Apache Struts— A Quick Introduction

*Mohd Iqbal**

Apache Struts is a Java/JSP-based framework for building web-based applications using J2EE platform. It makes developers more productive by giving them pre-built components to assemble applications that are flexible and scalable. Struts is built using industry best practices including the popular MVC design pattern and it can be deployed in a wide range of environments. It's an open source framework which is supported by a number of tool vendors. This article gives a quick introduction to struts framework using the classic HelloWorld application so that one may consider it for one's next web development.

Introduction

Struts is the premier framework for building Java-based Web applications. Using the Model-View-Controller (MVC) design patterns, Struts solves many of the problems associated with developing high-performance, business-oriented Web applications that use Java Servlets and Java Server Pages.

Strut framework was intended to address issues that were inherent in using either Java Server Pages or servlet implementation of an application. For instance, servlets can generate HTML pages, but doing so is very tedious. On the other hand, JSP can work easily with traditional HTML pages, but JSP pages have other disadvantages. JSP are special cases of servlets. In particular, separating content from the presentation of that content is difficult using JSP. So, both JSP and Servelets can work together to mitigate the disadvantages of each.

MVC Architecture: Basis for Struts

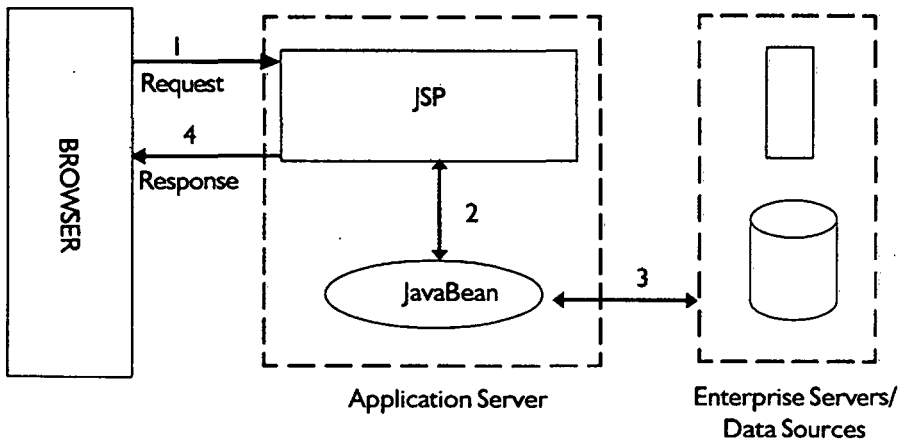
Understanding of the Model-View-Controller (MVC) architecture is crucial for understanding Struts. MVC is based on an older Graphical User Interface (GUI) design pattern that has been around for some time, with its origins in the Smalltalk World.

Based on MVC the JSP specifications advocated two philosophical approaches for building applications using JSP technology. These approaches, termed the Model 1 and Model 2 architectures, differ essentially in the location at which the bulk of the request processing was performed.

In the Model 1 architecture, the request is received and largely processed through the JSP. If the JSP page requires services from any other application component, such

* Software Engineer, AppLabs Technologies Pvt. Ltd., Hyderabad. E-mail: reach_miqbal@yahoo.co.in

Figure 1: Model 1 Architecture

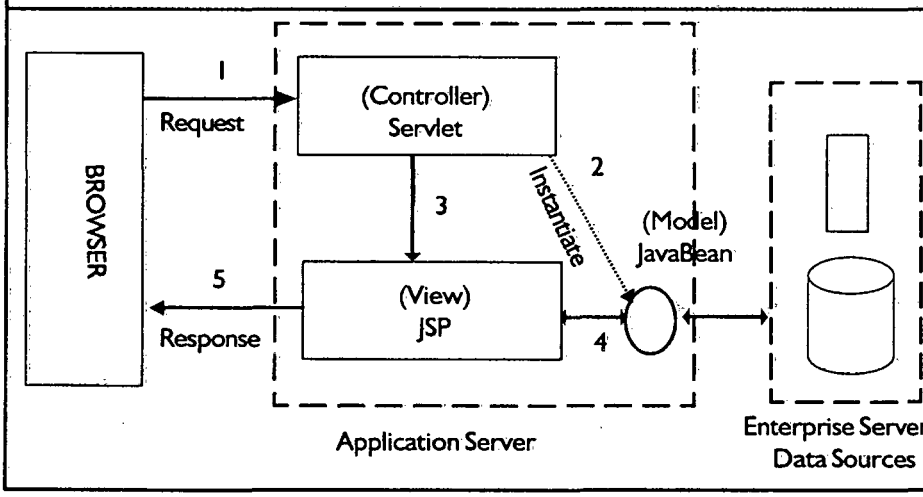


as a database, then you make the appropriate call from the page, return the data to the page, and format it for display.

