Servlets, JSP, Struts and MVC
(Part II)

Venkat Subramaniam
venkats@agiledeveloper.com
http://www.agiledeveloper.com/download.aspx

Abstract

In Part I, we discussed the Model I and Model II architecture. In this article, we show how the example presented in Part I may be implemented, more effectively, using the Struts framework. This article assume you have read Part I.

Installing Struts

I have downloaded and installed the file jakarta-struts-1.1-rc2.zip from the site http://apache.mirrors.pair.com/jakarta/struts/binaries/. I have extracted the contents of the above zip file and place it in c:\jakarta-struts-1.1-rc2 directory. Tomcat (4.1.24) has been installed on my system in c:\jakarta-tomcat-4.1.24 directory.

Setting up the Application

If you go to the C:\jakarta-struts-1.1-rc2\webapps directory, you will find a file named struts-blank.war. Simply make a copy of this file by right clicking on it and clicking on copy and then paste. Modify the name “Copy of struts-blank.war” to guess.war. Move the file guess.war to the following directory: C:\jakarta-tomcat-4.1.24\webapps. Now start tomcat by running the startup.bat in C:\jakarta-tomcat-4.1.24\bin. You will see that a directory named guess has been created under C:\jakarta-tomcat-4.1.24\webapps. This directory is almost ready for us to use for developing our struts application.

JSPs and Java Classes

From Part I, you may remember how the requests went directly to the servlets and the response came from the JSPs. The struts framework simply implements this pattern. Let’s first start examining the files we need to create.

Replace the file index.jsp in C:\jakarta-tomcat-4.1.24\webapps\guess with the following file:

```
<!-- index.jsp -->
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

<html:html>
<head>
<title>Guessing Game</title>
<html:base/>
</head>
<body bgcolor="white">

<html:errors/>
      <H1>Guessing Game<H1>
      <B>Welcome to Guessing Game</B>
      <%@ include file="GuessRequest.htm" %>
</body>
```

```
</html:html>
```

Let's take a closer look at a couple of things to note. First, we have taglib definition at the top that prefixes to the libraries to be used. Second, we use the <html:…> tags to generate HTML instead of directly writing the HTML tags. Notice also that the JSP page includes the contents of the GuessRequest.htm, the content of which is shown below:

```
<!-- GuessRequest.htm -->
    <html:form action="Guess.do" focus="guess">
            Enter your guess:
                  <html:text property="guess"/>
            <html:submit property="submit" value="Send"/>
    </html:form>
```

The GuessRequest.htm has a html:form tag that will generate the HTML tags for the FORM. Notice the focus attribute is used to indicate that when the page is displayed, the guess textbox should have focus. This is one of the several advantages of using the html tags provided in struts. The relevant html content is generated for us, the state of the controls is maintained and lesser code is needed to do that.

Note that the action attribute has a value "Guess.do." The extension "do" is not set on stone. It may be configured to what ever you would like it to be. There is a mapping under the hood (as we will see in the next section) that will route the request to the proper servlet. The rest of the details of the contents of this file are fairly obvious.

We have another jsp pages ContinueGuessing.jsp as shown below:

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

<jsp:useBean id="Info" scope="session"
            class="com.agiledeveloper.Info" />

<html:html>
<head>
<title>Guessing Game</title>
<html:base/>
</head>
<body bgcolor="white">

<html:errors/>
<H1>Guessing Game<H1>
Number of Attempts
<jsp:getProperty name="Info"  property="attempts" />
<BR>
<jsp:getProperty name="Info"  property="message" />
<BR>
<%@ include file="GuessRequest.htm" %>
</body>
</html:html>
```

This page accesses a bean named Info (of type com.agiledeveloper.Info) and displays the attempts property and the message property of this bean. It then includes the contents of the GuessRequest.html which we discussed above.

Finally, the last jsp page involved is the GameOver.jsp as shown below:

```
<-- GameOver.jsp -->
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

<jsp:useBean id="Info" scope="session"
            class="com.agiledeveloper.Info" />

<html:html>
<head>
<title>Guessing Game</title>
<html:base/>
</head>
<body bgcolor="white">

<html:errors/>
<H1>Guessing Game<H1>
Number of Attempts
<jsp:getProperty name="Info"  property="attempts" />
<BR>
<jsp:getProperty name="Info"  property="message" />
<BR>
Would you like to start a new game?

