

Apache Struts: Prepopulating and Redisplaying Input Forms

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet/JSP/Struts/JSP Training: courses.coreservlets.com

Slides © Marty Hall, <http://www.coreservlets.com>, books © Sun Microsystems Press

Agenda

- **Using the Struts `html:form` tags to build HTML forms that have two important characteristics:**
 - They can be prepopulated based on the values in a `JavaBean`.
 - That is, the initial values of the form elements can be taken from a `Java` object.
 - They can be redisplayed when they are submitted with incomplete or incorrect values.
 - Specifically, when they are redisplayed, they can maintain the values that the end user already entered.

www.coreservlets.com

Summary of Techniques Introduced in This Lecture

- **Add an input attribute to the action element in `struts-config.xml`.**
 - This attribute tells the system to associate a form-bean of the type designated by name (which should match a name in the form-bean element) with the input form.
- **Use the appropriate scope in the action declaration.**
 - Use `scope="request"` if you want to prepopulate a form only (i.e., display initial values).
 - Use `scope="session"` if also you want to redisplay the form with previous values intact (i.e., redisplay an incomplete form without making the user reenter values they already entered).
 - Default scope is `session` (surprisingly), but always list it explicitly.

www.coreservlets.com

Summary of Techniques Introduced in This Lecture (Cont.)

- **Use `html:form` to declare the input form in the initial JSP page.**
 - The address of this input page must match the input attribute of the action element from the previous step.
- **Using the Struts `html:form` element instead of the standard HTML FORM element yields four results:**
 - A bean is associated with the form.
 - A bean of the type specified by form-bean is automatically used.
 - The Web application prefix is prepended automatically.
 - You say `<html:form action="/actions/...">` to get `<FORM ACTION="/webAppPrefix/actions/..." ...>`.
 - The `.do` suffix is appended automatically.
 - You say `<html:form action="/actions/blah">` to get `<FORM ACTION="/webAppPrefix/actions/blah.do" ...>`.
 - POST, not GET, is the default METHOD.
 - You say `<html:form action="/actions/blah">` to get `<FORM ACTION="/webAppPrefix/actions/blah.do" METHOD="POST">`.

www.coreservlets.com

Summary of Techniques Introduced in This Lecture (Cont.)

- **Use `html:text` and similar elements to declare the input fields of the form.**
 - The NAME of each input field is taken from the bean property name, and the VALUE is taken from the bean property value. For example, using `<html:text property="firstName"/>` is equivalent to first declaring a bean of the appropriate type, then doing `<INPUT TYPE="TEXT" NAME="firstName" VALUE="<%= theBean.getFirstName() %>">`.
 - Not only does this provide initial values for your formfields, but it also makes it easier for you to be sure that the field names match the bean property names. Since, in the execute method of the Action object, the form-bean is filled in by matching up request parameter names with bean property names, it is critical that the names stay in synch.

www.coreservlets.com

Struts Flow of Control

- **A form submits data to a URL of the form `blah.do`. The new wrinkle in this lecture is that this form uses the `html:form` tag to associate a bean with the form to give the form initial values.**
 - That `blah.do` address is mapped by `struts-config.xml` to an Action object, whose execute method handles the request.
 - One of the arguments to execute is a form bean that is automatically created and whose properties are automatically populated with the incoming form data.
 - The Action object then invokes business logic and data-access logic, placing the results in normal beans stored in request, session, or application scope.
 - The Action uses `mapping.findForward` to return a condition, and the conditions are mapped by `struts-config.xml` to various JSP pages.
 - The system forwards the request to the appropriate JSP page, where the `bean:write` tag is used to output properties of the form bean and results beans.
 - Optionally, both the input form and the final JSP pages can use `bean:message` to output standard messages and labels as defined in a system properties file.

www.coreservlets.com

The Six Basic Steps in Using Struts: Updates

- **Modify struts-config.xml**
 - We still use the action and forward elements to specify the Action object and destination URLs, and we still use the form-bean element to declare form beans.
 - However, in addition to the path, type, name and scope attributes of the action element, we add a fifth attribute: input. This attribute tells the system to associate a form-bean of the type designated by name (which should match a name in the form-bean element) with the input form.
- **Define a form bean.**
 - This bean will be defined in the same way as before. However, in addition to being used in the execute method of the Action and (optionally) in the final JSP pages, the bean will also be used in the initial input form to give names and values to the various input elements.

