



Apache Struts: Handling Request Parameters with Form Beans

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

- Defining form beans
- Declaring form beans
- Outputting properties of form beans
- Defining and outputting regular beans
- Using properties files
- Using the JSP 2.0 expression language with Struts

www.coreservlets.com

Struts Flow of Control

- A form submits data to a URL of the form *blah.do*.
- That address is mapped by `struts-config.xml` to an Action object, whose `execute` method handles 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
 - 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 property file.

www.coreservlets.com

The Six Basic Steps in Using Struts: Updates for Bean Use

- Modify `struts-config.xml`.
 - In addition to the `action` and `forward` elements used to specify the Action object and destination URLs, we use the `form-bean` element to declare form beans.
 - We also add `name` and `scope` attributes to the action element to tie the form bean to the Action.
 - Optionally, we add a `message-resources` element to declare a properties file containing standard messages, names, and labels.
- Define a form bean.
 - This bean normally extends `ActionForm` and has properties (i.e., getter and setter methods) corresponding to each of the incoming request parameters.
 - Alternatively, the bean can extend `DynaActionForm`, in which case it will contain a `Map` representing the request parameters.

www.coreservlets.com

The Six Basic Steps in Using Struts: Updates for Bean Use

- Create results beans.
 - These are normal beans of the sort used in MVC when implemented directly with `RequestDispatcher`. That is, they represent the results of the business logic and data access code. These beans are stored in request, session, or application scope with the `setAttribute` method of `HttpServletRequest`, `HttpSession`, or `ServletContext`, just as in normal non-Struts applications.
- Create an Action object to handle requests.
 - Rather than calling `request.getParameter` explicitly as in the previous example, the `execute` method casts the `ActionForm` argument to the specific form bean class, then uses getter methods to access the properties of the object.

www.coreservlets.com

The Six Basic Steps in Using Struts: Updates for Bean Use

- Create form that invokes *blah.do*.
 - This form can use the `bean:message` tag to output standard messages and text labels that are defined in the properties file that is declared with `message-resources` in `struts-config.xml`.
- Display results in JSP.
 - 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 the standard properties file.

www.coreservlets.com

Example 1: Form and Result Beans

- **The URL** `http://hostname/struts-test/actions/register3.do` should be handled by the class `RegisterAction3`.
- **Instead of reading form data explicitly with `request.getParameter`, the `execute` method of `RegisterAction3` uses a bean that is automatically populated from the request data.**
 - As in the previous example, this method returns "success", "bad-address", or "bad-password", and these return values result in the JSP pages `/WEB-INF/results/result3.jsp`, `/WEB-INF/results/bad-address3.jsp`, and `/WEB-INF/results/bad-password3.jsp`, respectively.
 - However, instead of displaying static HTML as before, these pages display specific values that `RegisterAction3` creates and stores in beans.

7 Apache Struts Beans www.coreservlets.com

New Features of This Example

- **The use of a bean to represent the incoming form data.**
 - This bean extends the `ActionForm` class, is declared in `struts-config.xml` with the `form-bean` tag, and is referenced in `struts-config.xml` with `name` and `scope` attributes in the action element.
- **The use of a regular bean to represent custom results.**
 - As with beans used with `jsp:useBean`, this bean need not extend any particular class and requires no special `struts-config.xml` declarations.
- **The use of the Struts bean:write tags to output bean properties in the JSP page that displays the final results.**
 - This is basically a more powerful and concise alternative to the standard `jsp:getProperty` tag. Before we use `bean:write`, we have to import the "bean" tag library as follows.


```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
```

8 Apache Struts Beans www.coreservlets.com

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
- **Declare any form beans that are being used.**
 - First, we need a form-bean entry (within the form-beans entry) with the following two attributes:
 - **name:** a name that will match the name attribute of the action element.
 - **type:** the fully qualified classname of the bean.
 - Here is an example:


```
<form-beans>
  <form-bean name="userFormBean"
    type="coreservlets.UserFormBean"/>
</form-beans>
```

