The Struts Application Framework

Strut Your Stuff!

Mark Bennett
Craig Lonsbury
Gemma Mio

The Struts Application Framework

Introduction

Java Server Pages and Java Servlets are two exiting technologies that enable developers to bring the power and flexibility of the Java platform to the web. By using JSP and Servlets, developers are able to offer exciting and complex applications to their users without relying on them to provide anything more than a web browser. As well, Servlets are becoming an important component of the developing Java SOAP web services model.

Developing web applications in any language can be difficult, and Java is no exception. However, the Struts Framework can help. By providing basic support for many common elements of a web application, it empowers developers to create more reliable applications in a more timely fashion than ever before. This includes support for everything from server- and client-side input validation to database connection pooling, internationalization and the creation of dynamic page templates. As well, it helps separate page content from page layout, so that graphic artists and designers can work to develop attractive pages, while developers work to create the business logic that drives the application, without interfering with each other.

Struts itself is best explained as three different things: An extension to the standard Java Server Context; A framework within which to develop web applications; and, a set of tag libraries for use with Java Server Pages. The rest of this report will serve to describe Struts from each of these different perspectives.

What is Struts?

An Extension

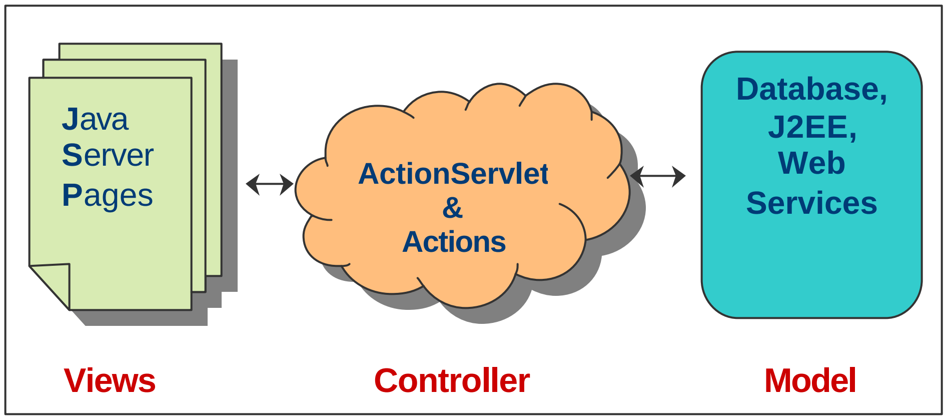
The Struts Framework is an extension of the standard Java Server Context (JSC). Struts is not meant to be used outside the context of a Java Server such as Tomcat, WebSphere, or iPlanet. The Struts package itself consists of only one small .jar archive and an xml configuration file, as well as a number of optional tag library descriptors with are meant to be deployed in your applications /WEB-INF/lib and /WEB-INF directories, respectively. Other features such as support for database pooling may require that the JDBC drivers for your chosen database, or other software, be installed.

As well, JSP's and Servlets may exist within this context without using Struts. This means that you may choose to use Struts when deploying part of your application, but provide an in-house or third-party solution for the rest. It's up to you, and it's always easy to transition your code to use Struts at a later date.

A Framework

Now that it's clear what Struts is in relation to the JSE, you may be curious as to what exactly it does? The simple answer is that it provides a framework which developers and designers may use to aid them when crafting their web applications.

So how does it do that? Well, first of it starts by providing the developers with the components of a Model / Controller / View architecture. These components encourage the creation of a three tier application where the data and the actions available upon that data (the Model) are separated from the input and output presented to the user (the View) by an object or set of objects that broker the interaction between the two (the Controller).