<form action="index.jsp">
      <html:submit property="submit" value="Yes"/>
      <html:submit property="submit" value="No"
                                    onclick="window.close();" />
</form>
</body>
</html:html>
```

I think you will find the purpose of this file and the details of its contents obvious.

The above JSP files are located in the C:\programs\jakarta-tomcat-4.1.24\webapps\guess directory.

Now, let us examine the Java classes we have. Under the C:\programs\jakarta-tomcat-4.1.24\webapps\guess\WEB-INF\src directory, we create a directory named com and under that we create a directory named agiledeveloper. In this directory we place three files GuessAction.java, GuessForm.java and Info.java.

Let’s first examine the Info.java.

```
//Info.java
package com.agiledeveloper;
```

```
public class Info
{
	private int attempts;
	private String message;

	public void setAttempts(int value)
	{
		attempts = value;
	}

	public int getAttempts() { return attempts; }

	public void setMessage(String value)
	{
		message = value;
	}

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

Info is simply a JavaBean with two properties: attempts and message. It has getters and setters for these two properties.

Let's now look at the GuessForm.java:

```
//GuessForm.java
package com.agiledeveloper;

import org.apache.struts.action.*;

public class GuessForm extends ActionForm
{
    private int guess;

    public void setGuess(int value)
    {
            guess = value;
    }

    public int getGuess()
    {
      return guess;
    }
}
```

GuessForm is a JavaBean which has one property guess. It is used to hold the content of the form that the user will fill in from the GuessRequest.htm (included in the two jsp pages as discussed above). The struts framework will take the data from the HTML form and put the information into a Form bean and send it to the action class for processing.

Finally, let's examine the GuessAction.java:

```
//GuessAction.java
package com.agiledeveloper;
```

```
import javax.servlet.http.*;
import org.apache.struts.action.*;

public class GuessAction extends Action
{
	public ActionForward execute(ActionMapping mapping,
		ActionForm form,

	HttpServletRequest request,

	HttpServletResponse response)
	{
		ActionForward forward = null;

		if (form instanceof GuessForm)
		{
			HttpSession session = request.getSession();

			// If this is new request, target will not be present
			int target = 0;
			try
			{
				target =
					Integer.parseInt(
						session.getAttribute("target").
						toString());
			}
			catch(Exception ex)
			{
				int newTarget = (int)(Math.random() * 100);
				session.setAttribute("target", "" + newTarget);
				session.setAttribute("attempts", "1");
			}

			int attempts = 1;
			int guess = -1;
			String forwardPage = "Continue";
			String message = "Aim higher";

			target =
				Integer.parseInt(
						session.getAttribute("target").
						toString());
			attempts =
				Integer.parseInt(
					session.getAttribute("attempts").
					toString());
			attempts++;
			session.setAttribute("attempts", "" + attempts);

			GuessForm guessForm = (GuessForm) form;
			guess = guessForm.getGuess();

			if (guess == target)
			{
				forwardPage = "ItsOver";
				message = "Congratulations!";
```

```
                        session.invalidate();
                        session = request.getSession();
                    }
                    else
                    {
                        if (guess < target)
                            message = "Aim higher!";
                        else
                            message = "Aim lower!";
                    }

                    Info info = new Info();
                    info.setAttempts(attempts);
                    info.setMessage(message);
                    session.setAttribute("Info", info);

                    forward = mapping.findForward(forwardPage);

                }

                return forward;
        }
}
```

Note that this class inherits from org.apache.struts.action.Action class. It overrides the execute method. The execute method does two main things. First, it processes the request received. Second, it directs the framework to forward the control to one of the jsp pages depending on the condition. This page, to forward control to, is specified as part of the returned object of type ActionForward. Note that we do not actually have the names of the jsp pages in this file. Instead, the forwardPage variable is set to either "Continue" or "ItsOver."