The Six Basic Steps in Using Struts: Updates

- **Create results beans.**
 - These are normal beans of the sort used in MVC when implemented directly with RequestDispatcher, and are created and used in the same way as described in the previous lecture.
- **Create an Action object to handle requests.**
 - As in the previous lecture, rather than calling request.getParameter explicitly, the execute method casts the ActionForm argument to the specific form bean class, then uses getter methods to access the properties of the object.

The Six Basic Steps in Using Struts: Updates

- **Create form that invokes blah.do.**
 - Rather than using the standard HTML FORM and INPUT tags, we now use html:form and html:text (and related tags) to build the input form.
 - The html:form tag automatically associates a bean with the form, and html:text automatically uses the bean property names for each textfield NAME and the bean property values for each textfield VALUE.
 - In addition, as in the previous lecture, this form can still use the bean:message tag to output standard messages and text labels.
- **Display results in JSP.**
 - As before, the JSP page uses the bean:write tag to output properties of the form and result beans.
 - It may also use the bean:message tag to output standard messages and text labels that are defined in a properties file.

Example 1: Prepopulating Forms

- **In many cases, you want the initial form to be based on data defined in your application. If the application data changes, you want the initial values of the form fields to change automatically. To implement this behavior:**
 - In struts-config.xml, your action element should have an input attribute listing the relative address of the JSP page containing the input form.
 - In struts-config.xml, your action element should specify scope="request".
 - In the input form, you should use html:form, and should specify action="/path/blah", not action="/webAppPrefix/path/blah.do". Also, recall that POST is the default method for html:form.
 - In your input form, you should use <html:text property="propertyName"/>. to declare input textfields. Each textfield NAME will be taken from the bean property name, and each textfield VALUE will be taken from the bean property value.
 - In your input form, use html:button, html:checkbox, html:textarea, etc., to declare submit buttons, checkboxes, text areas, etc.

Step 1 (Modify struts-config.xml)

- **Designate Action classes to handle requests for blah.do.**
 - As before, we use the action element.
- **Specify the URLs that apply in various situations.**
 - As before, we use multiple forward elements within the action element.
- **Declare any form beans that are being used.**
 - As before, we use name and type attributes in the form-bean declaration.
 - As before, we also add name (the bean name as given in form-bean) and scope (request since we aren't redisplaying the form) attributes to the action declaration.
 - Finally, to tie the bean to the input form, we add one new attribute of action: input. The input attribute specifies the relative address of the JSP page that contains the input form.

Step 1 (Modify struts-config.xml) – Example of input Element

```
<action path="/actions/signup1"
        type="coreservlets.SignupAction1"
        name="contactFormBean"
        scope="request"
        input="/forms/signup1.jsp">
    ...
</action>
```

Step 1 (Modify struts-config.xml) – Final Code

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC ... >
<struts-config>
  <form-beans>
    <form-bean name="contactFormBean"
              type="coreservlets.ContactFormBean"/>
  </form-beans>
  <action-mappings>
    <action path="/actions/signup1"
           type="coreservlets.SignupAction1"
           name="contactFormBean"
           scope="request"
           input="/forms/signup1.jsp">
      <forward name="missing-value"
              path="/WEB-INF/results/missing-value.jsp"/>
      <forward name="success"
              path="/WEB-INF/results/confirmation.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

www.coreservlets.com

Step 2 (Define a Form Bean)

- As before, the form bean will be automatically filled in with the incoming form parameters and passed to the execute method.
- In addition, a new instance of the form bean will be created and used to fill in the fields of the input form, with the NAME of the input field coming from the bean property name and the VALUE coming from the bean property value.
 - For example, the following slide shows a form bean corresponding to contact information for a person that will be added to an email/fax list in our application. It contains properties for first name, last name, email address, etc.

