, ... (more validators)

HTML Tag Library

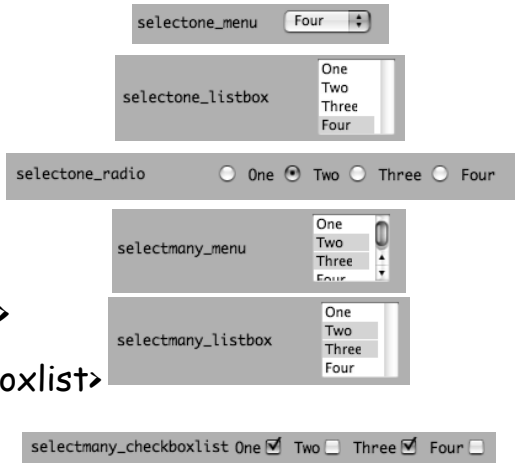
- <%@ taglib uri='http://java.sun.com/jsf/html' prefix='h' %>
- <h:form formName='registerForm'>...</h:form>
- <h:panel_grid columns='3' panelClass='table'/>
- <h:panel_group>...</h:panel_group>

HTML Tag Library (cont.)

- <h:output_text value='Welcome'/>
- <h:input_text valueRef='someBean.aProperty'/>
- <h:command_button actionRef='form.doIt'/>
- <h:command_hyperlink actionRef='form.doIt'/>

HTML Tag Library (cont.)

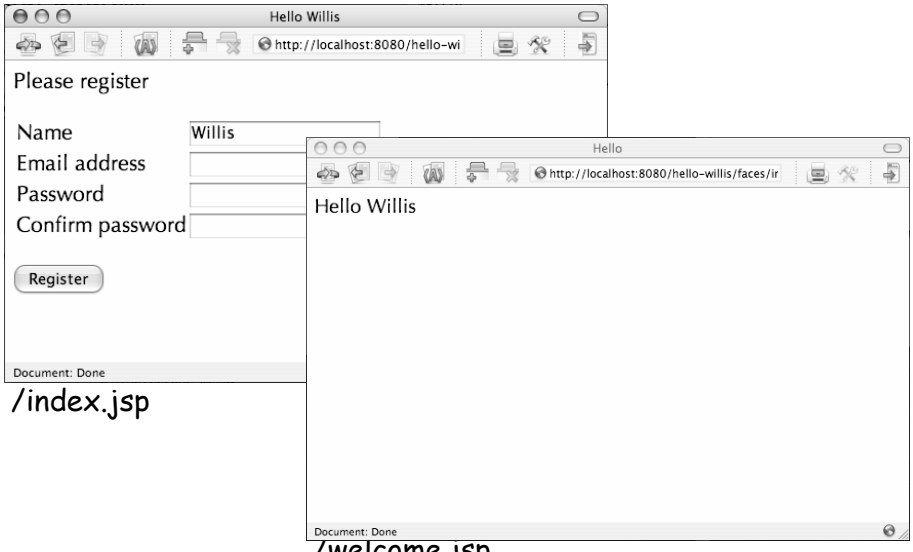
- <selectone_menu>
- <selectone_listbox>
- <selectone_radio>
- <selectmany_menu>
- <selectmany_listbox>
- <selectmany_checkboxlist>



Next...

- Introduction
- Input demo
- JSF Tag Libraries
- Hello Willis
- Tabbed pane demo
- Converters and Validators
- The JSF lifecycle and event model
- Flags demo

Hello Willis



/index.jsp

/welcome.jsp

The bean class

```
public class RegisterForm {
   private String name, email, password, passwordConfirm;
   public void setName(String name)     { this.name = name; }
   public String getName()              { return name; }
   public String getPassword()          { return password; }
   public void setPassword(String password)  {
      this.password = password;
   }
   public String getPasswordConfirm() { return passwordConfirm; }
   public void setPasswordConfirm(String passwordConfirm) {
     this.passwordConfirm = passwordConfirm;
   }
   public void setEmail(String email) { this.email = email; }
   public String getEmail() { return email; }


}
```

The form

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

<f:view locale='en-US'>
  <f:loadBundle basename='messages' var='msgs'/>
  ...
  <h:form>
    <h:panel_grid columns='2'>

      <h:output_text value='#{msgs.namePrompt}'/>
      <h:input_text   value='#{registerForm.name}'/>
      ...