9 Apache Struts Beans www.coreservlets.com

Step 1 (Modify struts-config.xml), Continued

- **Update action declaration**
 - After declaring the bean with `form-bean`, we need to add two new attributes to the action element. In addition to the `path` and `type` attributes used in the earlier examples, we use:
 - **name:** a bean name matching the name in the `form-bean` declaration.
 - **scope:** request or session. Surprisingly, session is the default, but always explicitly list the scope anyhow. We want request here.
 - Here is an example.


```
<action path="/actions/register3"
  type="coreservlets.RegisterAction3"
  name="userFormBean"
  scope="request">
```

10 Apache Struts Beans www.coreservlets.com

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="userFormBean"
      type="coreservlets.UserFormBean"/>
  </form-beans>
  <action-mappings>
    ...
    <action path="/actions/register3"
      type="coreservlets.RegisterAction3"
      name="userFormBean"
      scope="request">
      <forward name="bad-address"
        path="/WEB-INF/results/bad-address3.jsp"/>
      <forward name="bad-password"
        path="/WEB-INF/results/bad-password3.jsp"/>
      <forward name="success"
        path="/WEB-INF/results/result3.jsp"/>
    </action>
  </action-mappings></struts-config>
```

11 Apache Struts Beans www.coreservlets.com

Step 2 (Define a Form Bean)

- **A form bean is a Java object that will be automatically filled in with the incoming form parameters and passed to the `execute` method. Requirements:**
 - **It must extend `ActionForm`.**
 - The argument to `execute` is of type `ActionForm`. Cast the value to your real type, and then each bean property has the value of the request parameter with the matching name.
 - **It must have a zero argument constructor.**
 - The system will automatically call this default constructor.
 - **It must have settable bean properties that correspond to the request parameter names.**
 - That is, it must have a `setBlah` method corresponding to each incoming request parameter that you want inserted automatically. The properties should be of type `String` (i.e., each `setBlah` method should take a `String` as an argument).
 - **It must have gettable bean properties for each property that you want to output in the JSP page.**
 - That is, it must have a `getBlah` method corresponding to each bean property that you want to display in JSP without using Java syntax.

12 Apache Struts Beans www.coreservlets.com

Step 2 (Define a Form Bean) -- Code Example

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

public class UserFormBean extends ActionForm {
    private String email = "Missing address";
    private String password = "Missing password";

    public String getEmail() { return(email); }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() { return(password); }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

12

Apache Struts Beans

www.coreservlets.com

Step 3 (Create Results Beans)

- These are normal beans of the type used in regular MVC (i.e., implemented with RequestDispatcher)
 - The form bean represents the *input* data: the data that came from the HTML form. In most applications, the more important type of data is the *result* data: the data created by the business logic to represent the results of the computation or database lookup.
 - Results beans need to have getter and setter methods like normal JavaBeans, but need not extend any particular class or be declared in struts-config.xml.
 - They are stored in request, session, or application scope with the setAttribute method of HttpServletRequest, HttpSession, or ServletContext, respectively.

14

Apache Struts Beans

www.coreservlets.com

Step 3, Bean Code Example

```
package coreservlets;

public class SuggestionBean {
    private String email;
    private String password;

    public SuggestionBean(String email, String password) {
        this.email = email;
        this.password = password;
    }

    public String getEmail() {
        return(email);
    }

    public String getPassword() {
        return(password);
    }
}
```

15

Apache Struts Beans

www.coreservlets.com

Step 3, Utility Code Example (To Build SuggestionBean)

```
package coreservlets;

public class SuggestionUtils {
    private static String[] suggestedAddresses =
        { "president@whitehouse.gov",
          "gates@microsoft.com",
          "palmisano@ibm.com",
          "ellison@oracle.com" };
    private static String chars =
        "abcdefghijklmnopqrstuvwxyz0123456789#@$%^*?!";

    public static SuggestionBean getSuggestionBean() {
        String address = randomString(suggestedAddresses);
        String password = randomString(chars, 8);
        return(new SuggestionBean(address, password));
    }
    ...
}
```

16

Apache Struts Beans

www.coreservlets.com

Step 4 (Create an Action Object to Handle Requests)

- This example is similar to the previous one except that we do not call request.getParameter explicitly.
 - Instead, we extract the request parameters from the already populated form bean.
 - Specifically, we take the ActionForm argument supplied to execute, cast it to UserFormBean (our concrete class that extends ActionForm), and then call getter methods on that bean.
 - Also, we create a SuggestionBean and store it in request scope for later display in JSP.

17

Apache Struts Beans

www.coreservlets.com

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