Apache Struts: Handling Forms

www.coreservlets.com

Step 2 (Define a Form Bean) – Code for ContactFormBean

```
package coreservlets;
import org.apache.struts.action.*;

public class ContactFormBean extends ActionForm {
  private String firstName = "First name";
  private String lastName = "Last name";
  private String email = "user@host";
  private String faxNumber = "xxx-yyy-zzzz";

  public String getFirstName() {
    return(firstName);
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  ... // lastName, emailAddress
}
```

Apache Struts: Handling Forms

www.coreservlets.com

Step 3 (Create Results Beans)

- The form bean represents the input data: the data that came from the HTML form.
- The results beans represent the output data: the data created by the business logic to represent the results of the computation or database lookup.
- The next slide shows a bean (MessageBean) that will be used to store simple error messages.

Apache Struts: Handling Forms

www.coreservlets.com

Step 3 (Create Results Beans) – Code for MessageBean

```
package coreservlets;

public class MessageBean {
  private String message = "";

  public String getMessage() {
    return(message);
  }

  public void setMessage(String message) {
    this.message = message;
  }
}
```

Apache Struts: Handling Forms

www.coreservlets.com

Step 4 (Create an Action Object to Handle Requests)

- In this example, we check each of the required input parameters (first name, last name, email address, and fax number) to see if it is missing.
 - If it is missing, we use mapping.findForward to return a "missing-value" condition, which is mapped by struts-config.xml to an error page that contains a message saying which parameter was missing.
 - If all parameters are present, we return "success", which results in an order-confirmation page.
- This example is similar to the previous one.
 - Again, we do not call request.getParameter explicitly, but instead extract the request parameters from the already populated form bean. Specifically, we take the ActionForm argument supplied to execute, cast it to ContactFormBean (our concrete class that extends ActionForm), and then call getter methods on that bean.
 - Also, we pass the error message to the JSP page by creating a MessageBean and storing it in request scope.

Apache Struts: Handling Forms

www.coreservlets.com

Step 4 (Create an Action Object to Handle Requests) -- Code

```
public class SignupAction1 extends Action {
    public ActionForward
        execute(ActionMapping mapping,
                ActionForm form,
                HttpServletRequest request,
                HttpServletResponse response)
        throws Exception {
        ContactFormBean userBean = (ContactFormBean)form;
        String firstName = userBean.getFirstName();
        String lastName = userBean.getLastName();
        String email = userBean.getEmail();
        String faxNumber = userBean.getFaxNumber();
        MessageBean messageBean = new MessageBean();
        request.setAttribute("messageBean", messageBean);
        if (isMissing(firstName)) {
            makeWarning(request, "first name");
            return(mapping.findForward("missing-value"));
        } else ...
    }
}
```

16

Apache Struts: Handling Forms

www.coreservlets.com

Step 5 (Create Form that Invokes blah.do)

- This form is very different from the ones in the previous examples.

– Instead of using the standard HTML FORM tag, we import and use the `html:form` tag. The Web application prefix, the `.do` suffix, and the POST method are all generated automatically. So, we use

```
<%@ taglib uri="/WEB-INF/struts-html.tld"
    prefix="html" %>
<html:form action="/actions/signup1">
```

```
...
</html:form>
```

to get something equivalent to:

```
<FORM ACTION="/signup/actions/signup1.do"
    METHOD="POST">
```

```
...
</FORM>
```

20

Apache Struts: Handling Forms

www.coreservlets.com

Step 5 (Create Form that Invokes blah.do), Continued

- Since the location of the input form matches the input attribute of the action declaration in `struts-config.xml`, a bean is automatically created when the input page is accessed.
 - The type of bean created is determined by looking at the name that goes with the input attribute, and finding the form-bean with the same name.
 - In this case, this process results in a `ContactFormBean` being created.
- After declaring the form and associating it with a bean, we use `html:text` to build input elements whose NAME and VALUE are taken from the form bean property names and values.

