

## Struts - A Primer

white paper

By Steve Cornish  
© ThoughtBreak 2006

# Struts – A Primer

## *What is Struts?*

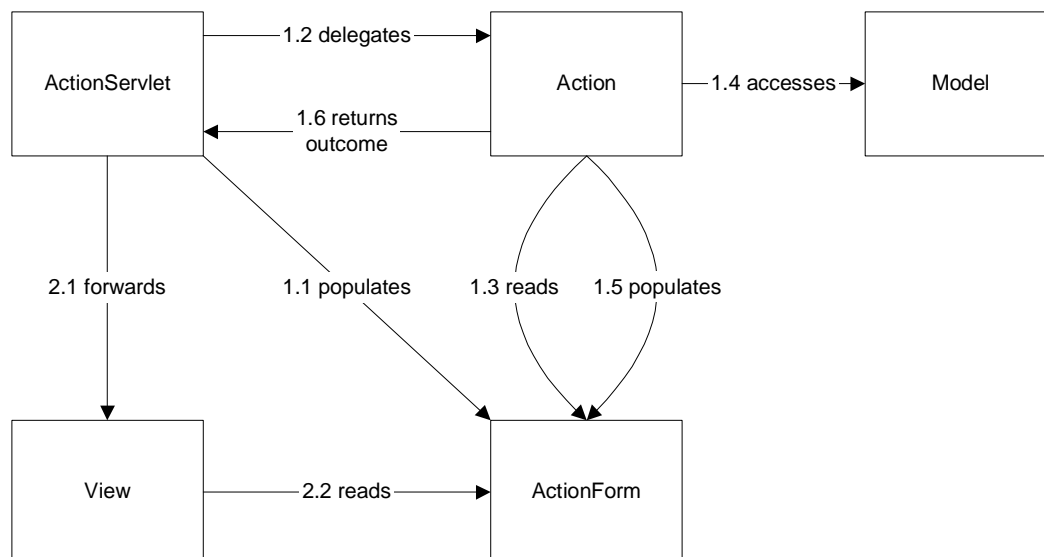
**Apache Struts** is a flexible open-source framework for developing Java web applications. It is the most popular Java web app framework in use today.

Struts is based on standard technologies like Java Servlets, JavaBeans, ResourceBundle and XML, and makes use of many Jakarta Commons packages. It uses and extends the Java Servlet API to encourage developers to adopt an MVC architecture.

It was originally created by Craig McClanahan (also known for the Tomcat Application Server) and donated to the Apache Foundation in May, 2000. Formerly under the Apache Jakarta Project, Struts is now a top level project.

This framework enables design and implementation of large web applications to be split across various roles; page designers, component developers and other developers can each handle their own particular bits of the project, all in tandem and in a decoupled manner.

## *How does Struts work?*



The hosting application server receives a request that it maps to the Struts ActionServlet.

- 1.1 The ActionServlet determines which ActionForm corresponds to this URI, based on configuration from struts-config.xml.

- The ActionServlet then populates the ActionForm from request parameters using reflection.
- 1.2 The ActionServlet determines which Action should be invoked for this URI, based on configuration from struts-config.xml. The ActionServlet then invokes the Action passing the request context and Action Form
  - 1.3 The Action reads request data from the ActionForm.
  - 1.4 The Action accesses the model using data access technologies like JDBC, EJB, Hibernate, etc.
  - 1.5 The Action populates the ActionForm with state needed to render the view.
  - 1.6 The Action returns a logical outcome to the ActionServlet (e.g. 'success', 'failure', 'notLoggedIn')
  - 2.1 The ActionServlet determines where the request should be forwarded for this Action and outcome, based on configuration in struts-config.xml. The ActionServlet then redirects to the target (e.g. a JSP).
  - 2.2 The View component (e.g. a custom JSP) pulls state from the ActionForm to render the response.

It is easier to conceptualise the View component as a JSP, but the View can be any Java-based presentation framework: JSP, JSF (JavaServer Faces), Velocity templates, XSLT etc. JSPs make a good choice as the View component because Struts provides a set of powerful tag libraries that can be used to access data in the ActionForms.