```
public class RegisterAction3 extends Action {
    public ActionForward execute(ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
        UserFormBean userBean = (UserFormBean)form;
        String email = userBean.getEmail();
        String password = userBean.getPassword();
        SuggestionBean suggestionBean =
            SuggestionUtils.getSuggestionBean();
        request.setAttribute("suggestionBean", suggestionBean);
        if ((email == null) ||
            (email.trim().length() < 3) ||
            (email.indexOf("@") == -1)) {
            return(mapping.findForward("bad-address"));
        } else if ((password == null) ||
            (password.trim().length() < 6)) {
            return(mapping.findForward("bad-password"));
        } else {
            return(mapping.findForward("success"));
        }
    }
}
```

18

Apache Struts Beans

www.coreservlets.com

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

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>New Account Registration</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>New Account Registration</H1>
<FORM ACTION=" ../actions/register3.do"
METHOD="POST">
  Email address: <INPUT TYPE="TEXT"
NAME="email"><BR>
  Password: <INPUT TYPE="PASSWORD"
NAME="password"><BR>
  <INPUT TYPE="SUBMIT" VALUE="Sign Me Up!">
</FORM>
</CENTER>
</BODY></HTML>
```

16

Apache Struts Beans

www.coreservlets.com

Step 6 (Display Results in JSP) Alternatives for Beans

- Use JSP scripting elements.
 - This approach is out of the question; it is precisely what Struts is designed to avoid.
- Use `jsp:useBean` and `jsp:getProperty`.
 - This approach is possible, but these tags are a bit clumsy and verbose.
- Use the JSTL `c:out` tag.
 - This approach is not a bad idea, but it is hardly worth the bother loading JSTL just for this situation. So, unless you are using JSTL elsewhere in your application anyhow, don't bother with `c:out`.

20

Apache Struts Beans

www.coreservlets.com

Step 6 (Display Results in JSP) Alternatives for Beans

- Use the JSP 2.0 expression language.
 - This is perhaps the best option if the server supports JSP 2.0. In these examples, we will assume that the application needs to run on multiple servers, some of which support only JSP 1.2.
- Use the Struts `bean:write` tag.
 - This is by far the most common approach when using Struts. Note that, unlike `c:out` and the JSP 2.0 expression language, `bean:write` automatically filters special HTML characters, replacing `<` with `<` and `>` with `>`. You can disable this behavior by specifying `<bean:write name="beanName" property="beanProperty" filter="false">`.
 - So, in this example we use `bean:write`. Before we do so, however, we have to import the "bean" tag library as follows.


```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
```

21

Apache Struts Beans

www.coreservlets.com

Step 6 (Display Results in JSP) First Possible Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Illegal Email Address</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Illegal Email Address</H1>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
The address
"<bean:write name="userFormBean" property="email"/>"
is not of the form username@hostname (e.g.,
<bean:write name="suggestionBean" property="email"/>).
<P>
Please <A HREF=" ../forms/register3.jsp">try again</A>.
</CENTER>
</BODY></HTML>
```

22

Apache Struts Beans

www.coreservlets.com

Step 6 (Display Results in JSP) Second Possible Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Illegal Password</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Illegal Password</H1>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
The password
"<bean:write name="userFormBean" property="password"/>"
is too short; it must contain at least six characters.
Here is a possible password:
<bean:write name="suggestionBean" property="password"/>.
<P>
Please <A HREF=" ../forms/register3.jsp">try again</A>.
</CENTER>
</BODY></HTML>
```