21

Apache Struts: Handling Forms

www.coreservlets.com

Step 5 (Create Form that Invokes blah.do), `html:text` Details

- The following

```
First name: <html:text property="firstName"/>
```

results in something similar to:

```
<% coreservlets.ContactFormBean contactBean =
    new coreservlets.ContactFormBean(); %>
```

```
First name:
```

```
<INPUT TYPE="TEXT" NAME="firstName"
    VALUE="<%= contactBean.getFirstName() %>">
```

22

Apache Struts: Handling Forms

www.coreservlets.com

Step 5 (Create Form that Invokes blah.do) – Final Code

```
... the
Single Provider of Alert Memos
system lets you sign up for them all in one easy
request!
<P>
<CENTER>
<%@ taglib uri="/WEB-INF/struts-html.tld"
    prefix="html" %>
<html:form action="/actions/signup1">
    First name: <html:text property="firstName"/><BR>
    Last name: <html:text property="lastName"/><BR>
    Email address: <html:text property="email"/><BR>
    Fax number: <html:text property="faxNumber"/><BR>
    <html:submit value="Sign Me Up!"/>
</html:form>
</CENTER>
</BODY></HTML>
```

23

Apache Struts: Handling Forms

www.coreservlets.com

Step 6 (Display Results in JSP)

- In this example, there are two possible JSP pages:
 - One for missing input
 - One for success.
- We use the `MessageBean` to customize the error messages in the page that is used for missing input.
 - That way, there does not need to be a separate page for each type of error.
- As in the previous example, the `bean:write` tag is used to output bean properties without having to resort to explicit Java code.

24

Apache Struts: Handling Forms

www.coreservlets.com

Step 6 (Display Results in JSP) -- First Possible Page

```

<!DOCTYPE ...>
<HTML>
<%@ taglib uri="/WEB-INF/struts-bean.tld"
    prefix="bean" %>
<HEAD><TITLE>Missing
<bean:write name="messageBean" property="message"/>
</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H2>Missing
<bean:write name="messageBean" property="message"/>!
</H2>
Please <A HREF="../forms/signup1.jsp">try again</A>.
</CENTER>
</BODY></HTML>
    
```

26 Apache Struts: Message Forms www.coreservlets.com

Step 6 (Display Results in JSP) -- Second Possible Page


```

...<H1>Confirmation</H1>
Congratulations. You are now signed up for the
Single Provider of Alert Memos network!
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<UL>
<LI>First name:
<bean:write name="contactFormBean" property="firstName"/>
<LI>Last name:
<bean:write name="contactFormBean" property="lastName"/>
<LI>Email address:
<bean:write name="contactFormBean" property="email"/>
<LI>Fax number:
<bean:write name="contactFormBean" property="faxNumber"/>
</UL>
To be removed from the network, send email
<A HREF="mailto:blackhole@spam-network.com">here</A>.
</CENTER>
</BODY></HTML>
    
```

26 Apache Struts: Message Forms www.coreservlets.com

Example 1: Results

- First, the HTML form is invoked with the URL <http://localhost/signup/forms/signup1.jsp>.
 - Notice that the textfields are prepopulated with the values of the ContactFormBean bean properties.




27 Apache Struts: Message Forms www.coreservlets.com

Example 1: Results

- The action in the `html:form` element is `/actions/signup1`, and this results in the URL <http://localhost/signup/actions/signup1.do>.
 - This address is mapped by `struts-config.xml` to the `SignupAction1` class, whose `execute` method is invoked.
 - This method examines the first name, last name, email address, and fax number to see if any are missing. If so, it returns `mapping.findForward` with a value of "missing-value".
 - That value is mapped by `struts-config.xml` to `/WEB-INF/results/missing-value.jsp`, which is the final result displayed to the user for missing input.
 - However, since the `MessageBean` stores the specific error message, the error page is different in each of the cases. For example, the following screen shots show the results when the form is submitted with a missing first and last name, respectively.

