



Step by Step Guide for building a simple Struts Application




1



Sang Shin

sang.shin@sun.com
www.javapassion.com
Java™ Technology Evangelist
Sun Microsystems, Inc.

2



Disclaimer & Acknowledgments

- Even though Sang Shin is a full-time employees of Sun Microsystems, the contents here are created as their own personal endeavor and thus does not reflect any official stance of Sun Microsystems.
- Sun Microsystems is not responsible for any inaccuracies in the contents.
- Acknowledgments:
 - The source code examples are from [Keld Hansen](#)



3




Revision History

- 11/10/2003: version 1: created by Sang Shin
- Things to do
 - Speaker notes need to be added
 - Contents still need to be polished


4



Sample App We are going to build





5




Sample App

- ? Keld Hansen's submit application
- ? The source files and Ant build.xml file can be found in the hands-on/homework material in our class website
 - Creating ActionForm object
 - Creating Action object
 - Forwarding at either success or failure through configuration set in struts-config.xml file
 - Input validation
 - Internationalization
- ? You can also build it using NetBeans


6



Steps to follow



7



Steps

1. Create development directory structure
2. Write web.xml
3. Write struts-config.xml
4. Write ActionForm classes
5. Write Action classes
6. Create ApplicationResource.properties
7. Write JSP pages
8. Write ant build script
9. Build, deploy, and test the application

8



Step 1: Create Development Directory Structure




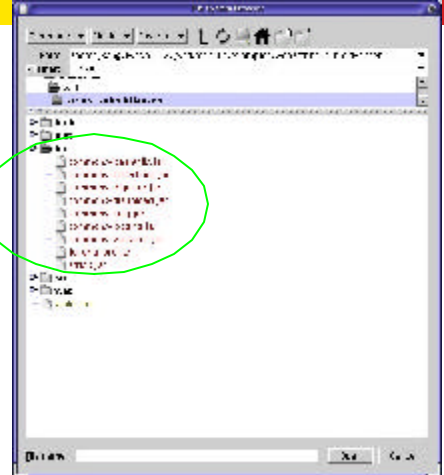
9



Development Directory Structure

- ? Same development directory structure for any typical Web application
 - We will use the source/build directory structure of J2EE 1.4 SDK sample Web applications
- ? Ant build script should be written accordingly

10

**Struts
*.jar files**

11



Step 2: Write web.xml Deployment Descriptor



12

web.xml

- ? Same structure as any other Web application
 - ActionServlet is like any other servlet
 - Servlet definition and mapping of ActionServlet
- ? There are several Struts specific `<init-param>` elements
- ? Struts tag libraries also need to be defined

13

Example: web.xml

```

1 <!DOCTYPE web-app
2 PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
3 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
4
5 <web-app>
6   <display-name>Advanced J2EE Programming Class Sample App</display-name>
7
8   <!-- Standard Action Servlet Configuration (with debugging) -->
9   <servlet>
10     <servlet-name>action</servlet-name>
11     <servlet-class>
12       org.apache.struts.action.ActionServlet
13     </servlet-class>
14     <init-param>
15       <param-name>application</param-name>
16       <param-value>ApplicationResources</param-value>
17     </init-param>
18     <init-param>
19       <param-name>config</param-name>
20       <param-value>/WEB-INF/struts-config.xml</param-value>
21     </init-param>
22   </servlet>

```

14

Example: web.xml

```

1 <!-- Standard Action Servlet Mapping -->
2 <servlet-mapping>
3   <servlet-name>action</servlet-name>
4   <url-pattern>*.do</url-pattern>
5 </servlet-mapping>
6
7 <!-- Struts Tag Library Descriptors -->
8 <taglib>
9   <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
10  <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
11 </taglib>
12 <taglib>
13   <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
14   <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
15 </taglib>
16 <taglib>
17   <taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
18   <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
19 </taglib>
20
21 </web-app>
22