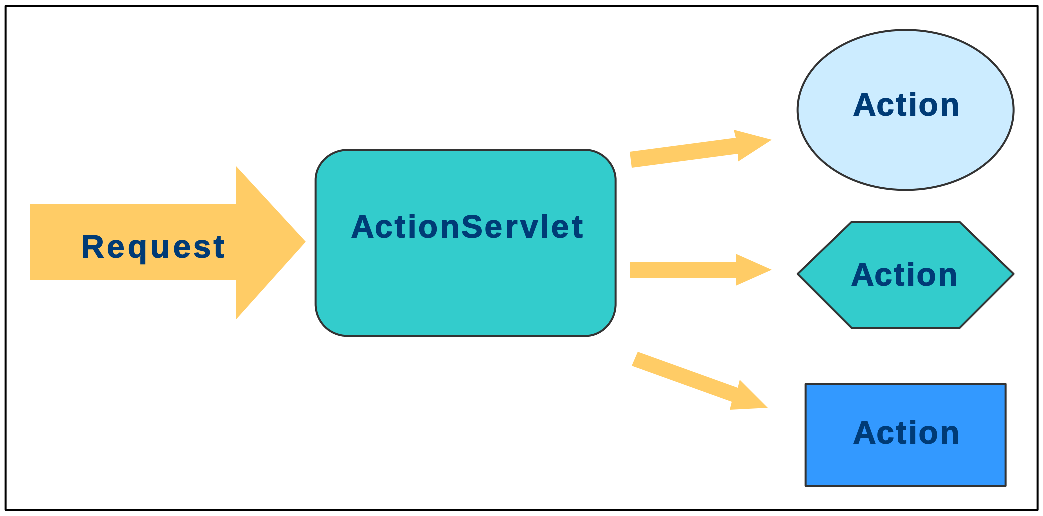


The Model is committed to maintaining the integrity of its data and therefore will not allow the Controller to access its internal data structures. In and of itself however, the Model cannot successfully complete the actions requested by the user, and relies upon the Controller to direct it in the sequence and nature of its actions. Finally, the View collects the input from the user and turns this into data and requests for actions to be sent to the Controller. As well, the View is responsible for displaying the current state of the model as well as passing any messages received from the Controller to the client. These messages are usually errors, though in the case of larger transactions they may also be confirmations that an action has been completed successfully.

In the Struts framework the Model, Controller, and View tiers all map directly to various parts of the JSE and the objects provided by the Struts framework. The Model is usually represented by the DBMS, J2EE server, a legacy system, and various web services or any combination of these. The Controller is represented by something known as the ActionServlet, as well as its corresponding Action objects. The View is represented by JSP's and a new type of bean called a FormBean.

To elaborate, perhaps it would be simplest to illustrate the flow of control through the web application as an action is requested by the client and is processed by the rest of the Struts Framework.

A client first initiates an action by sending a request to a Java Server Context. This request's path must be of a form that matches the mapping assigned to the ActionServlet. Normally this usually involves mapping all requests for resources ending with "*.do" to the ActionServlet. This Servlet is a class provided by Struts and is an instance of the **org.apache.struts.action.ActionServlet** class. It is loaded when a web application is initialized and reads in the "struts-config.xml" configuration file. This file specifies the actions that this ActionServlet understands, as well as various other Struts parameters. The ActionServlet receives the request, and looks at the path to determine the action that it is associated with. The ActionServlet then forwards the request and response to an instance of the class associated with this action and prepares to receive future requests. All classes associated with an action must extend the **org.apache.struts.action.Action** class. This provides a number of helpful methods as well as a number of methods required by the ActionServlet. When programming your own actions, you need to keep in mind that they are multi-threaded and must therefore be thread-safe.



The actions themselves are where much of the real work gets done. Each action may be associated with a FormBean. FormBeans are created by the developer and must extend the **org.apache.struts.action.ActionFormBean** class. The FormBean works with a corresponding JSP to collect input from the client on the actions behalf. For each input on the JSP form, the FormBean should have a corresponding bean property. When the user submits the JSP form, the values from the form are mapped to their corresponding bean properties and used to populate the FormBean.