Looking at this code you may have a couple of questions.

1. How in the world does Struts know that Guess.do should map over to GuessAction.java?
2. How does it know that if forwardPage is "Continue," it should transfer control to ContinueGuessing.jsp and if it is "ItsOver," it should transfer control to GameOver.jsp.

The answer is simple. It does not know at this point. There is no magic. We need to provide this information to the struts framework in a configuration files!

Configuring the Forms and Forwards

We will be making two changes to the file struts-config.xml that you can find under the directory C:\jakarta-tomcat-4.1.24\webapps\guess\WEB-INF.

First look for the element <form-bean> and add the following as the child of that element:

```
<form-bean name="GuessForm" type="com.agiledeveloper.GuessForm" />
```

After you add this line, the partial content of the file would look as shown below:

```
<!-- ========================================== Form Bean Definitions -->


    <form-beans>
     <form-bean name="GuessForm" type="com.agiledeveloper.GuessForm" />


    <!-- sample form bean descriptor for an ActionForm

        <form-bean

            name="inputForm"

            type="app.InputForm"/>

    end sample -->


    <!-- sample form bean descriptor for a DynaActionForm
```

Analyze the line we just added. We are telling the framework that the bean named GuessForm is an object of type com.agiledeveloper.GuessForm. This is simply introducing the form bean to the framework.

The second change is to look for the element <action-mapping> element and add the following as the child of that element:

```
<action path="/Guess" type="com.agiledeveloper.GuessAction"
      name="GuessForm" input="index.jsp">
            <forward name="Continue" path="/ContinueGuessing.jsp" />
            <forward name="ItsOver" path="/GameOver.jsp" />
</action>
```

After you add this line, the partial content of the file would look as shown below:

```
<!-- ================================== Action Mapping Definitions -->


    <action-mappings>
      <action path="/Guess" type="com.agiledeveloper.GuessAction"
            name="GuessForm" input="index.jsp">
            <forward name="Continue" path="/ContinueGuessing.jsp" />
            <forward name="ItsOver" path="/GameOver.jsp" />
      </action>

            <!-- Default "Welcome" action -->

            <!-- Forwards to Welcome.jsp -->
```

```
        <action
            path="/Welcome"
            type="org.apache.struts.actions.ForwardAction"
            parameter="/pages/Welcome.jsp"/>


    <!-- sample input and input submit actions
```

Let's analyze this element we just added. Look only at the action element tag along with the attributes in it. The path attribute says that Guess (by which we mean Guess.do) should map over to com.agiledeveloper.GuessAction servlet (the one that inherits from the Action class). The form bean that should be sent to the Action's execute method is GuessForm. The input attribute (with value of "index.jsp") represents the form to which the control should be transferred should an error occur in validating the input (i.e., before the request could be processed). Now look at the children of the action element. This is where the conditions are mapped to the pages to which the controls need to be transferred to. Note that "Continue" is mapped to the ContinueGuessing.jsp and "ItsOver" is mapped to the GameOver.jsp. The `mapping.findForward(forwardPage);` within the execute method of the GuessAction uses this information to decide which page it should route the flow of control to.