23

Apache Struts Beans

www.coreservlets.com

Step 6 (Display Results in JSP) Third Possible Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Success</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>You have registered successfully.</H1>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<UL>
  <LI>Email Address:
    <bean:write name="userFormBean" property="email"/>
  <LI>Password:
    <bean:write name="userFormBean" property="password"/>
</UL>
<P>
(Version 3)
</CENTER>
</BODY></HTML>
```


24

Apache Struts Beans

www.coreservlets.com

Example 1: Results


- First, the HTML form is invoked with the URL `http://localhost/struts-test/forms/register3.jsp`



www.coreservlets.com

Example 1: Results


- In the first case, the form is given "Bill Gates" as an email address and is submitted.



www.coreservlets.com


Example 1: Results

- The form's ACTION results in the URL `http://localhost/struts-test/actions/register3.do`.
 - This address is mapped by `struts-config.xml` to the `RegisterAction3` class, whose `execute` method is invoked.
 - The second argument to `execute` is the form bean; it contains all of the incoming request data
 - This method examines the email address from the form-bean, determines it is illegal because it lacks an "@" sign, and returns `mapping.findForward` with a value of "bad-address".
 - That value is mapped by `struts-config.xml` to `/WEB-INF/results/bad-address3.jsp`, which is the final result displayed to the user for this input.
 - Unlike the previous example, this JSP page shows both the illegal email address (taken from the form bean) and a suggested legal one (taken from the `SuggestionBean`).



www.coreservlets.com


Example 1: Results



www.coreservlets.com


Example 1: Results

- In the second case, the form is given a legal email address (`balmer@microsoft.com`) but a password that is only five characters long (`steve`).



www.coreservlets.com

Example 1: Results



www.coreservlets.com

Example 1: Results

- In the third case, the form is given a legal email address and password.
- The form's ACTION results in the URL `http://localhost/struts-test/actions/register3.do`.
 - This address is mapped by `struts-config.xml` to the `RegisterAction3` class, whose `execute` method is invoked.
 - The second argument to `execute` is the form bean; it contains all of the incoming request data
 - This method finds no problem with either the email address or the password in the form bean, so returns `mapping.findForward` with a value of "success".
 - That value is mapped by `struts-config.xml` to `/WEB-INF/results/result3.jsp`, which is the final result displayed to the user for this input.

www.coreservlets.com

Example 1: Results

www.coreservlets.com

Example 2: Properties Files

- This example is very similar to the previous one. However, some of the standard prompts and messages are taken from a properties file instead of being hardcoded in the JSP pages. To accomplish this, two new Struts features are used.
 - Declaring the properties file with the `message-resources` entry in `struts-config.xml`.
 - Using `bean:message` to output messages from the properties file.
- Advantages to using messages from properties files
 - If a message is used in several places, it can be updated with a single change. This is consistent with the Struts philosophy of making as many changes as possible in config files, not in Java or JSP code.
 - If you use messages pervasively in your JSP pages, you can internationalize your application by having multiple properties files corresponding to the locale (e.g., `application.properties`, `application-es.properties`, `application-jp.properties`, etc., as with standard I18N in Java).

www.coreservlets.com

Step 1 (Modify struts-config.xml)

- The Action object and JSP URLs are declared with `action` and `forward` in the same manner as in the previous examples.
- In addition, the `message-resources` element is used to refer to a properties file as follows.


```
<message-resources parameter="resources.application"
                    null="false"/>
```

 - The `parameter` attribute refers to the location of the properties file, relative to `WEB-INF/classes` and with the `.properties` file extension implied. So, for example, "resources.application" refers to `WEB-INF/classes/resources/application.properties`, and "foo.bar.baz" refers to `WEB-INF/classes/foo/bar/baz.properties`.
 - The `null` attribute determines whether missing messages should be flagged. If the value is true (or the `null` attribute is omitted), references to nonexistent messages result in empty strings. If the value is false, references to nonexistent messages result in warning messages like `???keyName???`.

www.coreservlets.com

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="userFormBean"
              type="coreservlets.UserFormBean"/>
  </form-beans>
  <action-mappings>
    ...
    <action path="/actions/register4"
            type="coreservlets.RegisterAction3"
            name="userFormBean"
            scope="request">
      <forward ...> ...
    </action>
  </action-mappings>
  <message-resources parameter="resources.application"
                    null="false"/>
</struts-config>
```

www.coreservlets.com

Step 1 (Properties File)

- `WEB-INF/classes/resources/application.properties`

```
form.title=New Account Registration
form.emailPrompt=Email Address
form.passwordPrompt=Password
form.buttonLabel=Sign Me Up!
form.emailError=Illegal Email Address
form.passwordError=Illegal Password
form.passwordLength=six
```

www.coreservlets.com

Steps 2, 3, and 4

- **Define a form bean.**
 - This example uses the same UserFormBean as the previous example.
- **Create results beans.**
 - This example uses the same SuggestionBean and SuggestionUtil classes as the previous example.
- **Create an Action object to handle requests.**
 - This example uses the same RegisterAction3 class as the previous example (download the source code here).
 - However, the struts-config.xml file associates the Action with a different incoming URL (register4.do instead of register3.do), and associates the return values with different JSP pages.