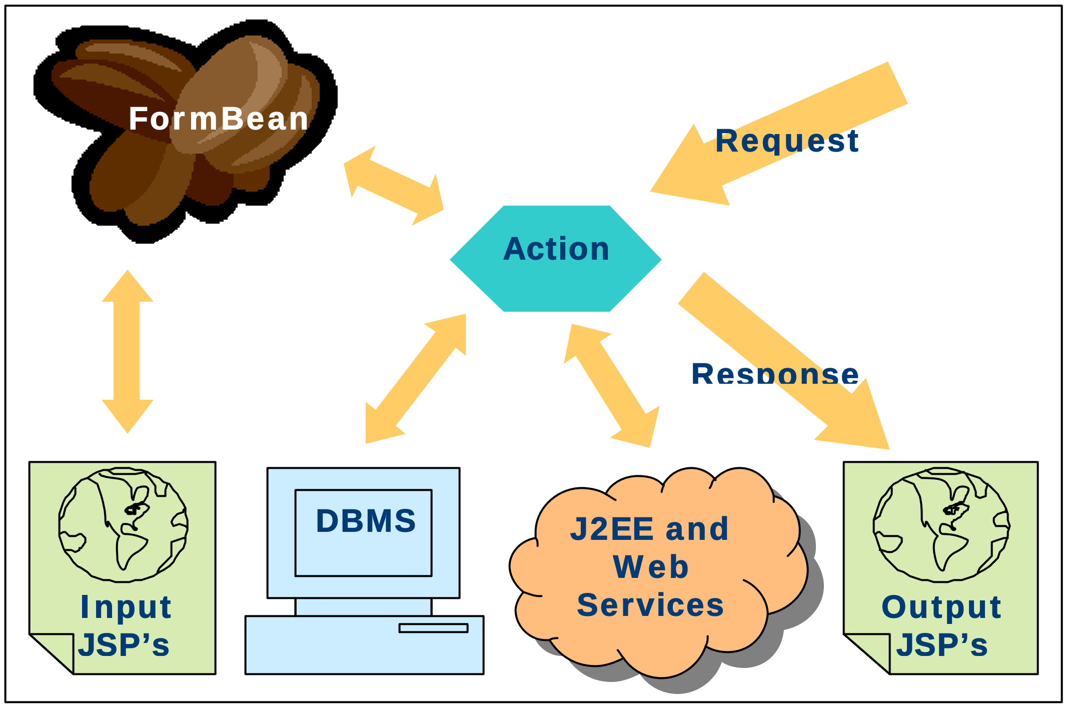
FormBean's are able to validate the input from the user based on its data type and value. If a value collected does not correspond with the correct data type, the web client is re-directed back to the JSP form to enter a new value. The action associated with this bean is responsible for testing the collected values based on their relationship with the Model and the business logic of the application. This means testing for required values, as well as verifying that users inputs fall within the constraints of the business logic. By maintaining this abstraction, the same FormBean class may be used with multiple actions.

There is also a standard method for adding cancel buttons to forms, which allows the web client to cancel an action if they cannot provide satisfactory input while still allowing the application the opportunity to respond appropriately.

When a users input has been validated, the action may begin attempting to satisfy the action request. This can involve interacting with J2EE, DMBS, XML or other web services. In fact, this may involve anything that can be preformed using J2SE. On smaller applications it is normally acceptable to place business logic directly into the action, although for larger distributed applications the use of J2EE, CORBA, or web services is recommended. Doing so will improve reliability, scalability, and flexibility. Under no circumstance should business logic be placed in the Java Server Pages. This will allow web page developers to freely design attractive websites, without accidentally disrupting the performance of the web application.

When the action is completed, the web client is forwarded to a JSP where the results of the action are available. This JSP may be different depending upon the results of the action.

If at any point in this process an error occurs, such as a bad user input, an inaccessible resource, or an exception, there is a standard and application wide error handling mechanism in place. An error may be stored as an **org.apache.struts.action.ActionError,** which is then stored in an **org.apache.struts.action.ActionErrors** object specific to that user session. JSP's, Servlets, and actions may then retrieve these errors based on name and type. A similar mechanism also exists for the passing of messages.



Besides offering this workflow, Struts also provides standard support for database connection pooling and internationalization of applications.

A Collection of Tags

Struts also includes a set of tag libraries intended to help the developer with a number of common tasks as well as to assist in integrating JSP with the Struts Framework. They are designed especially to facilitate the separation of presentation and business logic and to implement the flow of control to and from the **ActionServlet**.

HTML Tag Library

The first of these collections is the HTML tag library. This library includes tags designed to help create dynamic HTML user interfaces. This includes tags for rendering browser specific JavaScript and tags to aid with URL rewriting and session tracking.

As well this library provides an important set of tags that support mapping form inputs to FormBeans. Though it is possible to accomplish this without the use of the Struts HTML form element tags, they do greatly reduce the time and effort involved.

Another important use for the HTML tag library is linking. Links between pages can be created in both hyperlink and anchor formats, which is especially useful for generic forwards since it does not require the creation of heavier objects.

Finally internationalization and regionalization is supported by HTML tags that respond and encode based up the client encoding, and by allowing error messages and image tags to be retrieved from locale specific property files, as per the Java internationalization specification.

Bean Tag Library

The bean tag library encapsulates most of the features required to display both static and dynamic web pages. It provides tags for working with, creating, and removing beans. The bean tags help to define new beans from any objects associated with the current request and make them accessible to the remainder of the page via scripting variables or scope attributes.

