Chapter 14
Struts

14.1 Introduction

Struts is an application development framework that is designed for and used with the popular J2EE (Java 2, Enterprise Edition) platform. It cuts time out of the development process and makes developers more productive by providing them a series of tools and components to build applications with. It is non-proprietary and works with virtually any J2EE-compliant application server. Struts falls under the Jakarta subproject of the Apache Software Foundation and comes with an Open Source license (meaning it has no cost and its users have free access to all its internal source code).

The struts framework is an open source framework for creating well-structured web based applications. The struts framework is based on the **Model View Controller** (MVC) paradigm which distinctly separates all the three layers - Model (state of the application), View (presentation) and Controller (controlling the application flow). This makes struts different from conventional JSP applications where sometimes logic, flow and UI are mingled in a Java Server Page.

The struts framework is a complete **web framework** as it provides complete web form components, validators, error handling, internationalization, tiles and more. Struts framework provides its own Controller component. It integrates with other technologies for both Model and View components. Struts can integrate well with Java Server Pages (JSP), Java Server Faces (JSF), JSTL, Velocity templates and many other presentation technologies for View. For Model, Struts works great with data access technologies like JDBC, Hibernate, EJB and many more.

The dictionary calls a framework "A structure for supporting or enclosing something else, especially a skeletal support used as the basis for something being constructed." This perfectly describes Struts—a collection of Java code designed to help you build solid applications while saving time. It provides the basic skeleton and plumbing; you focus on the layout and look of each room.

Interestingly, the dictionary offers an alternative definition of a framework: "A set of assumptions, concepts, values, and practices that constitutes a way of viewing reality."

This describes Struts as well—it's a way of looking at things. Struts saves you time by allowing you to view complex applications as a series of basic components: Views, Action Classes, and Model components.

Using a framework means that you don't have to spend time building your entire application. You can focus on coding the business logic and the presentation layer of the application—not the overhead pieces like figuring out how to capture user input or figuring out how to generate drop-down boxes on a Web page.

Another benefit of using a framework is that it allows your code (at least in the case of Struts) to be highly platform independent. For example, the same Struts code should work under Tomcat on an old Windows machine as runs using Weblogic on Linux or Solaris in production. And this can be accomplished without even recompiling in many cases—the same Web application (or ". war" file) can simply be copied from one server to another.

14.1.1 How Does Struts Work?

Struts is based on the time-proven Model-View-Controller (MVC) design pattern. The MVC pattern is widely recognized as being among the most well-developed and mature design patterns in use. By using the MVC design pattern, processing is broken into three distinct sections aptly named the Model, the View, and the Controller. These are described in the following subsections:

A) Model Components

Model components provide a "model" of the business logic or data behind a Struts program. For example, in a Struts application that manages customer information, it may be appropriate to have a "Customer" Model component that provides program access to information about customers.

It's very common for Model components to provide interfaces to databases or back-end systems. For example, if a Struts application needs to access employee information that is kept in an enterprise HR information system, it might be appropriate to design an "Employee" Model component that acts as an interface between the Struts application and the HR information system.

Model components are generally standard Java classes. There is no specifically required format for a Model component, so it may be possible to reuse Java code written for other projects.

B) View Components

View components are those pieces of an application that present information to users and accept input. In Struts applications, these correspond to Web pages.

View components are used to display the information provided by Model components. For example, the "Customer" Model component discussed above would need a View component to display its information. Usually, there will one or more View components for each Web page in a Struts application.

View components are generally built using JavaServer Page (JSP) files. Struts provides a large number of "JSP Custom Tags" (sometimes referred to as Struts Tags) which extend the normal capabilities of JSP and simplify the development of View components.

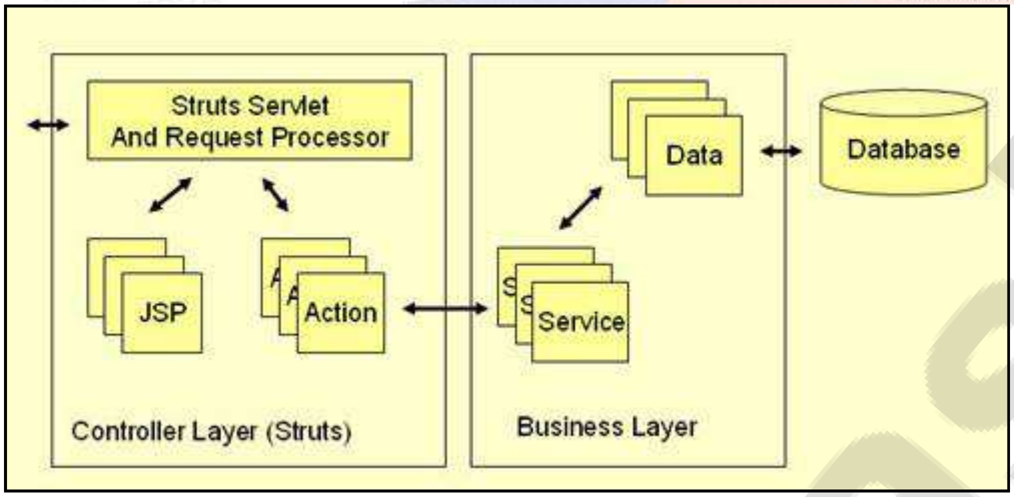
C) Controller Components

Controller components coordinate activities in the application. This may mean taking data from the user and updating a database through a Model component, or it may mean detecting an error condition with a back-end system and directing the user through special error processing. Controller components accept data from the users, decide which Model components need to be updated, and then decide which View component needs to be called to display the results.