28 Apache Struts: Message Forms www.coreservlets.com

Example 1: Results



29 Apache Struts: Message Forms www.coreservlets.com

Example 1: Results

- In the next case, the form is submitted with all of the required parameters.
- The action in the `html:form` element is `/actions/signup1`, and this results in the URL <http://localhost/signup/actions/signup1.do>.
 - This address is mapped by `struts-config.xml` to the `SignupAction1` class, whose `execute` method is invoked.
 - This method examines the first name, last name, email address, and fax number to see if any are missing. Since none are, it returns `mapping.findForward` with a value of "success".
 - That value is mapped by `struts-config.xml` to `/WEB-INF/results/confirmation.jsp`, which is the final result displayed to the user.

30 Apache Struts: Message Forms www.coreservlets.com

Example 1: Results

here.'"/>

Example 2: Redisplaying Forms

- In the previous example, we show how to display a form whose **initial values** come from a newly-created (i.e., request-scoped) JavaBeans component.
- In this example, we show how to **redisplay** a form and base its values on a **session-scoped bean**.
 - This second technique lets you submit a form, check if all the required data is supplied, and redisplay the form if any of the data is missing.
 - Most importantly, when you redisplay the form, you can maintain the previous values that the end user entered.
 - To implement this behavior, you declare a form-bean in struts-config.xml as before, use the input attribute of action to designate the URL of the input page (so that a bean can be created) as before, and again use the html:form and html:text tags in the input form.
 - However, there are three differences, as listed on the next slides.

Redisplaying Forms: First Task

- In struts-config.xml, your action element should specify **scope="session"** instead of **scope="request"**.
 - The reason that the bean must be stored in the HttpSession object is so that it can be filled in by the Action when the form is submitted, yet still be accessed later when the input form is redisplayed. Here is an example:


```
<action path="/actions/signup2"
          type="coreservlets.SignupAction2"
          name="contactFormBean"
          scope="session"
          input="/forms/signup2.jsp">
...
</action>
```

Redisplaying Forms: Second Task

- In struts-config.xml, the forward entry corresponding to missing data should supply the address of the input form for the path, rather than supplying the address of a new JSP page as in the previous examples.
 - It is also common to use **redirect="true"** so that the system uses `response.sendRedirect` instead of `RequestDispatcher.forward` to transfer to the input form. Using a redirect is slightly slower since it involves a roundtrip network connection, but it exposes the URL of the input form to the users, which is more familiar since they use that URL when they originally access the form. Here is an example:


```
<forward name="missing-value"
          path="/forms/signup2.jsp"
          redirect="true"/>
```

Redisplaying Forms: Third Task

- When redisplaying your input form, you should display an error message that tells the user which data they failed to supply.
 - This message can be stored in a separate session-scoped bean, or, more simply, you can use a special bean property of the form bean to store error messages. By making the default value of this bean property be an empty string, you can avoid logic that requires you to distinguish the initial display of the form from a redisplay. Eg:


```
public class ContactFormBean extends ActionForm {
  private String warning = "";
  ...
  public String getWarning() { return(warning); }

  public void setWarning(String baseWarning) {
    this.warning =
      "<H2><FONT COLOR=RED>Missing " +
      baseWarning + "!</FONT></H2>"; ...www.coreservlets.com
```

Redisplaying Forms: Third Task, Continued

- Here is an example of how to output the error message.
 - Notice that we use **filter="false"** because the error message contains HTML tags.


```
<bean:write name="contactFormBean"
            property="warning"
            filter="false"/>
```

Step 1 (Modify struts-config.xml)

- Designate Action classes to handle requests for *blah.do*.
 - As before, we use the action element.
- Specify the URLs that apply in various situations.
 - As before, we use multiple forward elements within the action element. However, for the "missing-value" entry, we specify the location of the original input form, rather than the location of a new JSP page. We also use `redirect="true"` so that the system uses an HTTP redirect, not `RequestDispatcher.forward`. Eg:


```
<forward name="missing-value"
          path="/forms/signup2.jsp"
          redirect="true"/>
```
- Declare any form beans that are being used.
 - As before, we use name and type attributes in the form-bean declaration. As before, we also add name, input, and scope attributes to the action declaration. However, in this case we use `scope="session"` instead of `scope="request"` so that the values are available to the input-form-page when we redirect to it.