Although the Model 1 architecture should be perfectly suitable for simple applications, it may not be desirable for complex projects that may lead to an unclear definition of roles and allocation of responsibilities, causing project-management headaches.

Model 2 approach takes the best characteristics of the JSP and servlet approaches and enables these technologies to work together. Here servlet acts as a processing layer (the controller). The servlet accepts requests and determines how to satisfy those requests. In that sense, the servlet regulates incoming requests and outgoing responses. The business logic represents the model in the MVC architecture. The

Figure 2: Model 2 Architecture



business logic code does the processing for the page. If the request that comes into the servlet is a database query, the servlet passes that request to a SQL call or similar database code. The JSP page is the presentation layer (the view), where the user interacts with the application. It prompts for input and displays the results. JSP simply passes the input data to the servlet, then receives the data returned and presents to the user.

The advantage of this Model 2 MVC architecture is very clear. First, it clearly separates computation from presentation, no processing occurs on the JSP page, and no data formatting in the servlet or business logic. The main advantage of this separation is that Java programmers can concentrate on servlet code and HTML writers on the JSP. Second, the controller servlet makes all of the decisions on the page. This helps us improve an application's performance and scalability, because requests can be directed to different components of the architecture and even to different servers.

Struts Framework : How does it Work?

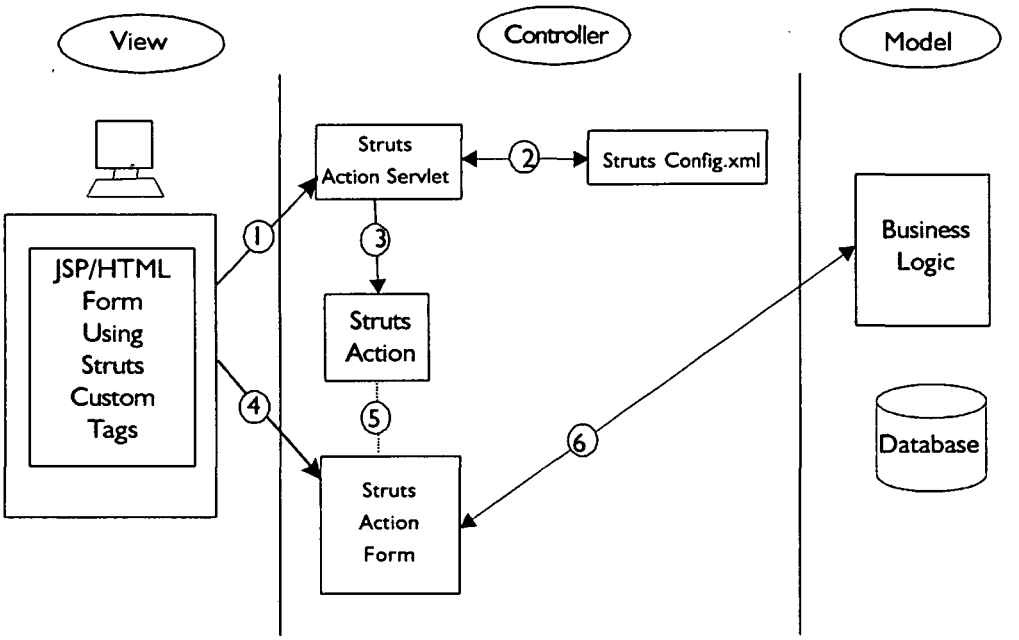
The Struts framework is broken down into four major components: The model that holds the business logic, the view which is a JSP page with a set of JSP custom tags that presents information to the user, the controller which is a servlet that handles incoming HTTP requests to an appropriate action, and a number of utility classes and XML files supporting the controller and loose coupling between the components. The controller plays the important role that handles and directs user requests to other objects in the framework.