As well these tags can help render bean, or bean properties, to the output response. This second set of tags is particularly useful as they aid in internationalization by allowing messages to be retrieved from a locale specific user file.

Finally, the bean tag library provides significant enhancements to the basic JSP *<jsp:useBean>* tag by extending three methods of referencing bean properties: simple, nested, and indexed.

Logic Tag Library

The logic tag library performs all the first and second order logic functions and thus provides conditional code generation. As well it includes tags for collection iteration and application flow management.

It should be noted that, while most binary logic tags are available, Struts does not provide some of the more complex String operations. Nevertheless, developers can implement their own custom tags, which are merely Java classes that implement special interfaces.

Template Tag Library

Finally the template tag library is used for creating dynamic JSP templates. This library provides the dynamic capabilities similar to those static compatibilities from stylesheets or the standard JSP `include` directive. Though the least used of the tag libraries, it is helpful when a developer wishes to be able to change layout and content across multiple pages. It is often used in the development of new web applications when such things may change frequently.

Where is Struts going?

With all of this already in Struts, there are still places it needs to go in the future. Already new releases are under development to include tag support for XML and XSLT. However, these new tags are likely not to be included into later stable releases as Sun has now incorporated many of Struts tag libraries into their new Java Server Pages Standard Tag Libraries (http://jcp.org/aboutJava/communityprocess/review/jsr052/index.html). These will include tag libraries to handle iteration, flow control, XML, DBMS, and internationalization amongst other things.

One area of future Struts growth is in new input form elements. It is hoped that there will soon be grid inputs, for spreadsheet style entry. This is currently possible but more difficult than it needs to be. As well, the select input elements will likely be reworked to provide a standard way for a FormBean or action to populate the selections list.

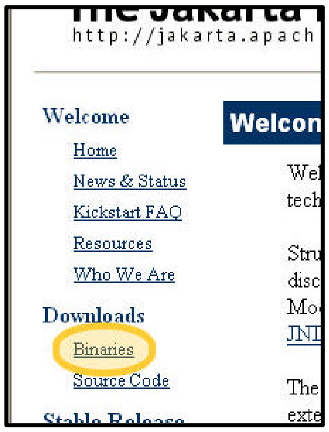
One other feature that is always in demand is support for multi-part, “Wizard” style input forms. There are multi-part forms similar to the “Wizards” available in many popular programming API’s. It is currently possible to implement simple multi-part forms, though more complex “Wizards” require the ability to change the forms that are displayed based upon different user input.

It should be noted however that many of these functions are already available as add-ons to the Struts Framework. For a list of many of the most popular, please visit http://jakarta.apache.org/struts/doc-1.0.2/userGuide/resources.html.

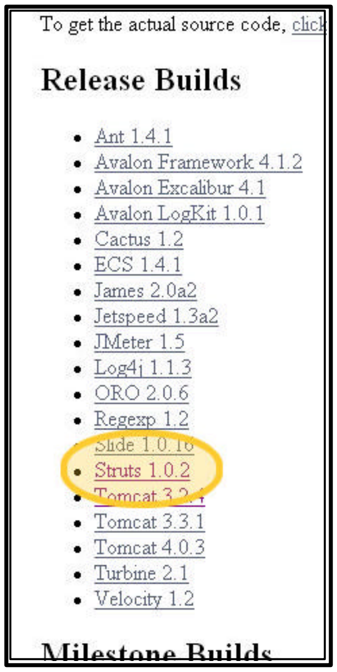
Getting started

Now that you have seen what the Struts Framework is all about, you might be wondering how to get started using it.

The first thing you want to do is to visit the official Struts homepage at http://jakarta.apache.org/struts. From there you can download a copy of Struts. I’d also recommend taking a minute to read their User Guide (http://jakarta.apache.org/struts/doc-1.0.2/userGuide/index.html) if you have a minute as it provides specifics on configuring Struts to perform the way you want it.



At the time of this writing, the current production release of Struts is version 1.0.2. To install it, first visit the Struts homepage. Select the 'Binaries' link in the 'Download' section. Now click on 'Struts 1.0.2' from the 'Release Builds' section and then select the distribution appropriate for your operating system and development requirements. For these instructions, it will be assumed that you will be developing on a Microsoft Windows operating system, however the directions should differ only slightly on other platforms. In this case, we want to download "jakarta-struts-1.0.2.zip". This file includes example applications as well as documentation, if you are simply interested in the core Struts libraries then download the "jakarta-struts-1.0.2-lib.zip" file from the "/lib" directory.