## ***Key Components of a Struts application***

### **Web.xml**

The web application deployment descriptor must initialise the Struts Servlet and associated taglibs.

### **Struts-config.xml**

This is the configuration file for Struts. It contains:

- A list of Form Beans (ActionForm implementations) containing:
  - A local name of the form bean
  - The full classname of the ActionForm
- A list of default (logical) outcomes each with a target URI
- A list of Action mappings containing:
  - The action URI
  - The full classname of the Action
  - The form bean this Action uses
  - A list of (logical) outcomes each with a target URI
- The name of the base resources file

e.g.:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration
1.3//EN"
    "http://struts.apache.org/dtds/struts-config_1_3.dtd">
<struts-config>
  <form-beans>
    <form-bean
      name="logonForm"
      type="org.demo.LogonForm" />
  </form-beans>
  <action-mappings>
    <action
      path="/Welcome"
      forward="/pages/Welcome.jsp" />
    <action
      path="/Logon"
      forward="/pages/Logon.jsp" />
    <action
      path="/LogonSubmit"
      type="org.demo.LogonAction"
      name="logonForm"
      scope="request"
      validate="true"
      input="/pages/Logon.jsp">
      <forward
        name="success"
        path="/pages/Welcome.jsp" />
      <forward
        name="failure"
        path="/pages/Logon.jsp" />
    </action>
    <action
      path="/Logoff"
      type="org.demo.LogoffAction">
      <forward
        name="success"
        path="/pages/Logoff.jsp" />
    </action>
  </action-mappings>
  <message-resources parameter="resources.properties" />
</struts-config>
```

## Actions

Each Action maps onto one or more URIs, and is responsible for processing a request (or family of requests). Action classes read request parameters from the Form Bean, access the model, and populate the Form Bean with data required by the view.

All Actions are sub-classes of **org.apache.struts.action.Action**.

## Action Forms / Form Beans

Each Action works with an Action Form. The form is populated with the request context before being passed to the Action, and the Action further populates the Form with data for the view to access.

All Action Forms are sub-classes of **org.apache.struts.action.ActionForm**.

## **View**

As previously mentioned, the View can be implemented in any number of Java technologies, most commonly JSP, or Tiles (a layout management framework built on JSP).

View components can make use of the Action Form and anything else stored in the Request, Session and Application contexts in order to render the response.

## ***What's so great about Struts?***

On the face of it, what has been described so far could be implemented with Servlets and JSPs. So what does Struts provide that compels use?

### **Application configuration**

Struts-config.xml enables configuration based loose-coupling between the Actions and the Views.

### **ActionForm population and validation**

The ActionForm is populated and validated before the Action gets a sniff. If the form fails validation, control is redirected to the 'input' page, which has the responsibility to communicate the validation failures to the user. Validation can be taken a step further with...

### **Struts Validator**

Rather than manually implement the validate() method on each form, you can use Struts Validator to declare the validation rules to be performed. It is also possible to cause client-side validation JavaScript to be generated.