The controller plays an important role handling and directing the user requests to other objects in the framework

The Struts framework revolves around an ActionMapping structure. According to this Struts provides a single ActionServlet (controller) that takes all requests from the browser and maps to an appropriate Action subclass in struts-config.xml file. This is based on a pattern called Front Controller. When browser requests come with parameters say, when a user submits a HTML form, the struts framework encloses the parameter in an ActionForm bean. If the user enters incorrect values in the form, the ActionForm may perform validations. The ActionForm bean is also used to pre-populate a form with default vales that are obtained from database or some backend system. The Action subclass that is mapped to the user request takes control from the ActionServlet and accesses the business logic stored in the model component for further processing. Then it forwards the control to a chosen view component to display the results of the action. This is depicted in Figure 3.

1. All browser requests are submitted to Struts ActionServlet.
2. Struts ActionServlet determines which Action to be called using Mapping, which is pre-configured in struts-config.xml file.
3. ActionServlet passes the control to Action subclass.

Figure 3: Struts in Action



Source: IBM.

4. When the HTML form is submitted, ActionForm subclass is automatically populated with the form data.
5. Action subclass can access the form data that is stored in the ActionForm subclass. This subclass is passed to the back-end business logic for further actions.
6. The Action subclass invokes the back-end business-logic.

"Hello World" using Struts

Let's create the classic "Hello World" application using Struts framework as described below.

- User will enter a person's name to say "Hello!" to the person and receives the output string "Hello< name>!"when submits.
- User cannot submit a blank form. If he does, he will receive an error message as help to fill the form correctly.
- Let us assume, the user cannot say Hello to a particular person "Iqbal". That means user cannot enter this name.
- The name entered by the user will be stored in a database.

The above scenario seems very basic but has little bit functionality for the model, view and controller components. The following steps can be used to develop the application.

1. Building the JSP File for View Component

The View is typically a JSP file. Struts provide a number of custom tags to develop this view. Here is the code for **hello.jsp** file.

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
 <head>
 <title><bean:message key="hello.jsp.title"/></title>
 <html:base/>
 </head>
 <body>
 <h2><bean:message key="hello.jsp.page.heading"/></h2>
 <html:errors/><p>
 <logic:present name="examples.hello" scope="request">
<h2> Hello <bean:write name="examples.hello" property="person" />!</h2>
 </logic:present>

<html:form action="/HelloWorld.do?action=gotName" focus="username" >
 <bean:message key="hello.jsp.prompt.person"/>
 <html:text property="person" size="16" maxlength="16"/><br>
 <html:submit property="submit" value="Submit"/>
 <html:reset/>
 </html:form><br>
 </body>
</html:html>
```

In the above JSP file the "taglib" tags indicate that it is using the Struts bean, html, and logic tag libraries. This is the standard syntax to load the tag library and make the tags available for use in the file. The HTML tags in this file, include <html:errors>, <html:form>, and <html:text>.

- <html:errors>— This tag is used to access and present the results of Struts' data validation.
- <html:form>— This tag is used for all HTML form processing. It binds the form fields to properties in Struts form beans. It also binds the fields into Struts' automatic form validation. Form beans are Java beans that transfer the values entered in a form to the controller component. Each form field will be tied to a corresponding property in the form bean.
- <html:text>— This tag is used inside an <html:form> tag. It ties a text field in the form to a property in the form bean.

There are two Struts bean tags in the file <bean:message> and <bean:write> tags:

- <bean:message>— This tag is used to output locale-specific text from a MessageResources bundle.
- <bean:write>— This is a general purpose tag that is used to output property values from a bean.

One logic tag is included in the application. <logic:present>. These tags deliver its output only if a bean is present and available to the JSP page.

There is one properties file for each locale for which the application needs to present localized text. The Application.properties file contains the actual text content to be stored. Text in Application.properties file is:

```
; Application Resource that are specific to the Hello.jsp file

hello.jsp.title=Hello-First program on Struts
hello.jsp.page.heading=Hello World! A first Struts application
hello.jsp.prompt.person=Please enter a name to say hello

; Validation and error messages for HelloForm.java and HelloAction.java

examples.hello.dont.talk.to.iqbal= don't talk to Iqbal!!!
examples.hello.no.person.error=Please enter a <i>PERSON</i> to say hello to!
```

2. Building the ActionForm Bean

When user clicks on submit button in hello.jsp file, the data from this form is populated in to java bean called form bean. Form bean has properties that match with all the fields of form hello.jsp. When the form is submitted, the bean properties are automatically populated. Form bean also support for automatic data validation and resetting of bean property values. Here is the code for Form Bean file called **HelloForm.java**

```
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public final class HelloForm extends ActionForm
{
 private String person = null;
 public String getPerson()
```

