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

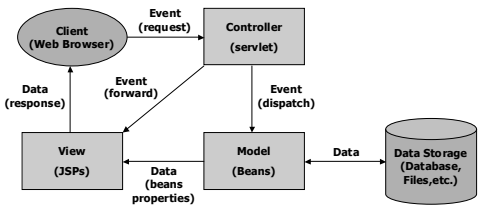
Layer Architecture:

- 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)

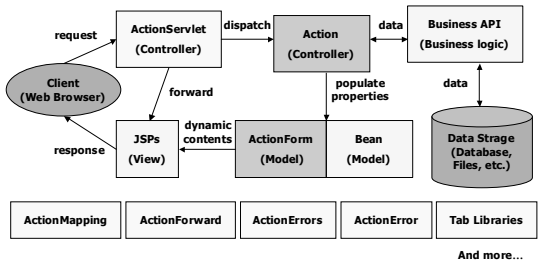
Web Application Framework

- Help developers build Web applications
- Applications share common set of functionality
- Provide classes and interfaces that can be used/extended by developers

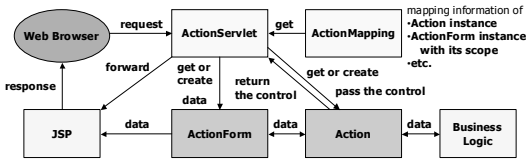
The Struts Framework

- 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

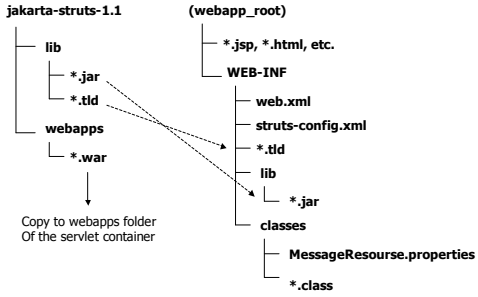
How to use - Prerequisite

- 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 version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
   Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
    <display-name>Your webapp name</display-name>
    <!-- add elements here -->
</web-app>
```

How to use – Configuration (2)

- servlet element

```
<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>
    <load-on-startup>1</load-on-startup>
</servlet>
```

How to use – Configuration (3)

- servlet mapping element

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

How to use – Configuration (4)

- taglib elements

```
<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-nested </taglib-uri>
    <taglib-location> /WEB-INF/struts-nested.tld </taglib-location>
</taglib>
```

How to use – Configuration (5)

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

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
        "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
    <!-- add elements here -->
</struts-config>
```

How to use – Configuration (6)

- form-beans element

```
<form-beans>
    <form-bean name="registrationForm" type="RegistrationFrom" />
    <form-bean name="loginForm"  type="LoginForm" />
    ...
    <form-bean name="orderForm"  type="OrderForm" />
</form-beans>
```

How to use – Configuration (7)

- global-forwards element

```
<global-forwards>
    <forward name="home" path="/index.jsp" />
    <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

```
<action-mappings>
    <action path="/login" type="LoginAction"
            name="loginForm" scope="request" input="/login.jsp">
    </action>
    ...
    <action path="/logout" type="LogoutAction" >
    </action>
</action-mappings>
```

How to use – Configuration (9)

- message-resourses element

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

How to use – Configuration(10)

- MessageResource.properties

```
#format: <key>=<value>
#errors
errors.header=<hr />The following errors occur: <ul>
errors.footer=</ul><hr />
errors.prefix=<li>
errors.suffix=</li>
error.input.required={0} is required.
error.user.invalid=Invalid 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

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<html>
<head><title>Login</title></head>
<body>
    <h3>Log In</h3>
    <html:errors />
    <html:form  action="/login.do" focus="username">
        Username: <html:text  property="username" /><br />
        Password: <html:password  property="password" redisplay="false"/><br />
        <html:submit>submit</html:submit>
    </html:form>
</body>
</html>
```

Development Example (2)

- LoginForm.java

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

public class LoginForm extends ActionForm {

    private String username = null;
    private String password = null;
    public String getUsername() { return username; }
    public String getPassword() { return password; }
    public void setUsername(String username) { this.username = username; }
    public void setPassword(String password) { this.password = password; }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        username = null;  password = null;
    }
```

Development Example (3)

- LoginForm.java (continued)

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {

    ActionErrors errors = new ActionErrors();

    if(username==null || username.length()==0) {
        ActionError error = new ActionError("error.input.required", "Username");
        errors.add(ActionErrors.GLOBAL_ERROR, error);
    }
    if(password==null) || password.length()==0) {
        ActionError error = new ActionError("error.input.required", "Password");
        errors.add(ActionErrors.GLOBAL_ERROR, error);
    }
    return errors;
  }
}
```

Development Example (4)

- LoginAction.java

```
import org.apache.struts.action.*;
import javax.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();
```

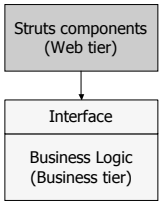
Development Example (5)

- LoginAction.java (continued)

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

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