Some words about

Struts

Hamid M. Porasl

1 Introduction to Struts

JSP is used as the presentation layer in n-tier web architectures to separate presentation from content. In Struts framework, JSPs represent the View in the Model-View-Controller design pattern.

JavaBeans components are nothing but Java objects that follow a well-defined design/naming pattern using public accessory methods for reading and mutator methods for modifying the Beans property values. There are also called the familiar getter/setter methods. Struts ActionFrorm class is actually nothing more than a Bean.

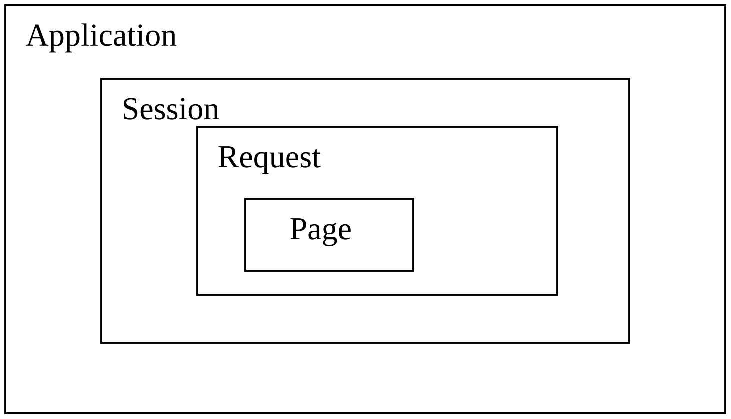


Figure 1 JSP object scopes

The <jsp:useBean> tag tries to obtain a reference to an instance that already does exist. It does so by using specified Id and scope. If the Bean was created previously and placed in a specific scope from another JSP, that instance will be used. If an instance isn't found, then a new instance is created. A sample JSP tag looks like follow:

```
<jsp:useBean id=”user”   class=”employee.user”   scope=”session” />
```

JSP keeps the front-end presentation separate from the middle and backend tiers. Custom tag libraries are a powerful feature available since JSP v1.1.

Struts (version 1.3.5) has four tag libraries, i.e.:

1. **org.apache.struts.taglib.bean**: The "struts-bean" tag library contains JSP custom tags useful in defining new beans (in any desired scope) from a variety of possible sources, as well as a tag to render a particular bean (or bean property) to the output response.

2. **org.apache.struts.taglib.html:** The "struts-html" tag library contains JSP custom tags useful in creating dynamic HTML user interfaces, including input forms.

3. **org.apache.struts.taglib.logic:** The "struts-logic" tag library contains tags that are useful in managing conditional generation of output text, looping over object collections for repetitive generation of output text, and application flow management.

4. **org.apache.struts.taglib.nested:** Nested tags & supporting classes extend the base struts tags to allow them to relate to each other in a nested nature. Containing:

    a. **org.apache.struts.taglib.nested.bean:** The nested bean tags extend the `org.apache.struts.taglib.bean` tags to allow them to relate to each other in a nested nature.

    b. **org.apache.struts.taglib.nested.html:** The nested html tags extend the `org.apache.struts.taglib.html` tags to allow them to relate to each other in a nested nature.

    c. **org.apache.struts.taglib.nested.logic**:The nested html tags extend the `org.apache.struts.taglib.logic` tags to allow them to relate to each other in a nested nature.