    </h:panel_grid>
    <h:command_button value='#{msgs.submitPrompt}'/>
  </h:form>
</f:view>
```

The properties file

namePrompt=Name
emailAddressPrompt=Email address
passwordPrompt=Password
passwordConfirmPrompt=Confirm password

submitPrompt=Register

welcomeWindowTitle=Welcome
welcomePageTitle=Welcome {0}

The managed bean

```
<faces-config>
  ...
  <managed-bean>

    <managed-bean-name>user</managed-bean-name>
    <managed-bean-class>beans.User</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>

    <managed-property>
      <property-name>name</property-name>
      <value>Willis</value>
    </managed-property>
  </managed-bean>
  ...
</faces-config>
```

Navigation

```
<faces-config>
  ...
  <navigation-rule>

    <from-view-id>/index.jsp</from-view-id>
    <navigation-case>
      <to-view-id>/welcome.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>
  ...
</faces-config>
```

The welcome page

```
<f:loadBundle basename='messages' var='msgs'/>
...
<f:view locale='en-US'>
  <h:output_message value='#{msgs.welcomePageTitle}'/>
    <f:parameter value='#{registerForm.name}'/>
  </h:output_message>
</f:view>
```

```
welcomePageTitle=Welcome {0}
```

Adding an action

```
<f:command_button value='#{msgs.submitPrompt}'/>
                  action='#{registerForm.action}'/>
```

```
public class RegisterForm {
  ...
  public String action() {
    // perform business logic
    if(someCondition)
      return "success";
    else
      return "failure";
  }
}
```

```
<managed-bean>
  <managed-bean-name>registerForm</managed-bean-name>
  <managed-bean-class>beans.RegisterForm</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

Navigation, redux

```
<faces-config>
  <navigation-rule>
    <from-view-id>/index.jsp</from-view-id>
    <navigation-case>
      <from-outcome>success</from-outcome>
      <to-view-id>/welcome.jsp</to-view-id>
    </navigation-case>

    <navigation-case>
      <from-outcome>failure</from-outcome>
      <to-view-id>/error.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>
</faces-config>
```

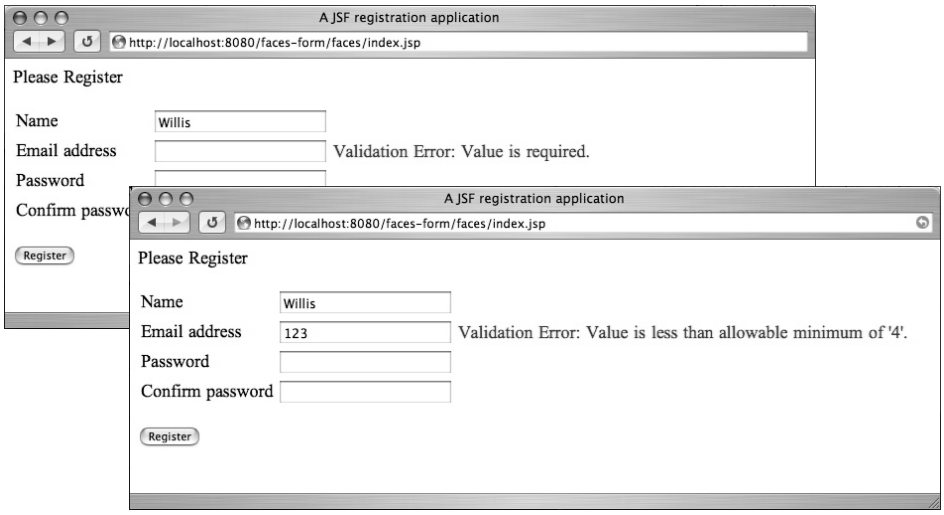
Next...

- Introduction
- Input demo
- JSF Tag Libraries
- Hello Willis
- Tabbed pane demo
- Converters and Validators
- The JSF lifecycle and event model
- Flags demo

Next...