```

15

Step 3: Write struts-config.xml



16



struts-config.xml

- ? Identify required input forms and then define them as <form-bean> elements
- ? Identify required Action's and then define them as <action> elements within <action-mappings> element
 - make sure same value of name attribute of <form-bean> is used as the value of name attribute of <action> element
 - define if you want input validation
- ? Decide view selection logic and specify them as <forward> element within <action> element

17



struts-config.xml: <form-beans>

```

1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2
3  <!DOCTYPE struts-config PUBLIC
4      "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
5      "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
6
7  <struts-config>
8
9      <!-- ===== Form Bean Definitions ===== -->
10     <form-beans>
11
12         <form-bean     name="submitForm"
13                        type="submit.SubmitForm"/>
14
15     </form-beans>

```

18



struts-config.xml: <action-mappings>

```

1
2  <!-- ===== Action Mapping Definitions ===== -->
3  <action-mappings>
4
5      <action  path="/submit"
6              type="submit.SubmitAction"
7              name="submitForm"
8              input="/submit.jsp"
9              scope="request"
10             validate="true">
11          <forward name="success" path="/submit.jsp"/>
12          <forward name="failure" path="/submit.jsp"/>
13      </action>
14
15  </action-mappings>
16
17 </struts-config>

```

19



Step 4: Write ActionForm classes



20



ActionForm Class

- ? Extend `org.apache.struts.action.ActionForm` class
- ? Decide set of properties that reflect the input form
- ? Write getter and setter methods for each property
- ? Write `validate()` method if input validation is desired (Struts 1.0)

21



Write ActionForm class

```

1 package submit;
2
3 import javax.servlet.http.HttpServletRequest;
4 import org.apache.struts.action.*;
5
6 public final class SubmitForm extends ActionForm {
7
8     /* Last Name */
9     private String lastName = "Hansen"; // default value
10    public String getLastName() {
11        return (this.lastName);
12    }
13    public void setLastName(String lastName) {
14        this.lastName = lastName;
15    }
16
17    /* Address */
18    private String address = null;
19    public String getAddress() {
20        return (this.address);
21    }
22    public void setAddress(String address) {
23        this.address = address;
24    }
25    ...

```

22



Write validate() method

```

1 public final class SubmitForm extends ActionForm {
2
3     ...
4     public ActionErrors validate(ActionMapping mapping,
5         HttpServletRequest request) {
6
7         ...
8
9         // Check for mandatory data
10        ActionErrors errors = new ActionErrors();
11        if (lastName == null || lastName.equals("")) {
12            errors.add("Last Name", new ActionError("error.lastName"));
13        }
14        if (address == null || address.equals("")) {
15            errors.add("Address", new ActionError("error.address"));
16        }
17        if (sex == null || sex.equals("")) {
18            errors.add("Sex", new ActionError("error.sex"));
19        }
20        if (age == null || age.equals("")) {
21            errors.add("Age", new ActionError("error.age"));
22        }
23        return errors;
24    }
25    ..
26 }

```

23



Step 5: Write Action classes



24



Action Classes

- ? Extend `org.apache.struts.action.Action` class
- ? Handle the request
 - Decide what kind of server-side Model objects (EJB, JDO, etc.) can be invoked
- ? Based on the outcome, select the next view

25



Example: Action Class

```

1 package submit;
2
3 import javax.servlet.http.*;
4 import org.apache.struts.action.*;
5
6 public final class SubmitAction extends Action {
7
8     public ActionForward execute(ActionMapping mapping,
9                                 ActionForm form,
10                                HttpServletRequest request,
11                                HttpServletResponse response) {
12
13         SubmitForm f = (SubmitForm) form; // get the form bean
14         // and take the last name value
15         String lastName = f.getLastName();
16         // Translate the name to upper case
17         //and save it in the request object
18         request.setAttribute("lastName", lastName.toUpperCase());
19
20         // Forward control to the specified success target
21         return (mapping.findForward("success"));
22     }
23 }

```

26



Step 6: Create ApplicationResource.properties and Configure web.xml accordingly



27



Resource file

- ? Create resource file for default locale
- ? Create resource files for other locales