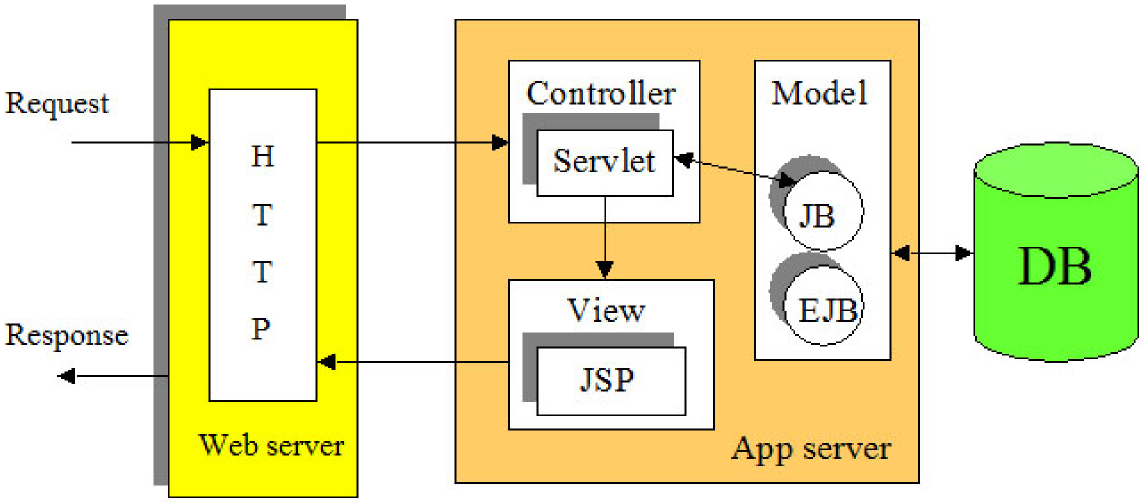


Figure 2 Data flow

1.1 Model

Action classes provide the business logic for the application. Dispatch is done from the Controller to the Action class. The dispatch is determined by using an XML based configuration file called *struts-config.xml*.

1.2 View

The View consists of JSP and a set of JSP custom tags that works in concrete with the Controller Servlet. There is no flow logic or business logic, and no model information is contained in the View. Using the *ActionForm* a class pass information from JSPs to the Model. The JSP file reads information from the ActionForm Bean using JSP tags. Tag libraries allows for a fast creation of forms for applications and also provides a built-in mechanism to use resource files for the internalization of screens.

1.3 Controller

The controller is a Servlet that uses the Command Design pattern to dispatch incoming requests to the appropriate Action classes. The controller Servlet, called the ActionServlet in the Struts framework, is configured in web server *web.xml* and also uses the *struts-config.xml* file for determining action mappings. There actually isn't any work that needs to be done other than configuring the web.xml to create an instance of *org.apache.struts.action.actionServlet*.