- Introduction
- Input demo
- JSF Tag Libraries
- Hello Willis
- Tabbed pane demo
- Converters and Validators
- The JSF lifecycle and event model
- Flags demo

Validators



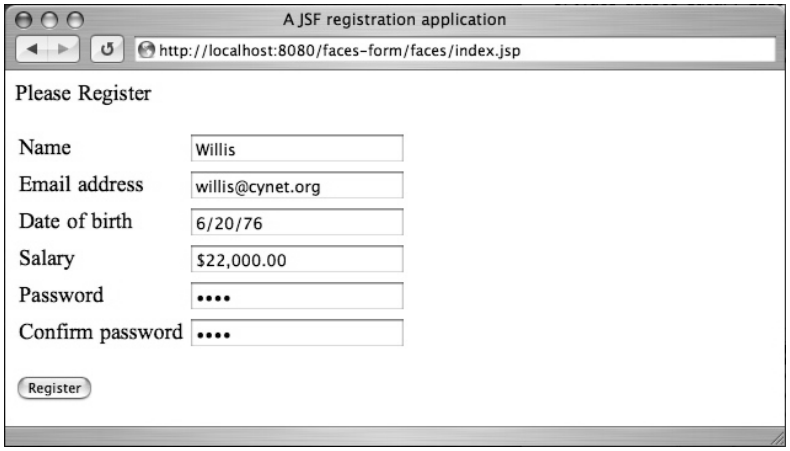
Using standard validators

```
<h:output_text value='#{msgs.emailAddressPrompt}'/>

<h:panel_group>

  <h:input_text id='email'
                value='#{registerForm.emailAddress}'
          required='true'>

        <f:validate_length minimum='4'/>

   </h:input_text>
</h:panel_group>
<h:output_errors for='email'/>
```

Standard validators

- required tag attribute
- <f:validate_length minimum='2' maximum='4'/>
- <f:validate_doublerange minimum='1.25/>
- <f:validate_longrange minimum='10000' maximum='100000'/>

Converters



The bean

```
public class User {
   private Date dateOfBirth = new Date("06/20/77");
   private double salary = 22000;
   ...
  public Date getDateOfBirth() { return date; }
  public void setDateOfBirth(Date date) {
     this.date = date;
  }
  public double getSalary() { return salary; }
  public void setSalary(double salary) {
     this. salary = salary;
  }
  ...
}
```

The converters

```
<h:input_text id='dateOfBirth'
              value='#{registerForm.dateOfBirth}'
       converter='DateTime'>
  <f:attribute name='dateStyle' value='short'/>
</h:input_text>


<h:input_text id='salary'
              value='#{registerForm.salary}'
       converter='Number'>
  <f:attribute name='numberStyle' value='currency'/>
</h:input_text>
```

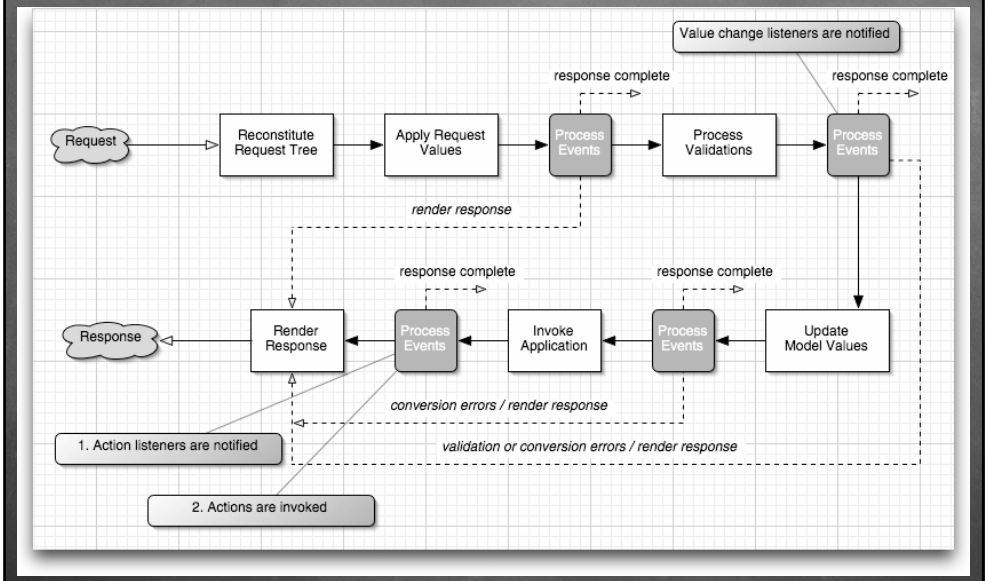
Standard converters

- Boolean
- Number (numberStyle)
- NumberFormat (formatPattern)
- Date (dateStyle, timezone)
- Time (timeStyle, timezone)
- DateTime (dateStyle, timeStyle, timezone)
- DateFormat (formatPattern, timezone)

Next...