One of the major contributions of Controller components is that they allow the developer to remove much of the error handling logic from the JSP pages in their application. (After all, if errors in processing occur, the Controller component forwards to an error-processing View component, not the primary results View component.) This can significantly simplify the logic in the pages and make them easier to develop and maintain.

Controller components in Struts are Java classes and must be built using specific rules. They are usually referred to as "Action classes."

Architecture Overview



All incoming requests are intercepted by the Struts servlet controller. The Struts Configuration file struts-config.xml is used by the controller to determine the routing of the flow. This flows consists of an alternation between two transitions:

From View to Action	A user clicks on a link or submits a form on an HTML or JSP page. The controller receives the request, looks up the mapping for this request, and forwards it to an action. The action in turn calls a Model layer (Business layer) service or function.
From Action to View	After the call to an underlying function or service returns to the action class, the action forwards to a resource in the View layer and a page is displayed in a web browser.

14.1.2 Basic Components of Struts

The Struts framework is a rich collection of Java libraries and can be broken down into the following major pieces:

Base framework

JSP tag libraries

Tiles plugin

Validator plugin

A brief description of each follows.

A) Base Framework

The base framework provides the core MVC functionality and is comprised of the building blocks for your application. At the foundation of the base framework is the Controller servlet: **ActionServlet**. The rest of the base framework is comprised of base classes that your application will extend and several utility classes. Most prominent among the base classes are the **Action** and **ActionForm** classes. These two classes are used extensively in all Struts applications. **Action** classes are used by **ActionServlet** to process specific requests. **ActionForm** classes are used to capture data from HTML forms and to be a conduit of data back to the View layer for page generation.

B) JSP Tag Libraries

Struts comes packaged with several JSP tag libraries for assisting with programming the View logic in JSPs. JSP tag libraries enable JSP authors to use HTML-like tags to represent functionality that is defined by a Java class.

Following is a listing of the libraries and their purpose:

HTML Used to generate HTML forms that interact with the Struts APIs.

Bean Used to work with Java bean objects in JSPs, such as accessing bean values.

Logic Used to cleanly implement simple conditional logic in JSPs.

Nested Used to allow arbitrary levels of nesting of the HTML, Bean, and Logic tags that otherwise do not work.

C) Tiles Plugin

Struts comes packaged with the Tiles sub framework. Tiles is a rich JSP templating framework that facilitates the reuse of presentation (HTML) code. With Tiles, JSP pages can be broken up into individual **'tiles'** or pieces and then glued together to create one cohesive page. Similar to the design principles that the core Struts framework is built on, Tiles provides excellent reuse of View code. As of Struts 1.1, Tiles is part of and packaged with the core Struts download. Prior to Struts 1.1, Tiles was a third-party add-on, but has since been contributed to the project and is now more tightly integrated.

D) Validator Plugin

Struts comes packaged, as of version 1.1, with the Validator subframework for performing data validation. Validator provides a rich framework for performing data validation on both the server side and client side (browser). Each validation is configured in an outside XML file so that validations can easily be added to and removed from an application declaratively versus being hard-coded into the application. Similar to Tiles, prior to Struts 1.1, Validator was a third-party add-on, but has since been included in the project and is more tightly integrated.

14.2 Struts framework core components

Struts Components

14.2.1 The Controller

The controller is responsible for intercepting and translating user input into actions to be performed by the model. The controller is responsible for selecting the next view based on user input and the outcome of model operations. The Controller receives the request from the browser, invoke a business operation and coordinating the view to return to the client.

The controller is implemented by a java servlet, this servlet is centralized point of control for the web application. In struts framework the controller responsibilities are implemented by several different components like

The ActionServlet Class
The RequestProcessor Class
The Action Class

The ActionServlet extends the **javax.servlet.http.httpServlet** class. The ActionServlet class is not abstract and therefore can be used as a concrete controller by your application.

The controller is implemented by the ActionServlet class. All incoming requests are mapped to the central controller in the deployment descriptor as follows.

```
<servlet>

   <servlet-name>action</servlet-name>

   <servlet-class>org.apache.struts.action.ActionServlet</servlet
-class>

</servlet>
```

All request URIs with the pattern *.do are mapped to this servlet in the deployment descriptor as follows.

```
<servlet-mapping>

   <servlet-name>action</servlet-name>

   <url-pattern>*.do</url-pattern>

<url-pattern>*.do</url-pattern>
```

A request URI that matches this pattern will have the following form.
`http://www.my_site_name.com/mycontext/actionName.do`

This receives all incoming requests. Its primary function is the mapping of a request URI to an action class selecting the proper application module. It's provided by the framework.