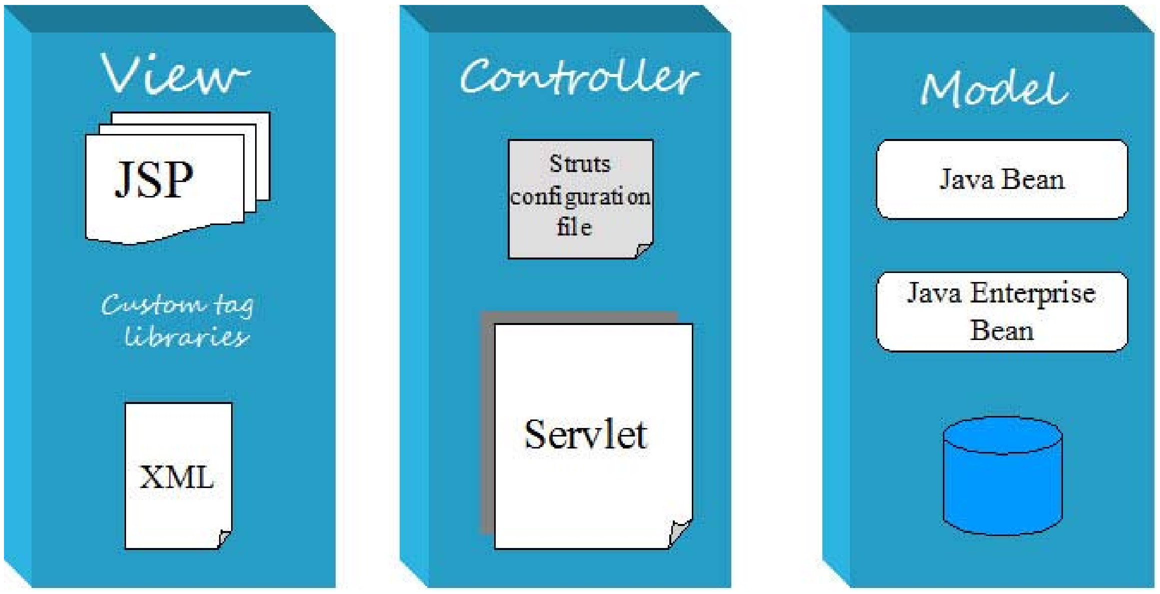


Figure 3 Struts model - MVC

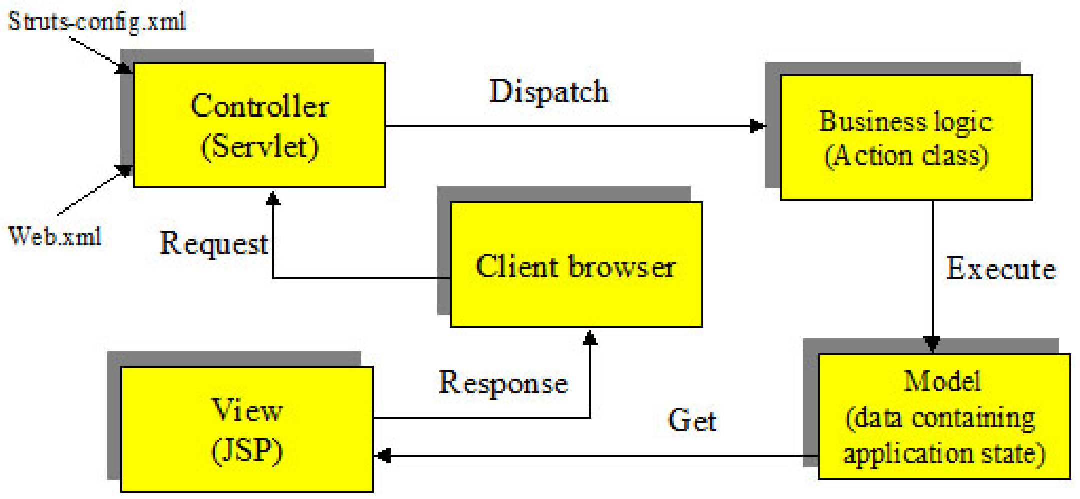


Figure 4 Application model using struts

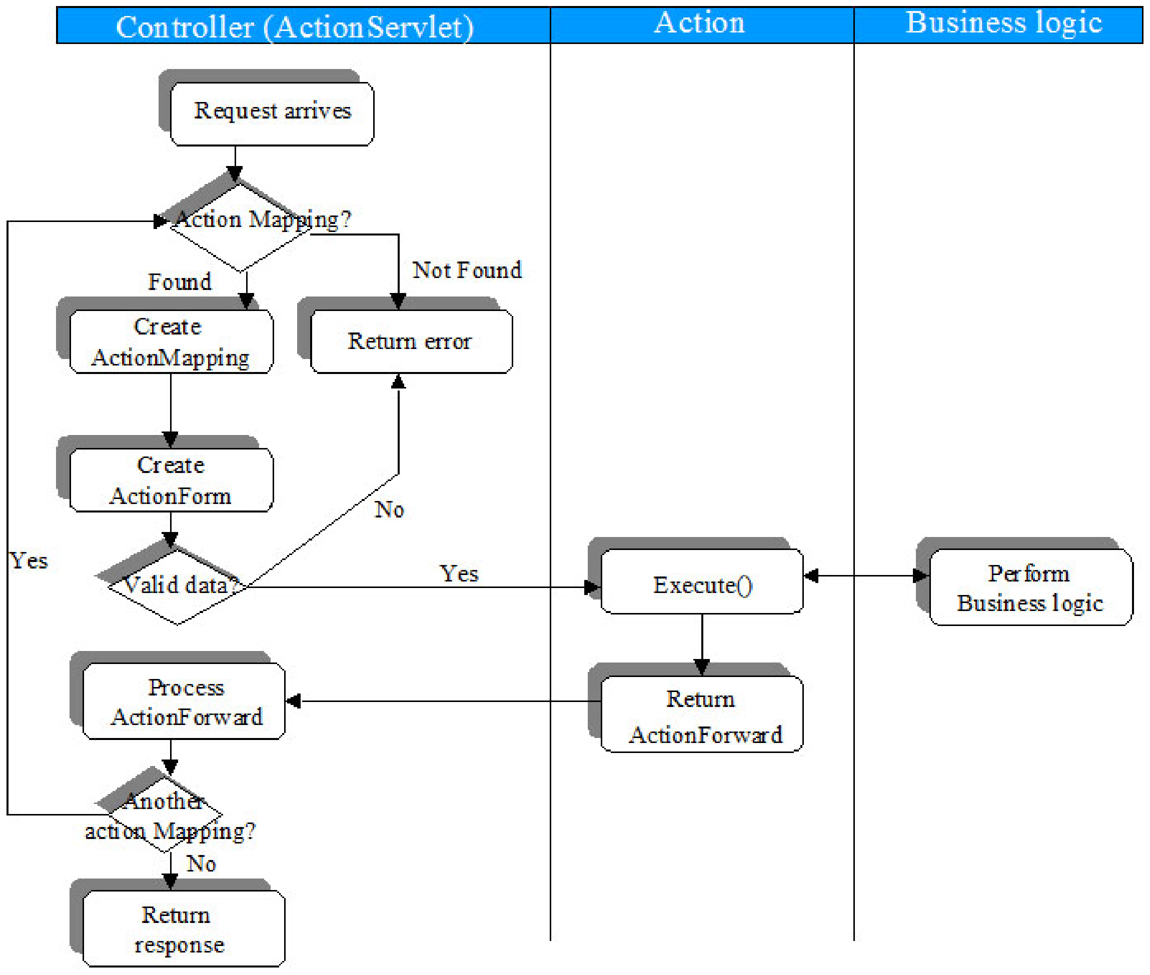


Figure 5   Struts activity diagram

1.4 Struts action class

1.4.1 Org.apache.struts.action.Action

The ActionServlet needs mechanism to determine how to route requests to Action classes. This is done by ***ActionMapping*** class. When ***ActionServlet*** knows about the mapping of a particular request to an instance of a particular Action class, the mapping is passed to the ***execute()*** method of the Action class. Below is an example of an action mapping:

```
<action-mappings>
<action         path=”/upload
                type=”UploadAction”
                name=”uploadForm”
                scope=”request”
                input=”/upload.jsp”>
<forward name=”success” path=”/display.jsp” />
</action>
</action-mappings>
```

path: is the request URI path used to select the mapping.
type: specifies fully qualified java class name of the Action implemented
name: specifies the name of the form Bean
scope: identifies the scope that can be requested or the session that the form bean creates
input: context-relative path of the input from to which control should be returned
validate: true or false
forward: provides Action-local names of logical ActionForward instance that can be returned by the action to control what does the actual presentation. It is possible to specify multiple forward declarations for each action.

1.4.2 Org.apache.struts.action.Actionforward