```
    {
    return (this.person);
    }

public void setPerson(String person)
    {
    this.person = person;
    }
public void reset(ActionMapping mapping,
HttpServletRequest request)
  {
          this.person = null;
  }

public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request)
  {
    ActionErrors errors = new ActionErrors();
    if ((person == null) || (person.length() < 1))
errors.add("person", new ActionError("examples.hello.no.person.error"));
    return errors;
  }
}
```

As you can see, a Struts form bean is nothing but a simple Java bean with methods added for input validation and to reset the properties to default values. This application has been defined to have two different types of errors: basic data/form validation errors and business logic errors.

- Form validation— In the data entry form, make sure that the user does not submit the form with the person field empty.
- Business logic— Enforce a rule that the user can't say hello to a person he is not allowed to talk to, in this case it is "Iqbal".

In the above program if validate() method returns an empty object of ActionErrors, Struts assumes there are no errors and processing moves to the Action class. If ActionErrors contains any ActionError elements, the user is redirected to the appropriate page to correct the errors. If processing is redirected for the user to correct the data entry, the ActionErrors object carries the individual ActionError elements back to the View for display. The View component can access the ActionErrors either directly or through the <html:errors> tag.

3. Building the Action Subclass

The central part of the Struts application is Action class. It is the center of all actions performed in response to the user requests. The Action class for the Hello World! application is in the file HelloAction.java looks like the following.

```
package examples.hello;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.util.MessageResources;
import org.apache.commons.beanutils.PropertyUtils;

public final class HelloAction extends Action
{
 public ActionForward execute(ActionMapping mapping,
              ActionForm form, HttpServletRequest request,
              HttpServletResponse response) throws Exception
    {
     MessageResources messages = getResources(request);
     ActionErrors errors = new ActionErrors();
 String person = (String)PropertyUtils.getSimpleProperty(form, "person");
     String badPerson = "Iqbal";
     if (person.equals(badPerson))
       {
      errors.add("person",
 new ActionError("examples.hello.dont.talk.to.iqbal", badPerson));
              saveErrors(request, errors);
              return (new ActionForward(mapping.getInput()));
       }
     HelloModel hm = new HelloModel();
     hm.setPerson(person);
     hm.saveToPersistentStore();
     request.setAttribute( Constants.HELLO_KEY, hm);
     request.removeAttribute(mapping.getAttribute());
     return (mapping.findForward("SayHello"));
     }
}
```

The execute() method is the primary method that must be overridden in an Action subclass. The framework calls this method after the form bean is populated and validated correctly. The four important parameters of execute method are:

- ActionMapping mapping—The ActionMapping provides access to the information stored in the configuration file (struts-config.xml) entry that configures the Action class.
- ActionForm form—This is the form bean. By this time, the form bean has been prepopulated and the validate() method has been called and returned with no errors. All the data entered by the user is available through the form bean.
- HttpServletRequest request—This is the standard JSP or Servlet request object.
- HttpServletResponse response—This is the standard JSP or Servlet response object.

As can be seen in the code, the text placed in the Application.properties file can be accesed by MessageResources messages = getResources(request);

This part of code loads a copy of the messageResources that were defined in the Application.properties file. Now Action subclass has full access to this file. The next part of the code describe the business logic validations, the logic here is that user is not allowed to say hello to a person called "Iqbal". As a programmer you can apply your own logic at this place. You can interact with the model component by creating an object of Model class. Here you can see the Controller creates a new Model component, sets a value in it, and calls a method to save the data to a persistent store. This is common way that Controller components will interact with a Model. You can pass data to the view component using standard servlet/JSP setAttribute() and getAttribute() method calls. The final step of the controller component is to forward the control to the corresponding view chosen to display the result.

Associated with the action subclass is a file called Constants.java. When you pass an object from the Action subclass to the view component using request.setAtribute(), you need to provide a name, that the JSP file can use to retrieve the object. In Struts, a convention has been adopted using a file called Constants.java to define these names. Basically it contains application scope attributes to store data. Here is the code for **Constants.java**

```
package examples.hello;

public final class Constants
{
 public static final String HELLO_KEY = "examples.hello";
}
```

4. Building the Model Component

Let's see the model component class that simply contains the name of the person user wants to say hello to. The file HelloModel.java looks like the following.