37

Apache Struts: Handling Forms

www.coreservlets.com

Step 1 (Modify struts-config.xml) -- Final Code

```
<struts-config>
  <form-beans>
    <form-bean name="contactFormBean"
              type="coreservlets.ContactFormBean"/>
  </form-beans>
  <action-mappings>
    ...
    <action path="/actions/signup2"
           type="coreservlets.SignupAction2"
           name="contactFormBean"
           scope="session"
           input="/forms/signup2.jsp">
      <forward name="missing-value"
              path="/forms/signup2.jsp"
              redirect="true"/>
      <forward name="success"
              path="/WEB-INF/results/confirmation.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

38

Apache Struts: Handling Forms

www.coreservlets.com

Step 2 (Define a Form Bean)

- This example uses the same `ContactFormBean` as the previous example.
 - However, in this example we make use of the `warning` property, which does not correspond to a request parameter but rather is used to send missing-entry-warnings to the input form.

```
package coreservlets;
import org.apache.struts.action.*;

public class ContactFormBean extends ActionForm {
  ...
  public String getWarning() { return(warning); }

  public void setWarning(String baseWarning) {
    this.warning =
      "<H2><FONT COLOR=RED>Missing " +
      baseWarning + "!</FONT></H2>";
  }
}
```

39

Apache Struts: Handling Forms

www.coreservlets.com

Step 3 (Create Results Beans)

- Since the confirmation page merely displays the data entered by the user (which is taken from the form bean), this example uses no results beans.

40

Apache Struts: Handling Forms

www.coreservlets.com

Step 4 (Create an Action Object to Handle Requests)

- This example is very similar to the previous one.
 - However, instead of storing missing-value warning messages in a separate bean that will be displayed in a separate JSP page, the warnings are stored in form-bean, which is used when the original input page is redisplayed.
 - To do this, we keep the *same* execute method, but replace the `showWarning` method with a version that stores the warnings in the form-bean (intended for the original input page) rather than in the `MessageBean` (intended for a custom error page).

41

Apache Struts: Handling Forms

www.coreservlets.com

Step 4 (Create an Action Object to Handle Requests) -- Code

```
package coreservlets;
import javax.servlet.http.*;

public class SignupAction2 extends SignupAction1 {
  protected void makeWarning(HttpServletRequest request,
                             String message) {
    HttpSession session = request.getSession();
    ContactFormBean contactFormBean =
      (ContactFormBean)session.getAttribute(
        "contactFormBean");
    contactFormBean.setWarning(message);
  }
}
```

42

Apache Struts: Handling Forms

www.coreservlets.com

Step 5 (Create Form that Invokes *blah.do*)

- **This form is very similar to the one in the previous example.**
 - Again, we use the `html:form` and `html:text` elements to build an HTML form whose formfield values are derived from bean properties.
 - However, this time the bean is session-scoped, so when the form is redisplayed the bean property values are the ones that the user previously entered.
 - Furthermore, we output a warning message that reminds the user which field they omitted. The default warning message is an empty string, so rather than testing to see if the form is being initially displayed or redisplayed, we always output the error message (with `filter="false"` because the error message can contain HTML tags).

44 Apache Struts: Handling Forms www.coreservlets.com

Step 5 (Create Form that Invokes *blah.do*) -- Code

```

...
<CENTER>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:form action="/actions/signup2">
  <bean:write name="contactFormBean" property="warning"
    filter="false"/>
  First name: <html:text property="firstName"/><BR>
  Last name: <html:text property="lastName"/><BR>
  Email address: <html:text property="email"/><BR>
  Fax number: <html:text property="faxNumber"/><BR>
  <html:submit value="Sign Me Up!"/>
</html:form>
</CENTER>
    
