Struts 1.2

Day 13 : Monday
Created Date : 03/09/2012
Created By : Alok Kumar Vishwakarma
Mentor : Satheesh Balasubramanian

- The MVC Design Pattern
- Framework control flow
- Configure the Struts Tag Libraries
- Sample Calculator Application in Struts1.2.8

The MVC Design Pattern

The term "MVC" originated with the SmallTalk Model-View-Controller framework. Under MVC, an application is seen as having three distinct parts. The problem domain is represented by the Model. The output to the user is represented by the View. And, the input from the user is represented by Controller.

Framework control flow

The framework provides several components that make up the **Control** layer of a MVC-style application. These include a controller component (servlet), developer-defined request handlers, and several supporting objects.
The Struts Taglib component provides direct support for the **View** layer of a MVC application. Some of these tags access the control-layer objects. Others are generic tags found convenient when writing applications. Other taglibs, including JSTL, can also be used with the framework. Other presentation technologies, like Velocity Templates and XSLT can also be used with the framework.
The **Model** layer in a MVC application is often project-specific. The framework is designed to make it easy to access the business-end of your application, but leaves that part of the programming to other products, like JDBC, Enterprise JavaBeans,Object Relational Bridge, or iBATIS, to name a few.
Let's step through how this all fits together.
When initialized, the controller parses a configuration file (struts-config.xml) and uses it to deploy other control layer objects. Together, these objects form the **Struts Configuration.** The Configuration defines (among other things) the collection of ActionMappings[org.apache.struts.action.ActionMappings] for an application.
The controller component consults the ActionMappings as it routes HTTP requests to other components in the framework. Requests may be forwarded to JavaServer Pages

or Action[org.apache.struts.action.Action]subclasses provided by the application developer. Often, a request is first forwarded to an Action and then to a JSP (or other presentation page). The mappings help the controller turn HTTP requests into application actions.

An individual ActionMapping[org.apache.struts.action.ActionMapping] will usually contain a number of properties including:

- a **request path** (or "URI"),
- the **object type** (Action subclass) to act upon the request, and
- other properties as needed.

The Action object can handle the request and respond to the client (usually a Web browser) or indicate that control should be forwarded elsewhere. For example, if a login succeeds, a login action may wish to forward the request onto the main Menu page.

Action objects have access to the application's controller component, and so have access to that members's methods. When forwarding control, an Action object can indirectly forward one or more shared objects, including JavaBeans, by placing them in one of the standard contexts shared by Java Servlets.

For example, an Action object can create a shopping cart bean, add an item to the cart, place the bean in the session context, and then forward control to another mapping. That mapping may use a JavaServer Page to display the contents of the user's cart. Since each client has their own session, they will each also have their own shopping cart.

Most of the business logic in an application can be represented using JavaBeans. An Action can call the properties of a JavaBean without knowing how it actually works. This encapsulates the business logic, so that the Action can focus on error handling and where to forward control.

JavaBeans can also be used to manage input forms. A key problem in designing Web applications is retaining and validating what a user has entered between requests. You can define your own set of input bean classes, by subclassing ActionForm[org.apache.struts.action.ActionForm]. The ActionForm class makes it easy to store **and validate** the data for your application's input forms. The ActionForm bean is automatically saved in one of the standard, shared context collections, so that it can be used by other objects, like an Action object or another JSP.

The form bean can be used by a JSP to collect data from the user ... by an Action object to validate the user-entered data ... and then by the JSP again to re-populate the form fields. In the case of validation errors, the framework has a shared mechanism for raising and displaying error messages.

Another element of the Configuration are the ActionFormBeans[org.apache.struts.action.ActionFormBeans.This is a collection of descriptor objects that are used to create instances of the ActionForm objects at runtime. When a mapping needs an ActionForm, the servlet looks up the form-bean descriptor by name and uses it to create an ActionForm instance of the specified type.