### **Error handling**

Application errors are handled in a consistent manner, by adding them to a collection of ActionErrors which are then available to the View using the Tag Libraries

### **Role based access control**

Each action can have a list of allowed roles configured against it.

*Note: SVAP uses a custom RequestProcessor to test the declared roles against the SVAP User's roles.*

### **Internationalisation/Localisation**

Struts provides access to a MessageResources class that acts as a façade around a set of resource bundles. This allows you to request a particular message string for a particular Locale instead of the Locale the server is running in.

The MessageResources are available to each Action (Action.getResources()), and also to JSPs via supplied Tag Libraries.

### **Tiles**

Tiles is a powerful templating library that allows you to construct views by combining various tiles. It supports inheritance which makes it particularly useful for defining common layouts once only.

### **Powerful Tag Libraries**

Struts provides Tag Libraries for writing localised resources, handling forms and performing logic. Whilst JSTL has replaced a lot of what Struts provided, the HTML tag library remains indispensable.

### **Open framework**

Struts does not force you to use a particular technology for data access, nor does it force you to use a particular technology for the View.

### ***How ThoughtBreak uses Struts***

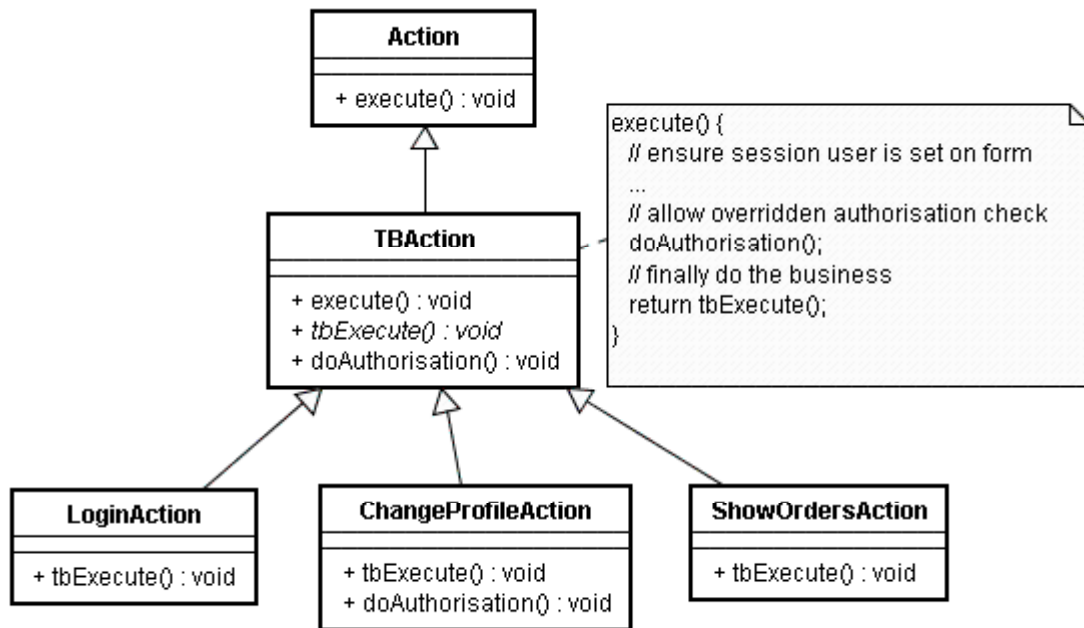
ThoughtBreak is a big fan of Struts, having worked with it for many years.

We find it delivers its promise of being suitable for large, complex web applications with multiple developers.

All our Actions subclass an application-specific custom base class that is responsible for performing common tasks before and after the Action-specific processing. This is an implementation of the

**Template Method pattern.**

E.g.:



We also make use of *role based access control*. We use a custom RequestHandler to perform the check of the declared roles against the Session user's roles in order to provide authorisation.

All of our forms also make use of an application-specific custom base class that provide common functions – e.g. access to the session user object. Our base form class extends *ValidatorForm*, meaning we can take advantage of the *Struts Validator* functionality.

We use *Tiles* for the View components, meaning we can take advantage of defining the look and feel once only. We make heavy use of both *JSTL* and the *Struts Tag Libraries* to render pages.

We also follow the Internationalisation guidelines – accessing all displayed string resources via the *Struts MessageResources* components.

In short, we take full advantage of what Struts gives us to work with.

## ***Pros and Cons***

- + Mature
- + Well documented
- + Widely supported
- + Encourages good practice (simplifies support and maintenance)
- + Open framework - can be used in conjunction with any Java technology
- Still possible to use incorrectly (JSP scriptlets can be turned off in next release)

## ***Resources***

### **Links**

Struts v1 home page - <http://struts.apache.org/1.x/index.html>

Wikipedia on Struts - [http://en.wikipedia.org/wiki/Apache\\_Struts](http://en.wikipedia.org/wiki/Apache_Struts)

Wikipedia on MVC - <http://en.wikipedia.org/wiki/Model-view-controller>

Struts FAQ - <http://www.jguru.com/faq/home.jsp?topic=Struts>