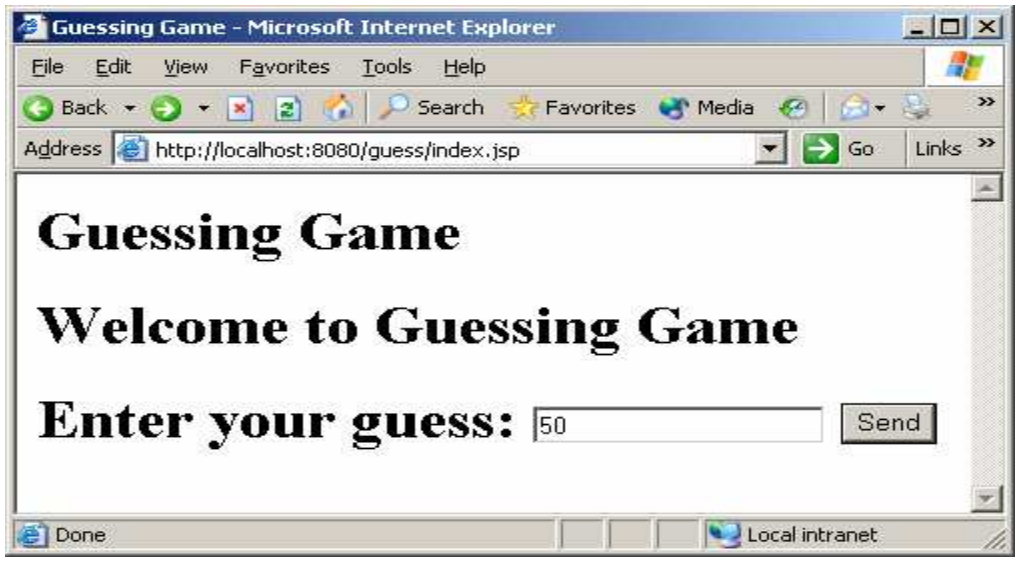
Getting the App running

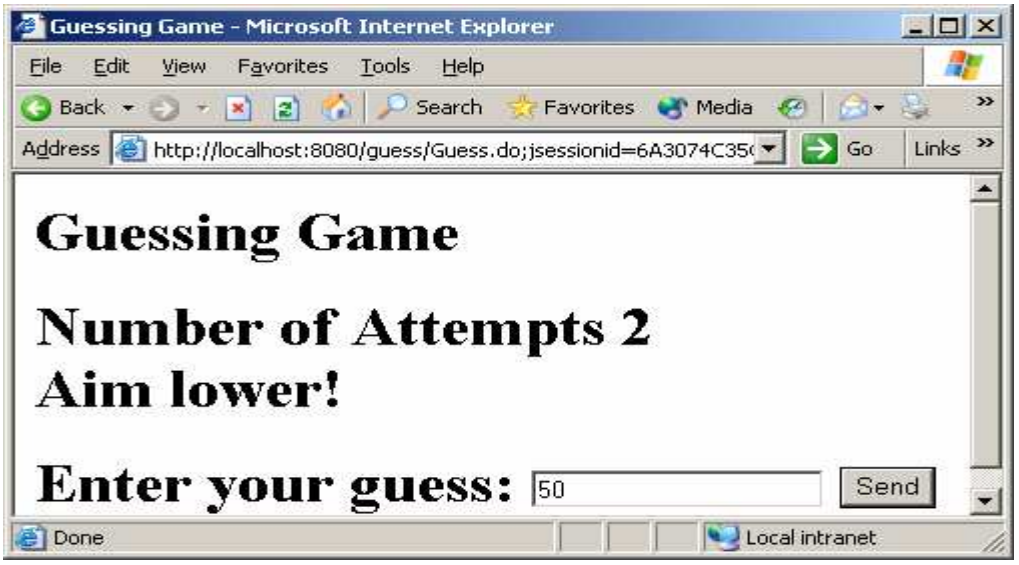
Now cd to the directory where the source code for the three classes discussed above are located, i.e., C:\jakarta-tomcat-4.1.24\webapps\guess\WEB-INF\src\com\agiledeveloper. You may use Ant to build your code or simply compile it using the following commands:

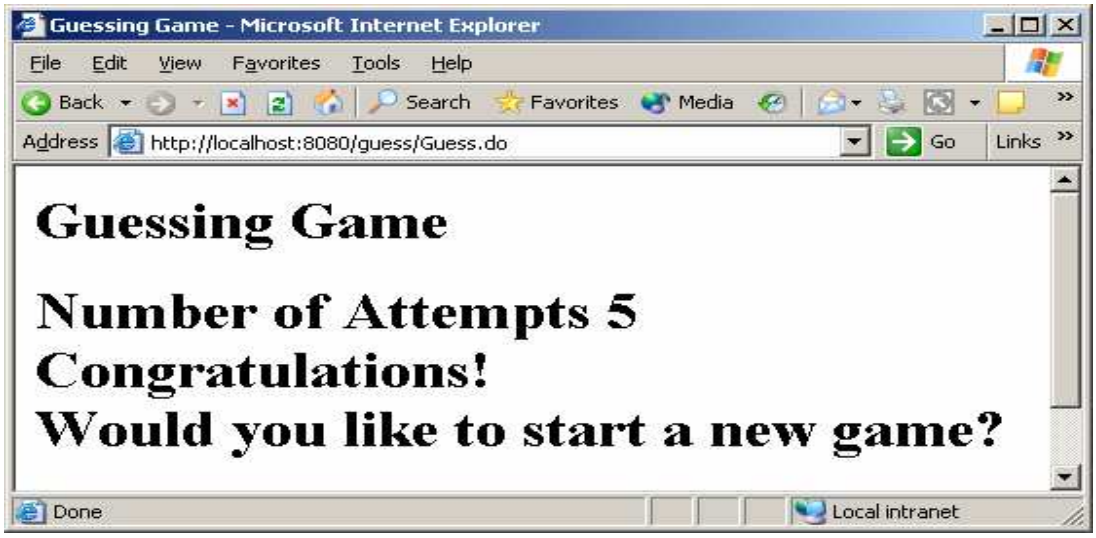
```
set cp = C:\jakarta-tomcat-4.1.24\webapps\guess\WEB-INF\lib\struts.jar
set cp=%cp%;C:\jakarta-tomcat-4.1.24\common\lib\servlet.jar
set classpath=%classpath%;cp
javac -d ..\..\..\classes *.java
```

This creates, under C:\jakarta-tomcat-4.1.24\webapps\guess\WEB-INF\classes, the directory named com. Under that directory, it creates directory named agiledeveloper and places the three .class files in it.

Now start (or restart) tomcat. Visit http://localhost:8080/guess and try the application.


Guessing Game - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/guess/index.jsp
Go Links
Guessing Game
Welcome to Guessing Game
Enter your guess: 50
Send
Done
Local intranet


Guessing Game - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/guess/Guess.do;jsessionid=6A3074C35
Go Links
Guessing Game
Number of Attempts 2
Aim lower!
Enter your guess: 50
Send
Done
Local intranet


Guessing Game - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/guess/Guess.do
Go Links
Guessing Game
Number of Attempts 5
Congratulations!
Would you like to start a new game?
Done
Local intranet

A look at the Struts classes used

The Action class (of the org.apache.struts package) is an adapter or a liaison between the incoming HTTP request and the business logic (control code). An instance of another class, ActionServlet, does the actual job of receiving the request, creating an instance of Action class (if needed) and invoking the execute method. In Struts 1.1, the old perform method has been deprecated and a new method named execute has been introduced. Note that one instance of the Action class is shared for multiple requests. So, write it in a thread safe manner and do not put any instance or static variables in your Action class implementation.

The purpose of the ActionForm JavaBean is to encapsulate the data that goes from the user input (form) to be processed by the execute method of the Action. Before the execute method is actually invoked, the ActionForm bean's validate method is invoked by the framework, giving it an opportunity to validate the input.

ActionForward represents the destination to which the ActionServlet will transfer control over to. The value for the ActionForward comes from one of several entries in the configuration file as shown in the above example.

ActionMapping inherits from ActionConfig (as of Struts 1.1). It provides a mapping of the request to an instance of an action class; and it provides access to the configuration information for action mapping.

Conclusion

In Part I of this article, we presented an example and discussed the Model I and Model II architecture. This article (Part II), implemented that example using the Struts framework. While we have merely scratched the surface of the framework, we hope this will wet your appetite to delve into it.

References

1. http://jakarta.apache.org/tomcat/index.html.
2. http://jakarta.apache.org/struts