Here is the sequence of events that occur when a request calls for an mapping that

uses an ActionForm:

- The controller servlet either retrieves or creates the ActionForm bean instance.
- The controller servlet passes the bean to the Action object.
- If the request is being used to submit an input page, the Action object can examine the data. If necessary, the data can be sent back to the input form along with a list of messages to display on the page. Otherwise the data can be passed along to the business tier.
- If the request is being used to create an input page, the Action object can populate the bean with any data that the input page might need.

The Struts Taglib component provides custom tags that can automatically populate fields from a JavaBean. All most JavaServer Pages really need to know is the field names to use and where to submit the form.

Other tags can automatically output messages queued by an Action or ActionForm and simply need to be integrated into the page's markup. The messages are designed for localization and will render the best available message for a user's locale.

The framework and Struts Taglib were designed from the ground-up to support the internationalization features built into the Java platform. All the field labels and messages can be retrieved from a message resource. To provide messages for another language, simply add another file to the resource bundle.

Internationalism aside, other benefits to the message resources approach are consistent labeling between forms, and the ability to review all labels and messages from a central location.

For the simplest applications, an Action object may sometimes handle the business logic associated with a request.**However, in most cases, an Action object should invoke another object, usually a JavaBean, to perform the actual business logic.** This lets the Action focus on error handling and control flow, rather than business logic. To allow reuse on other platforms, business-logic JavaBeans should not refer to any Web application objects. The Action object should translate needed details from the HTTP request and pass those along to the business-logic beans as regular Java variables.

In a database application, for example:

- A business-logic bean will connect to and query the database,
- The business-logic bean returns the result to the Action,
- The Action stores the result in a form bean in the request,
- The JavaServer Page displays the result in a HTML form.

Neither the Action nor the JSP need to know (or care) from where the result comes. They just need to know how to package and display it.

Other sections in this document cover the various framework components in greater detail. The Struts Taglib component includes several Developer Guides covering various aspects of the custom tags. A number of sample applications are bundled with the distribution that show how it all comes together.

Configure the Struts Tag Libraries

1. Strut Tag Libraries Manual Configuration
2. Strut Tag Libraries Automatic Configuration

Strut Tag Libraries Manual Configuration

The manual configuration is the old and classic way, used in **Struts version <= 1.1 and Servlet < 2.3** container. Download all the Struts dependencies, make sure the following "**tld**" files are copy to **WEB-INF** folder, you can find these files in the downloaded Struts library.

- struts-bean.tld
- struts-html.tld
- struts-logic.tld
- struts-tiles.tld

Declare the taglib uri in web.xml

web.xml

```
...
<taglib>
    <taglib-uri>
        http://struts.apache.org/tags-bean
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-bean.tld
    </taglib-location>
</taglib>
<taglib>
    <taglib-uri>
        http://struts.apache.org/tags-html
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-html.tld
    </taglib-location>
</taglib>
<taglib>
    <taglib-uri>
        http://struts.apache.org/tags-logic
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-logic.tld
```

```
    </taglib-location>
</taglib>
<taglib>
    <taglib-uri>
        http://struts.apache.org/tags-tiles
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-tiles.tld
    </taglib-location>
</taglib>
<taglib>
    <taglib-uri>
        http://struts.apache.org/tags-nested
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-nested.tld
    </taglib-location>
</taglib>
...
```

Now we can access it in JSP page. The JSP's **@taglib uri** have to match with web.xml **<taglib-uri>**

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>
```

we can define your own **taglib uri** name, for example

web.xml