```
package examples.hello;

public class HelloModel
```

```
{
    private String _person = null;
    public String getPerson()
    {
            return this._person;
    }

    public void setPerson(String person)
    {
    this._person = person;
    }

    public void saveToPersistentStore()
    {
    // code for storing the data in database goes here.
    }
}
```

The above program is very basic, simple Model component like Javabean. The saveToPersistentStore() method is just a stub method that in a real application might store the person in a database. Although this is a very basic example, it demonstrates a primary strength of the MVC framework. Using Model components to hide the implementation details for interacting with remote systems is one of the keys to using Struts effectively.

5. Building the struts-config file:

The Struts framework breaks down the application into components to simplify and speed up development. The job of the struts-config.xml file is to let you specify how the components go together and identify when they should be used. Here is the program for Struts-config.xml file.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
 <form-beans>
 <form-bean name="HelloForm" type="examples.hello.HelloForm"/>
 </form-beans>
 <action-mappings>
 <action path = "/HelloWorld"
 type = "examples.hello.HelloAction"
```

```
name = "HelloForm"
scope = "request"
validate = "true"
input = "/hello.jsp" >
<forward name="SayHello" path="/hello.jsp" />
</action>
</action-mappings>
<message-resources parameter="examples.hello.Application"/>
</struts-config>
```

The above configuration file contains one <form-bean>, one .<action>, and one <message-resource> entry. This configuration file says:

- Only one bean referred as HelloForm is defined in the application.
- Only one action is defined and is invoked by requesting the application deployment path say, http://localhost/Hello World.
- When the action takes the control it assigns the HelloForm to the user request that validates the user inputs. If validation fails, the user should be sent to the input page /hello.jsp to correct the inputs.
- There is a MessageResource bundle associated with the application and stored in the Application.properties file.

This quickly completes a simple Struts application program, which gives you a complete view of the working of the framework that can be mapped to the real world applications easily.

Conclusion

Struts answer some of the big problems of web application development. This approach helps in code reusability and flexibility. By separating the problem into smaller components, you will be more likely to reuse the application in many situations. Moreover, Struts enabled page designers and Java developers to focus on what they do best. Struts are much more complex than a simple single JSP page, but for larger systems Struts actually helps manage the complexity. Reviewing the Struts framework can give you a better understanding of JSP and Servlets technologies, and how to combine them for your next Web project. When you become familiar with the Struts framework, it might become an indispensable part of your next Web project.

Reference # 35J-2005-12-04-01

References

1. Govind Seshadri, "Understanding JavaServer Pages Model 2 Architecture Exploring the MVC Design Pattern" at http://www.javaworld.com/, 1999
2. http://www-128.ibm.com/developerworks/websphere/techjournal/0302_fung/fung.html
3. http://www.ftponline.com
4. http://struts.apache.org
5. http://www.fawcette.com/javapro

Appendix
Open Source Frameworks

One of the critical problems faced under the software development process is creating and maintaining a web framework. People spend a lot of money and time on just maintaining the framework. This brings the evaluation of open source web framework.

Open source web frameworks available are well documented, well tested and make it much easier to develop web applications. In order to bring more structure and maintainability to web applications, many java developers have developed Web frameworks that simplify the problems found in the standard J2EE.

A java web framework simplifies writing Web applications and it is very easy to deploy and maintain. Framework helps developers to be more productive because setting up the architecture for the web tier is taken care of by the framework. It provides a set of reusable patterns that developers can use to solve the common problems.

Selecting a web framework is an important task because the labor is expensive and productivity is important. Currently, java has two types of web frameworks: Request-based and Component-based.

Request-based Frameworks:This framework embraces the servlet API and generally pushes the information to the user interface. User interface is responsible for grabbing this information and displaying it to the user. Reusing of code can be done using JSP Tag libraries.

Component-based Frameworks:This framework exists because of the ugliness behind the JSPs and the difficulty in the code reuse. These frameworks provide an API for developing reusable features that are easily packaged and used across the applications.

Framework	Description	URL	Type
Struts	Uses MVC pattern, Gives pre-built components to assemble applications that are flexible and scalable.	Request-based	http://struts.apache.org
Spring MVC	A component of Spring Framework provides simpler API for J2EE.	Request-based	http://www.springframework.org
WebWork	Easy to use, and powerful, simple API.	Request-based	http://opensymphony.com
Cocoon	Derived from an XML-based content publishing technology.	Request-based	http://cocoon.apache.org
JSF	JSF has specification behind it, Uses JSP for its template technology.	Component-based	http://myfaces.apache.org
Tapestry	Uses HTML for its template technology.	Component-based	http://jakarta.apache.org/tapestry