An ActionForward represents a destination that the ActionServlet might be directed to perform a ***RequestDispatcher.forward()*** or ***HttpServletResponse.sendRedirect()*** Both of these mechanisms are used to transfer control to another JSP or servlet. The forward does so from the server side; the ***sendRedirect*** actually sends a new URL to the client, An ActionForward can be specified in the struts-config.xml file as follow:

```
        <forward    name="display" path="/display.jsp"  redirect=”false”/>
```

The name property defines a logical name by which this instance may be looked up in relation to particular *ActionMapping*.

It is possible to use the global-forwards section in the Struts-config.xml file. Global forwards are used to create logical name mappings for communally used JSPs. Below is an example for usage of global forwards:

```
<global-forwards>
```

```
<forward name=”logoff”     path=”/logoff.do”/>
<forward name=”logon”      path=”/logon.jsp”/>
<forward name=”success”    path=”/mainMenu.jsp”/>
</global-forwards>
```

1.4.3 org.apache.struts.action.ActionError

The mechanism to return errors that occur during an ActionForm Validation uses the ***ActionErro*r** class. ActionError is an encapsulation of an individual error message returned by the ***validate()*** method of an ActionForm. The error consists of a message key that is used to look up the text from the appropriate message resource file. The text in the message resource file can contain up to four placeholder objects that can be used for replacement in the message text. Placeholder objects are referenced in the message text using the same syntax used by JDK ***MessageFormat*** class. For more information see *java.text.MessageFormat* API.

