Struts
A JSP Web Application Framework

What is Struts?
- A JSP Web Application Framework
- Provided by the Apache Software Foundation
- Exploits MVC architecture (Model 2)

Why do we need Struts?
- Struts combines Java Servlets, Java ServerPages, custom tags, and message resources into a unified framework
- Cooperative, synergistic platform, suitable for development teams, independent developers

MVC Architecture
Layer Architecture:
- M – Model (Java Beans)
- V – View (JSPs)
- C – Controller (Servlet)

Appropriate for web applications that are:
- Complex and large in size
- Containing a lot of dynamic contents

MVC – How does it work?

MVC - Motivators
- Isolate presentation logic from business logic
- Make business logic independent from specific protocol (e.g. HTTP, XML, RMI)
Web Application Framework
- Help developers build Web applications
- Applications share common set of functionality
- Provide classes and interfaces that can be used/extended by developers

The Struts Framework
- Pre-implemented framework based on MVC architecture
- Delegated to Web tier
- Consists of
  - Components (Java classes) that implement back-bone mechanism of Struts
  - Extensive Custom Tag Libraries

Struts Components

Struts Basic Process

Provided functionalities (1)
- Message resource bundle and Internationalization (I18N)
  - Respond in appropriate language according to the property of Web browser that sent the request
- Input validator
  - Efficient mechanism to validate user input
- Error Handling
  - You can generate error messages and pass them to the appropriate page with simple mechanism

Provided functionalities (2)
- Tiles
  - Template (or layout) approach provided by Tiles Tag library for JSP development
- Advanced Access for Bean Properties
  - With Bean Tag Library, you can access not only simple properties of Beans, but also nested properties and/or indexed properties
- Plugin
  - Load and configure application-specific class as the web application is starting up
- And more...
**How to use Struts**
- Prerequisite Software
- Installation
- Configuration
- Development
- Deployment

**How to use - Prerequisite**
- JDK (J2SE 1.4.2 suggested)
- XML Parser Required
  Bundled with J2SE 1.4 or later
- Servlet Container with servlet.jar
- DBMS with JDBC (Optional)
- Ant Build System 1.5.4 or later (Optional)
  If you build Struts from the source distribution

**How to use – Installation (1)**
- Download the binary distribution or the source code distribution from http://struts.apache.org/acquiring.html
- Unzip (and build) the distribution
- Create your web application root folder, appropriate subfolders, two configuration files (web.xml and struts-config.xml), and a message resource properties file (MessageResource.properties)
- Copy required files from the distribution to your web application

**How to use – Installation (2)**

**How to use – Configuration (1)**
- web.xml
- servlet element
- servlet mapping element
- taglib elements

```xml
<web-app>
    <display-name>Your webapp name</display-name>
    <!-- add elements here -->
</web-app>
```

**How to use – Configuration (2)**
- servlet element

```xml
<service>
    <display-name>action</display-name>
    <service-class>
        org.apache.struts.action.ActionServlet
    </service-class>
    <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</service>
```
How to use – Configuration (3)

- servlet mapping element

```xml
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>/do</url-pattern>
</servlet-mapping>
```

How to use – Configuration (4)

- taglib elements

```xml
<taglib>
  <taglib-url>http://struts.apache.org/tags-bean</taglib-url>
  <taglib-location>/WEB-INF/tags-bean.tld</taglib-location>
</taglib>
```

How to use – Configuration (5)

- struts-config.xml
  - form-beans element
  - global-forwards element
  - action-mappings element
  - message-resources element

```xml
<struts-config>
  <!-- add elements here -->
</struts-config>
```

How to use – Configuration (6)

- form-beans element

```xml
<form-beans>
  <form-bean-name>registrationform</form-bean-name>
  <form-bean-name>loginform</form-bean-name>
  <form-bean-name>orderform</form-bean-name>
</form-beans>
```

How to use – Configuration (7)

- global-forwards element

```xml
<global-forwards>
  <forward name="login" path="/login.jsp"/>
  <forward name="main" path="/main.jsp"/>
  <forward name="error" path="/error.jsp"/>
</global-forwards>
```

How to use – Configuration (8)

- action-mappings element

```xml
<action-mappings>
  <action path="/login" type="LoginAction">
    <forward name="login" path="/login.jsp"/>
  </action>
</action-mappings>
```
How to use – Configuration (9)

- message-resources element

```xml
<message-resources parameter="MessageResources" />
```

How to use – Configuration (10)

- MessageResource.properties

```java
#format: <lkey> <value>
#errors
errors.header=<br />
The following errors occur: <ul>
errors.footer</ul><br />
errors.prefix=<br />
errors.suffix=<br />
error.input.required=(5) is required.
error.user.invalidUnvalid username or password.
```

How to use – Development

- Create JSPs
- Create subclasses of ActionForm class
  - i.e. LoginForm.class
  - Should be created for each form
- Create subclasses of Action class
  - i.e. LoginAction.class
  - Must be thread-safe
- Create other classes in business logic
  - i.e. StateBean classes

How to use – Development

- Copy your web application root folder under webapps folder of your servlet container
- Restart the servlet container to reload your web application
- Access to 
  http://hostname/webapp_root from your web browser

Development Example (1)

- login.jsp

```html
<!-- tags src="http://svn.apache.org/tags-html/profile="default" %>  
</html>
<head>  
<title>Login</title> </head>  
<body>  
<h3>Login</h3>  
<form action="login.do" action="/login">  
<html form errors/>

Username: <input type="text" property="username" /> <br />
Password: <input type="password" property="password" redisplay="false" /> <br />  
<input type="submit" submit="submit" />
</html>
</body>
</html>
```

Development Example (2)

- LoginForm.java

```java
import org.apache.struts.action.*;
import java.util.*;
public class LoginForm extends ActionForm {
  private String username = null;
  private String password = null;
  public String getUsername() { return username; }
  public void setUsername(String username) { this.username = username; }
  public void setPassword(String password) { this.password = password; }
  public boolean validateAction() {
    if (this.username.equals("admin") && this.password.equals("welcome")) {
      return true;
    }
    return false;
  }
  public void execute( ActionMapping mapping, HttpServletRequest request) {
    request.setAttribute("form", this);
  }
}
```
Development Example (3)

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request)
{
    ActionErrors errors = new ActionErrors();

    if (username == null || username.length() == 0)
        errors.add(ActionErrors.GLOBAL_ERROR, new ActionError("error.username"));

    if (password == null || password.length() == 0)
        errors.add(ActionErrors.GLOBAL_ERROR, new ActionError("error.password"));

    return errors;
}
```

Development Example (4)

```
import org.apache.struts.action.*;
import java.servlet.http.*;
import your.package.business.Service;

public class LoginAction extends Action
{
    public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
    {
        String username = ((LoginForm) form).getUsername();
        String password = ((LoginForm) form).getPassword();
    }
```

Development Example (5)

```
if (!Service.authenticate(username, password))
    {
        errors.add(ActionErrors.GLOBAL_ERROR,
                new ActionError("error.user.invalid"));
        saveErrors(request, errors);
        return mapping.getInputForward();
    }
```

Tips for Good Design

- **Separate your business logic from Struts components**
- **Allow only downward dependency**
  Struts components can call Business Logic but Business Logic cannot call Struts component or Servlet classes
- **Call business logic through Interface**
  Provide well-designed interface to be able to change implementation in business logic without change in web tier