```
...
<taglib>
    <taglib-uri>
        customer-anything/tags-bean
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-bean.tld
    </taglib-location>
</taglib>
...
```

Then access it via your custom **taglib uri** name.

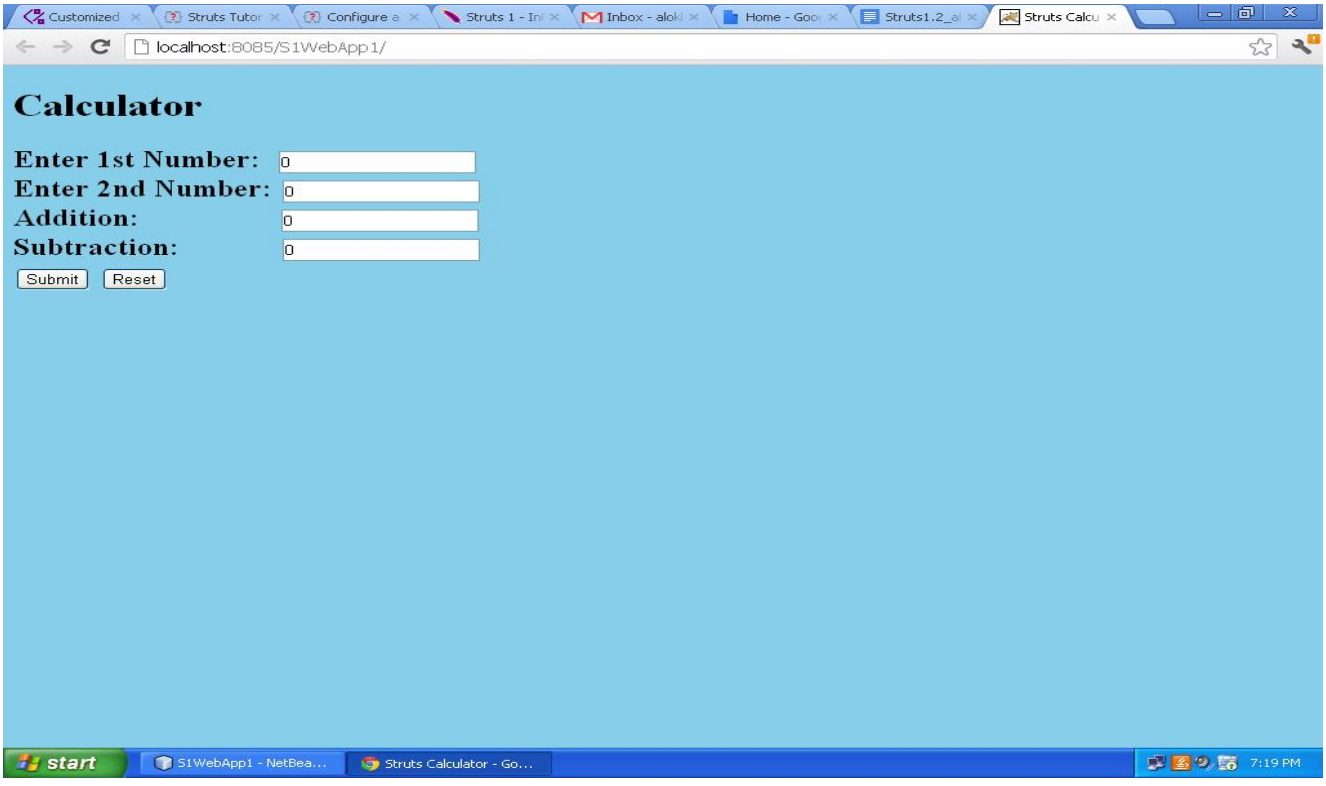
<%@ taglib uri="customer-anything/tags-bean" prefix="bean" %>

2. Strut Tag Libraries Automatic Configuration

This is the easy way, and used in **Struts version 1.2, 1.3 and Servlet 2.3/2.4 container** only. You do not need to define the "**tlds**" details in web.xml anymore, just include the **struts-taglib.jar** in your project classpath or copy it to WEB-INF/lib folder.
All the "**tld**" details are define inside the "**struts-taglib.jar\META-INF\tld**" folder. During deployment, the **struts-bean.tld**,**struts-html.tld**, **struts-logic.tld** and **struts-tiles.tld** will deploy automatically. However, you can access it via the following "**pre-fixed uri**" name only. In this method, you are not allow to change the "**taglib uri**" name.

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>

Sample Calculator Application in Struts1.2.8



Source Code:

index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@ taglib prefix="html" uri="http://jakarta.apache.org/struts/tags-html" %>
<html>
   <head>
      <title>Struts Calculator</title>
        </head>
   <body bgcolor="skyblue">
      <html:errors/>
      <html:form action="calcu">
         <h1>Calculator</h1>
         <h2>
            Enter 1st Number:  <html:text property="n1"/><br/>
            Enter 2nd Number: <html:text property="n2"/><br/>

Addition:           
                                           <html:text
property="addition"/><br/>

Subtraction:           
             
            <html:text property="subtraction"/><br/>
            <html:submit onclick="validate();"/>
            <html:reset onclick="clear();"/>
         </h2>
      </html:form>
   </body>
</html>
```