Errors are processed in *validate() method in ActionForm* as follow:

```
public ActionErrors validate(
    ActionMapping mapping,
    HttpServletRequest request) {

    ActionErrors errors = super.validate(mapping, request);

    //has the maximum length been exceeded?
    Boolean maxLengthExceeded =
        (Boolean) request.getAttribute(
            MultipartRequestHandler.ATTRIBUTE_MAX_LENGTH_EXCEEDED);

    if ((maxLengthExceeded != null) && (maxLengthExceeded.booleanValue())) {
        if (errors == null) {
            errors = new ActionErrors();
        }
        errors.add(
            ActionMessages.GLOBAL_MESSAGE ,
            new ActionMessage("maxLengthExceeded"));
        errors.add(
            ActionMessages.GLOBAL_MESSAGE ,
            new ActionMessage("maxLengthExplanation"));
    }
    return errors;
        }
```

1.4.4 org.apache.struts.action.ActionMessage

The ActionMessage work exactly the same as ActionErrror. ***ActionMessage*** introduced to Struts 1.1. This is useful when we want to display a generic information message to the user. As it is visible in example above ActionMessages like below can be included in Action class:

```
            ActionMessages  errors = new ActionMessages();
            errors.add(
                ActionMessages.GLOBAL_MESSAGE ,
                new ActionMessage("maxLengthExceeded"));
           saveMessages(request,errors);
```

JSP uses following tags to catch messages:

```
<logic:messagesParent message=”true”>
<tr>
    <html:messages id=”message” message=”true”>
    <td><bean:write name=”message” />>/td>
</html:messages>
</tr>
</logic:messagesPresent>
```

1.5 View

View components consists of the following:

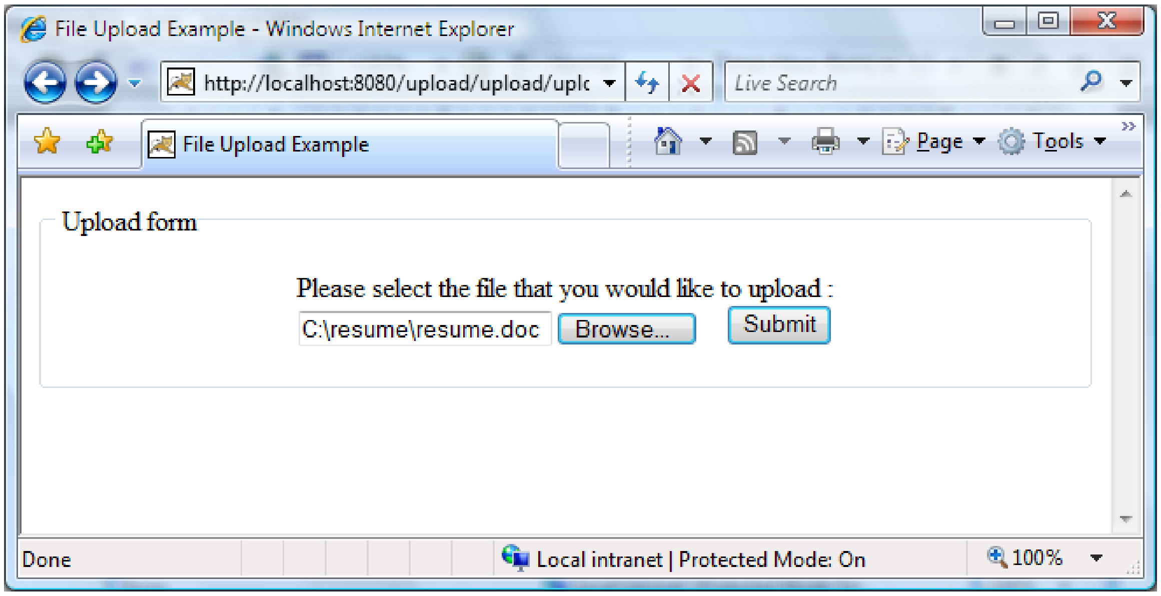
- *ActionForm*: which is an optiona; JavaBean associated with a form on JSP.
- *Tag libraries*: Custom tag libraries which are provided with Struts