37

Apache Struts Beans

www.coreservlets.com

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

- **Instead of directly listing the "Email Address" and "Password" prompts, the form takes them from the properties file.**
 - That way, if the prompts change (or if you have multiple versions in different languages), the prompts can be updated without modifying the actual JSP page.
 - Also, as we will see later, the same prompts are used in the JSP page that shows the results, so extracting the prompts from the properties file limits changes to one location, even though the prompts are used in multiple locations.

38

Apache Struts Beans

www.coreservlets.com

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

```
<!DOCTYPE ...>
<% taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<HTML>
<HEAD><TITLE><bean:message key="form.title"/>
</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1><bean:message key="form.title"/></H1>
<FORM ACTION="../actions/register4.do" METHOD="POST">
  <bean:message key="form.emailPrompt"/>:
  <INPUT TYPE="TEXT" NAME="email"><BR>
  <bean:message key="form.passwordPrompt"/>:
  <INPUT TYPE="PASSWORD" NAME="password"><BR>
  <INPUT TYPE="SUBMIT"
    VALUE="<bean:message key="form.buttonLabel"/>">
</FORM>
</CENTER>
</BODY></HTML>
```

39

Apache Struts Beans

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><bean:message key="form.emailError"/>
</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1><bean:message key="form.emailError"/></H1>
The address
" <bean:write name="userFormBean" property="email"/>
is not of the form username@hostname (e.g.,
<bean:write name="suggestionBean" property="email"/>).
<P>
Please <A HREF="../forms/register4.jsp">try again</A>.
</CENTER>
</BODY></HTML>
```

40

Apache Struts Beans

www.coreservlets.com

Step 6 (Display Results in JSP) Second Possible Page

```
<!DOCTYPE ...>
<HTML>
<% taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<HEAD><TITLE><bean:message key="form.emailError"/>
</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1><bean:message key="form.emailError"/></H1>
The address
" <bean:write name="userFormBean" property="email"/>
is not of the form username@hostname (e.g.,
<bean:write name="suggestionBean" property="email"/>).
<P>
Please <A HREF="../forms/register4.jsp">try again</A>.
</CENTER>
</BODY></HTML>
```

41

Apache Struts Beans

www.coreservlets.com

Step 6 (Display Results in JSP) Third Possible Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Success</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>You have registered successfully.</H1>
<% taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<UL>
  <LI><bean:message key="form.emailPrompt"/>:
    <bean:write name="userFormBean" property="email"/>
  <LI><bean:message key="form.passwordPrompt"/>:
    <bean:write name="userFormBean" property="password"/>
</UL>
(Version 4)
</CENTER>
</BODY></HTML>
```

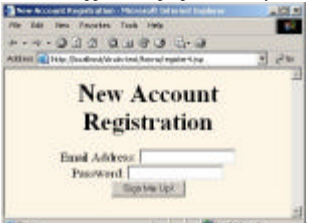
42

Apache Struts Beans

www.coreservlets.com

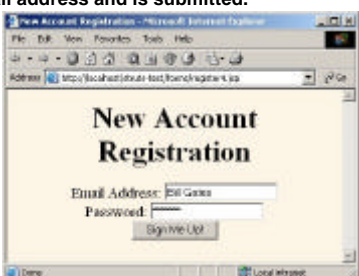
Example 2: Results

- First, the HTML form that uses bean:message is invoked with the URL `http://localhost/struts-test/forms/register4.jsp`.
 - Given the messages in `WEB-INF/classes/resources/application.properties`, this yields the following result.