welcomeStruts.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

<html:html locale="true">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title><bean:message key="welcome.title"/></title>
      <html:base/>
   </head>
   <body style="background-color: white">

                              <logic:notPresent     name="org.apache.struts.action.MESSAGE"
scope="application">
         <div  style="color: red">
            ERROR:  Application resources not loaded -- check servlet container
            logs for error messages.
         </div>
      </logic:notPresent>

      <h3><bean:message key="welcome.heading"/></h3>
      <p><bean:message key="welcome.message"/></p>

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

struts-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
        "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">


<struts-config>
   <form-beans>
      <form-bean name="CalcForm" type="calcu.CalcForm"/>
   </form-beans>

   <global-exceptions>

   </global-exceptions>

   <global-forwards>
      <forward name="welcome"  path="/Welcome.do"/>
   </global-forwards>

   <action-mappings>
      <action path="/calcu" type="calcu.MyAction" name="CalcForm" input="/index.jsp">
         <forward name="success" path="/index.jsp"/>
      </action>
   </action-mappings>

   <controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>

   <message-resources parameter="com/myapp/struts/ApplicationResource"/>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>2</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
    </servlet>
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
  <jsp-config>
    <taglib>
      <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
      <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
    </taglib>
    <taglib>
```

```
        <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
      </taglib>
      <taglib>
        <taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
      </taglib>
      <taglib>
        <taglib-uri>/WEB-INF/struts-nested.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-nested.tld</taglib-location>
      </taglib>
      <taglib>
        <taglib-uri>/WEB-INF/struts-tiles.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-tiles.tld</taglib-location>
      </taglib>
      </jsp-config>
   </web-app>
```

CalcForm.java

```
package calcu;

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 class CalcForm extends ActionForm
{
   private int n1,n2,addition,subtraction;
   public void setN1(int n1)
   {
      this.n1=n1;
   }
   public int getN1()
   {
      return n1;
   }
   public void setN2(int n2)
   {
      this.n2=n2;
```

```
    }
    public int getN2()
    {
        return n2;
    }
    public void setAddition(int addition)
    {
        this.addition=addition;
    }
    public int getAddition()
    {
        return addition;
    }
    public void setSubtraction(int subtraction)
    {
        this.subtraction=subtraction;
    }
    public int getSubtraction()
    {
        return subtraction;
    }
    public ActionErrors validate(ActionMapping mapping,HttpServletRequest request)
    {
        ActionErrors errors=new ActionErrors();
        if(n1==0)
        {
            errors.add("error.n1",new ActionError("error.n1"));
        }
        if(n2==0)
        {
            errors.add("error.n2",new ActionError("error.n2"));
        }
        return errors;
    }
}
```

MyAction.java

```
package calcu;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class MyAction extends Action
{
   @Override
                public ActionForward execute(ActionMapping mapping,ActionForm
form,HttpServletRequest request,HttpServletResponse response)
   {
      CalcForm calcForm=(CalcForm)form;
      int n1=calcForm.getN1();
      int n2=calcForm.getN2();
      calcForm.setAddition(n1+n2);
      calcForm.setSubtraction(n1-n2);
      return mapping.findForward("success");
   }
}
```