1.5.1 Org.apache.struts.action.ActionForm

Each JSP that has input data entry requirements, it is possible to have an *ActionForm*. An ActionForm is a JavaBean optionally associated with one or more ***ActionMappings***. Such a Bean will have had its properties initialized from the corresponding request parameters before the corresponding action’s execute() method is called. When the properties of this Bean have been populated, but before the execute() method of the Action is called, this Bean’s validate() method is called. This is done only if the Bean has requested validation in the struts-donfig.xml

1.6 An example

Struts package which is available for download at http://www.apache.org already contains some example applications. I have adapted the upload example in order to upload a file to the server side.

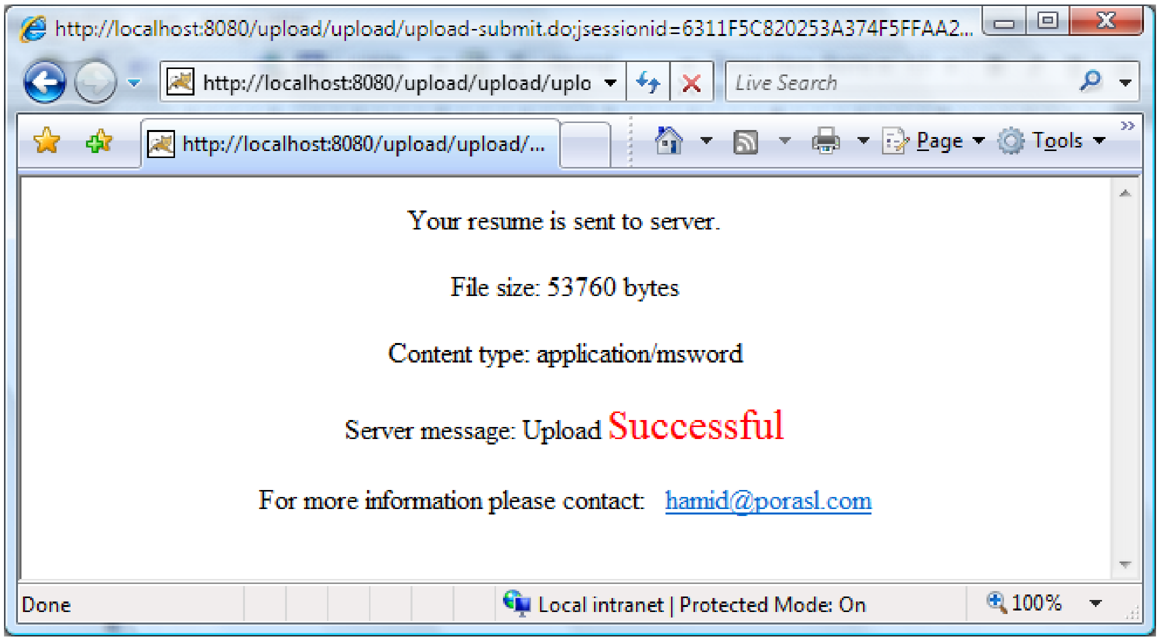

File Upload Example - Windows Internet Explorer
http://localhost:8080/upload/upload/uplc
Live Search
File Upload Example
Page
Tools
Done
Local intranet | Protected Mode: On
100%

Upload form

Please select the file that you would like to upload :

C:\resume\resume.doc Browse... Submit

Figure 6 An example using struts


http://localhost:8080/upload/upload/upload-submit.do;jsessionid=6311F5C820253A374F5FFAA2...
http://localhost:8080/upload/upload/uplo
Live Search
http://localhost:8080/upload/upload/...
Page
Tools
Done
Local intranet | Protected Mode: On
100%

Your resume is sent to server.

File size: 53760 bytes

Content type: application/msword

Server message: Upload Successful

For more information please contact: hamid@porasl.com

Figure 7 response from server after file upload

The WAR file is accessible at: http://porasl.com/software/struts/upload.war