- Introduction
- Input demo
- JSF Tag Libraries
- Hello Willis
- Tabbed pane demo
- Converters and Validators
- The JSF lifecycle and event model
- Flags demo

The JSF Lifecycle: RAP UI



The JSF event model

- Events wire UI logic to command and input components
- Two types of events: action and value changed
- Action events are fired by UICommands (buttons and hyperlinks)
- Value changed events are fired by UIInputs (textfields, textareas, listboxes, etc.)

Value change events

The JSP page

```
<h:form>
   <h:panel_grid columns='2'>
     ...
     <h:output_text id='cityPrompt'
                    value='#{msgs.cityPrompt}'/>
     ...
     <h:output_text id='statePrompt'
                    value='#{msgs.statePrompt}'/>
     ...
    <h:command_button value='#{msgs.submitPrompt}'/>
  </h:panel_grid>
</h:form>
```

The JSP page (cont)

```
   ...
   <h:output_text value='#{msgs.countryPrompt}'/>

    ...
    <h:selectone_menu onchange='submit()'
                      value='#{registerForm.country}'
        valueChangeListener='#{registerForm.countryChanged}'>

      <f:selectitems value='#{registerForm.countryNames}'/>

    </h:selectone_menu>
    <h:command_button value='#{msgs.submitPrompt}'/>
  </h:panel_grid>
</h:form>
```

The country names

```
public class RegisterForm {
   private static ArrayList countryItems = null;
   private static final String[] COUNTRY_NAMES =
                                        { "United States", "Canada" };
   public Collection getCountryNames() {
        if(countryItems == null) {
          ...
        }
        return countryNames();
   }
   ...

}
```

The country names (cont)

```
...
if(countryItems == null) {
  countryItems = new ArrayList();
  for(int i=0; i < COUNTRY_NAMES.length; ++i) {
     countryItems.add(new
            SelectItem(COUNTRY_NAMES[i],    // value
                       COUNTRY_NAMES[i],   // display
                       COUNTRY_NAMES[i])); // descrip
   }

 }
 return countryItems;
}
```

Making the switch

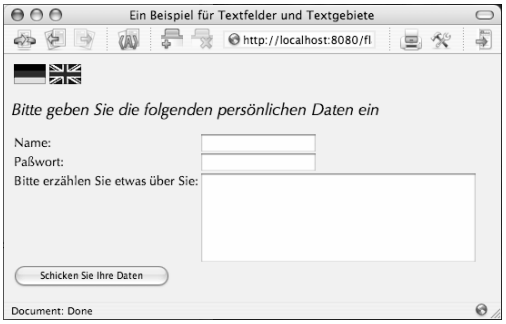
```
public class RegisterForm {
  ...
  public void countryChanged(ValueChangedEvent event) {
    UIOutput cityPrompt  = ((UIOutput)event.getComponent()).
                           findComponent("cityPrompt");
    UIOutput statePrompt  = ((UIOutput)event.getComponent()).
                           findComponent("statePrompt");
    if("United States".equals(event.getNewValue())) {
      cityPrompt.setValueBinding ("value",
                app.createValueBinding("#{msgs.cityPrompt}");
      statePrompt.setValueBinding ("value",
                app.createValueBinding("#{msgs.statePrompt}");
    }
    ...
```

Making the switch (cont)

```
      ...
      else {
        cityPrompt.setValueBinding ("value",
             app.createValueBinding("#{msgs.municipalityPrompt}");
        statePrompt.setValueBinding ("value",
      }       app.createValueBinding("#{msgs.provincePrompt}");

  }
  ...
}
```

Next...

- Introduction
- Input demo
- JSF Tag Libraries
- Hello Willis
- Tabbed pane demo
- Converters and Validators
- The JSF lifecycle and event model
- Flags demo



Thanks for coming

- What is JavaServer Faces (JSF)?
- JSF vs. Struts
- Using JSF Tag Libraries
- Converters and Validators
- Wiring JSF HTML tags to Java code
- The JSF lifecycle and event handling

Advanced JSF

- Data Table
- Custom Components and Renderers
- Immediate Events
- Actions and Action Listeners
- Custom converters and validators
- Implementing custom components