28

Example: ApplicationResource.properties

```

1 errors.header=<h4>Validation Error(s)</h4><ul>
2 errors.footer=</ul><hr>
3
4 error.lastName=<li>Enter your last name
5 error.address=<li>Enter your address
6 error.sex=<li>Enter your sex
7 error.age=<li>Enter your age

```

29

Step 7: Write JSP pages



30

JSP Pages

- ? Write one JSP page for each view
- ? Use Struts tags for
 - Handling HTML input forms
 - Writing out messages

31

Example: submit.jsp

```

1 <%@ page language="java" %>
2 <%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
3 <%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
4 <%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
5
6 <html>
7 <head><title>Submit example</title></head>
8 <body>
9
10 <h3>Example Submit Page</h3>
11
12 <html:errors/>
13
14 <html:form action="submit.do">
15   Last Name: <html:text property="lastName"/><br>
16   Address: <html:textarea property="address"/><br>
17   Sex: <html:radio property="sex" value="M"/> Male
18       <html:radio property="sex" value="F"/> Female<br>
19   Married: <html:checkbox property="married"/><br>
20   Age: <html:select property="age">
21     <html:option value="a" 0-19/></html:option>
22     <html:option value="b" 20-49/></html:option>
23     <html:option value="c" 50-69/></html:option>
24   </html:select><br>
25   <html:submit/>
26 </html:form>

```

32

Example: submit.jsp

```

1 <logic:present name="lastName" scope="request">
2   Hello
3 <logic:equal name="submitForm" property="age" value="a">
4   young
5 </logic:equal>
6 <logic:equal name="submitForm" property="age" value="c">
7   old
8 </logic:equal>
9 <bean:write name="lastName" scope="request"/>
10 </logic:present>
11
12 </body>
13 </html>

```

33

Step 8: Write Ant Build Script



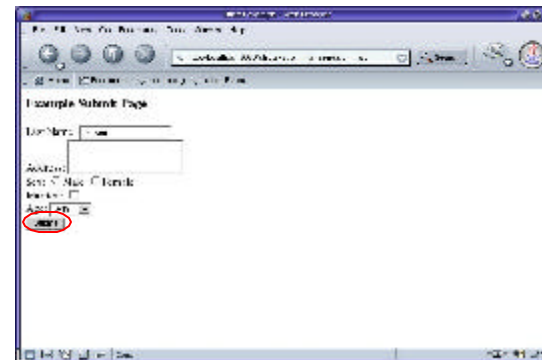
34

Step 9: Build, Deploy, and Test Application



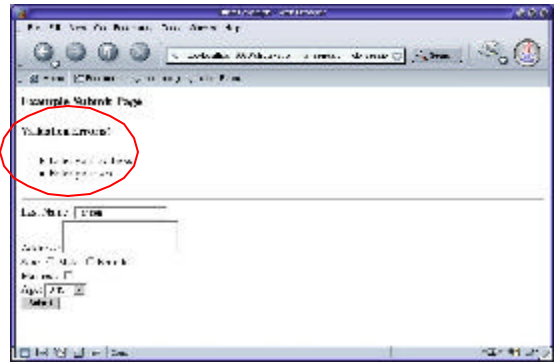
35

Accessing Web Application



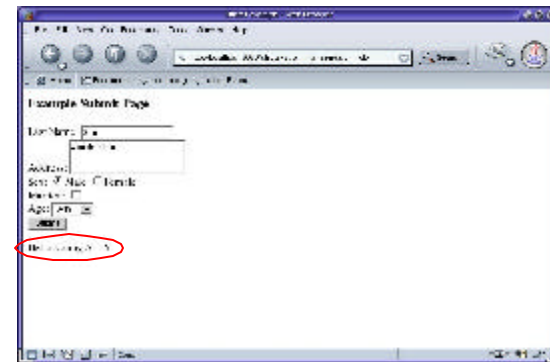
36

Accessing Web Application



37

Accessing Web Application



38

**Live your life
with Passion!**



39