Once this file has been downloaded, extract its contents to a convenient location on your computer's hard-disk such as the desktop. WinZip (www.winzip.com) is a popular utility for this purpose though Windows ME and XP include support for the many archives without requiring the installation of third-party software.

To enable your web application to use Struts start by copying a number of files from the extracted contents of the Struts archive. First copy the "struts.jar" file from the "/lib" directory to the "/WEB-INF/lib" directory of your web application. Now copy the "struts-bean.tld", "struts-html.tld", "struts-logic.tld", and "struts-template.tld" tag library descriptor files from the "/lib" directory of the archive to the "/WEB-INF" directory of your web application.

Next we want to create the "struts-config.xml" file. This is usually located in the "/WEB-INF" folder of your application. This file must be an XML document that conforms to the "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd" DTD. An example is as follows:

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

<!--Configuration file for the Struts Framework-->


<!DOCTYPE struts-config  PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration
1.0//EN"
          "http://jakarta.apache.org/struts/dtds/struts-
config_1_0.dtd">
```

```
<struts-config>
  <!--Data sources allow your application to utilize connection pooling
and help abstract the DBMS from the application.-->
  <!--
    <data-sources id="name">
    <data-source autoCommit="false" description="Example Data Source
Description" driverClass="org.postgresql.Driver" maxCount="4"
minCount="2" password="mypassword"
url="jdbc:postgresql://localhost/mydatabase" user="myusername"/>
  </data-sources>
  -->
  <!--Form beans are used by actions when collecting input-->
    <form-beans>
    <form-bean name="logonForm"
type="org.apache.struts.example.LogonForm">
      <icon>
        <small-icon></small-icon>
        <large-icon></large-icon>
      </icon>
      <display-name></display-name>
      <description></description>
      <set-property property="" value=""/>
    </form-bean>
  </form-beans>
  <!--Global forwards specify the target destinations available to the
application.-->
    <global-forwards type="org.apache.struts.action.ActionForward">
    <forward name="logon" path="/logon.jsp" redirect="false">
      <icon>
        <small-icon></small-icon>
        <large-icon></large-icon>
      </icon>
      <display-name></display-name>
      <description></description>
      <set-property property="" value=""/>
    </forward>
  </global-forwards>
  <!--Action mappings associate an action with a given URL.-->
    <action-mappings>
    <action path="/logon" type="org.apache.struts.example.LogonAction"
name="logonForm" scope="request" input="/logon.jsp" unknown="false"
validate="true">
      <icon>
        <small-icon></small-icon>
        <large-icon></large-icon>
      </icon>
      <display-name></display-name>
      <description></description>
      <set-property property="" value=""/>
      <!--Specifies targets specific to this action-->
            <forward name="logon" path="/logon.jsp" redirect="false">
        <icon>
          <small-icon></small-icon>
          <large-icon></large-icon>
        </icon>
        <display-name></display-name>
        <description></description>
        <set-property property="" value=""/>
      </forward>
        </action>
  </action-mappings>
</struts-config>
```

This tutorial does not provide a detailed description of how to configure this file as it is only an overview of the Struts Framework. In summary, however, this configuration file describes the Actions, ActionForms, ActionForwards, and data sources that are defined by this web application. Before you can use any instance of these objects in your application they must first be defined in this file.

Finally to complete the setup of the Struts Framework in your application, you must add the ActionServlet to your deployment descriptor. To do this, simply add the following xml to your application "web.xml" file:

```
  <servlet>
    <servlet-name>action</servlet-name>
    <display-name>ActionServlet</display-name>
    <description>The Struts Action Servlet</description>
    <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>
      <description>Location of the Struts configuration
file.</description>
    </init-param>
    <init-param>
      <param-name>application</param-name>
      <param-value>com.chindogu.ApplicationResources</param-value>
      <description>Application resource containing regionalized
messages and errors.</description>
    </init-param>
    <load-on-startup>99</load-on-startup>
  </servlet>
```

That's it you're done! You can now get started using the Struts framework in your application! If you are looking for a good Integrate Development Environment (IDE) to create your web applications in, you might consider giving "Forte for Java" a try. It's free, and more information about it is available from http://www.sun.com/forte/ffj.

A full list of many great Struts tutorials, books, and resources is available at http://jakarta.apache.org/struts/doc-1.0.2/userGuide/resources.html.