Example 2: Results

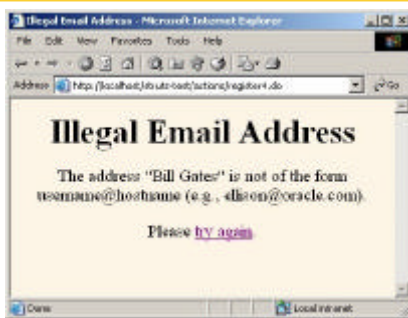
- In the first case, this form is given "Bill Gates" as an email address and is submitted.



Example 2: Results


- The form's ACTION results in the URL `http://localhost/struts-test/actions/register4.do`.
 - This address is mapped by `struts-config.xml` to the `RegisterAction3` class, whose `execute` method is invoked.
 - This method examines the email address, determines it is illegal because it lacks an "@" sign, and returns `mapping.findForward` with a value of "bad-address".
 - That value is mapped by `struts-config.xml` to `/WEB-INF/results/bad-address4.jsp`, which is the final result displayed to the user for this input.
 - Note that this example uses the same Action object (`RegisterAction3`) as the previous example, yet it uses a different incoming URL and different final JSP pages. This illustrates the flexibility that the `struts-config.xml` file gives over standard MVC where the JSP page addresses are hard-coded into the servlet code.
 - Given the messages in `WEB-INF/classes/resources/application.properties`, this yields the result on the next slide.

Example 2: Results



Example 2: Results

- In the second case, the form is given a legal email address (`balmer@microsoft.com`) but a password that is only five characters long (`steve`).



Example 2: Results

- The form's ACTION results in the URL `http://localhost/struts-test/actions/register4.do`.
 - This address is mapped by `struts-config.xml` to the `RegisterAction3` class, whose `execute` method is invoked.
 - This method examines the password, determines it is too short, and returns `mapping.findForward` with a value of "bad-password".
 - That value is mapped by `struts-config.xml` to `/WEB-INF/results/bad-password4.jsp`, which is the final result displayed to the user for this input.
 - Given the messages in `WEB-INF/classes/resources/application.properties`, this yields the result on the next slide.

Example 2: Results

www.coreservlets.com

Example 2: Results

- In the third case, the form is given a legal email address and password.
- The form's ACTION results in the URL **http://localhost/struts-test/actions/register4.do**.
 - This address is mapped by struts-config.xml to the RegisterAction3 class, whose execute method is invoked.
 - This method finds no problem with either the email address or the password, so returns mapping.findForward with a value of "success".
 - That value is mapped by struts-config.xml to /WEB-INF/results/result4.jsp, which is the final result displayed to the user for this input.
 - Given the messages in WEB-INF/classes/resources/application.properties, this yields the result on the next slide.

www.coreservlets.com

Example 2: Results

www.coreservlets.com

Using the JSP 2.0 Expression Language with Struts

- **Pros**
 - The JSP 2.0 EL is shorter and clearer
 - The JSP 2.0 EL can access bean subproperties and collection elements
- **Cons**
 - The bean:write tag filters HTML chars
 - There is no EL equivalent of bean:message
- **To use the EL, you have to change the web.xml declaration to the JSP 2.0 version:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://java.sun.com/xml/ns/j2ee web-app_2_4.xsd"
  version="2.4">
...
</web-app>
```

www.coreservlets.com

Struts Example: Confirmation Page (Classic Style)

```
...
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>
...

```

www.coreservlets.com

Struts Example: Confirmation Page (JSP 2.0 Style)

```
...
Congratulations. You are now signed up for the
Single Provider of Alert Memos network!
<UL>
  <LI>First name: ${contactFormBean.firstName}
  <LI>Last name: ${contactFormBean.lastName}
  <LI>Email address: ${contactFormBean.email}
  <LI>Fax number: ${contactFormBean.faxNumber}
</UL>
...

```

www.coreservlets.com

Summary

- **Form-beans extend ActionForm and are declared with form-bean in struts-config.xml**
- **Instantiate a form-bean by casting the ActionForm argument of execute to the concrete class**
 - The bean properties that match request parameter names are automatically filled in
- **Results beans extend no particular classes**
- **Both form and results beans can be output with bean:write or the JSP 2.0 expression language**
- **Load properties files with message-resources**
- **Output properties with bean:message**

64 Apache Struts 2.0 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