```

44 Apache Struts: Handling Forms www.coreservlets.com


Step 6 (Display Results in JSP)

- **In this example, there are two possible return values from the execute method of the Action object: "missing-value" and "success".**
 - Since the "missing-value" is mapped by `struts-config.xml` back to the original input page, there is only one real results page.
 - Since the results in this example are exactly the same as in the previous example, we use the *same* JSP page.

46 Apache Struts: Handling Forms www.coreservlets.com

Example 2: Results

- **First, the HTML form is invoked with the URL `http://localhost/signup/forms/signup2.jsp`.**
 - As before, the textfields are prepopulated with the values from the `ContactFormBean` properties



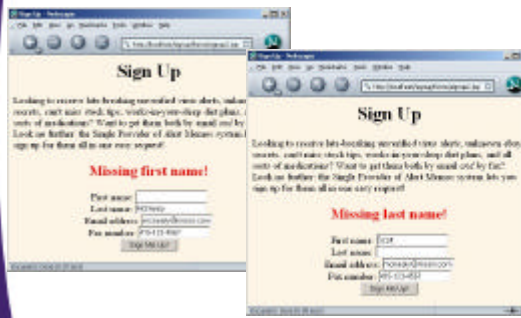
46 Apache Struts: Handling Forms www.coreservlets.com

Example 2: Results

- **The action in the `html:form` element is `/actions/signup2`, and this results in the URL `http://localhost/signup/actions/signup2.do`.**
 - This address is mapped by `struts-config.xml` to the `SignupAction2` class, whose `execute` method is invoked.
 - This method examines the first name, last name, email address, and fax number to see if any are missing. If so, it returns `mapping.findForward` with a value of "missing-value".
 - That value is mapped by `struts-config.xml` to the original input form. Since `redirect="true"` is used, transfer is by an HTTP redirect.
 - The `execute` method stores warnings about missing values in the warning property of the form bean (which defaults to an empty string), so notifying the user that they omitted a value is merely a matter of outputting that property.
 - For example, the following screen shots show the results when the form is submitted with a missing first and last name, respectively.

47 Apache Struts: Handling Forms www.coreservlets.com

Example 2: Results



48 Apache Struts: Handling Forms www.coreservlets.com

Example 2: Results

- In the next case, the form is submitted with all of the required parameters.
- The action in the `html:form` element is `/actions/signup2`, and this results in the URL `http://localhost/signup/actions/signup2.do`.
 - This address is mapped by `struts-config.xml` to the `SignupAction2` class, whose `execute` method is invoked.
 - This method examines the first name, last name, email address, and fax number to see if any are missing. Since none are, it returns mapping.`findForward` with a value of "success".
 - That value is mapped by `struts-config.xml` to `/WEB-INF/results/confirmation.jsp`, the *same* confirmation page as used in the previous example.

48 Apache Struts: Handling Forms www.coreservlets.com

Example 2: Results

49 Apache Struts: Handling Forms www.coreservlets.com

Other Capabilities of `html:` Library

- **`html:rewrite`**
 - Simplifies handling of relative URLs
 - You use URLs beginning with slashes, relative to Web-app home. System tacks Web app name on front.

```
<IMG SRC="<html:rewrite page='/images/pic.jpg' />"...>
```

 - See examples in section on Tiles
- **`html:base`**
 - Creates BASE definition so that relative URLs are with respect to real location of JSP page, not URL of Action.
 - Does not work when JSP pages are in WEB-INF, so of limited value
- **`html:javascript`**
 - Inserts JavaScript code for client-side validation
 - See examples in section on validation

51 Apache Struts: Handling Forms www.coreservlets.com

Summary

- **Add an input attribute to the action element in `struts-config.xml`.**
 - This attribute tells the system to associate a form-bean of the type designated by name (which should match a name in the form-bean element) with the input form.
- **Use the appropriate scope in the action declaration.**
 - Use request if you only want to prepopulate form
 - Use session if you also want to redisplay form
- **Use `html:form` and `html:text` to declare the input form in the initial JSP page**
 - Bean properties will automatically be used for the input field names and values

52 Apache Struts: Handling Forms www.coreservlets.com

Questions?

Core Servlets & JSP book: www.coreservlets.com
 More Servlets & JSP book: www.moreservlets.com
 Servlet/JSP/Struts Training Courses: courses.coreservlets.com

Slides © Marty Hall, <http://www.coreservlets.com>, books © Sun Microsystems Press