14.2.2 The struts-config.xml File

This file contains all of the routing and configuration information for the Struts application. This XML file needs to be in the WEB-INF directory of the application.

Table: Important attributes and elements of ActionMapping entry in struts-config.xml

Attribute/Element name	Description
Path	The URL path (either path mapping or suffix mapping) for which this Action Mapping is used. The path should be unique
Type	The fully qualified class name of the Action
Name	The logical name of the Form bean. The actual ActionForm associated with this Action Mapping is found by looking in the Form-bean definition section for a form-bean with the matching name. This informs the Struts application which action mappings should use which ActionForms.
Scope	Scope of the Form bean – Can be session or request
Validate	Can be true or false. When true, the Form bean is validated on submission. If false, the validation is skipped.
Input	The physical page (or another ActionMapping) to which control should be forwarded when validation errors exist in the form bean.
Forward	The physical page (or another ActionMapping) to which the control should be forwarded when the ActionForward with this name is selected in the execute method of the Action class.

The `ActionMappingsection` contains the mapping from URL path to an Action class (and also associates a Form bean with the path). The `typeattribute` is the fully qualified class name of the associated Action. Each action entry in the `action-mappings` should have a unique path. This follows from the fact that each URL path needs a unique handler. There is no facility to associate multiple Actions with the same path. The `name` attribute is the name of the Form bean associated with this Action. The actual form bean is defined in Form bean definition section. Table above shows all the relevant attributes discussed so far for the `action` entry in `action-mappings` section.

Example-

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!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>

  <!-- ========== Form Bean Definitions ============ -->

  <form-beans>

   <form-bean name="login" type="test.struts.LoginForm" />

  </form-beans>

  <!-- ========== Global Forward Definitions ========= -->

  <global-forwards>

  </global-forwards>


  <!-- ========== Action Mapping Definitions ======== -->

  <action-mappings>

    <action

      path="/login"

      type="test.struts.LoginAction" >

       <forward name="valid" path="/jsp/MainMenu.jsp" />

       <forward name="invalid" path="/jsp/LoginView.jsp" />

   </action>

  </action-mappings>
```

```
</struts-config>
```

14.2.3 Action Classes

It's the developer's responsibility to create these classes. They act as bridges between user-invoked URIs and business services. Actions process a request and return an ActionForward object that identifies the next component to invoke. They're part of the Controller layer, not the Model layer.

Action Class in Struts framework defines the **business logic**. An action class handles the client request and prepares the response. It also decides where the response should be forwarded. Basically an action class receives data from the presentation layer and forwards the data to the corresponding business layer. It also processes the data which comes from the business layer and forwards them back to the presentation layer. In short, an action class is described as:

- Action class should extend org.apache.struts.action.Action class.
- Should override the execute method of the Action class.
- The Action Servlet selects the Action class for incoming http request defined under the action mapping tag in the struts config.xml file.
- These classes are used to invoke the classes at the business layer or data access layer to get the data from the bean and store the processed data and return the result or error depending upon the situation.
- These classes are multi threaded. Hence you have to be careful while handling the action variable as they are not thread safe.

Types of action class:

We have following types of action classes in struts:

- Action - The basic action class in which we implement our business logic.
- Include Action - Similar as include page directive in jsp.
- Forward Action - Used in case we need to forward the request from one JSP to another. If we directly forward the request from one jsp it violates the MVC architecture. Hence an action class is used to do this job.
- Dispatch Action - Handles multiple operations in multiple methods. It is better to have one method per operation instead of merging the entire business logic in a single execute method of an action class.
- Look up Dispatch Action - Same as dispatch action but recommended not to use.
- Switch Action - used to switch between different modules in struts application.

Most commonly used action classes are:

- Action
- Dispatch Action

Difference between Action and DispatchAction classes

- Grouping related actions into one class is possible using DispatchAction class. But in Action class a group related action into one class is not possible. Several changes need to be done, when one plans to group the related action using Action class.
- Action class is used to perform single functionality. To perform multiple functionalities, we need to write an additional Action class.
- The complexity rises if Action class is used for multiple functionalities, whereas by using DispatchAction, the code complexity will noticeably be reduced.
- An Action class acts as a controller and is an adapter class between the web tier and the business tier.
- Dispatch class is used to combine the related operations, thus share the common resources, helper classes for instance.
- An Action is invoked by the ActionServlet to perform an operation that is depending on the URL request.
- A DispatchAction selects a method depending on the value of the request parameter that is configured in the xml file.

14.2.4 View Resources

View resources consist of Java Server Pages, HTML pages, JavaScript and Stylesheet files, Resource bundles, JavaBeans, and Struts JSP tags.

The view is responsible for rendering the state of the model. The presentation semantics are encapsulated within the view, therefore model data can be adapted for several different kinds of clients.The view modifies itself when a change in the model is communicated to the view. A view forwards user input to the controller.The view is simply a JSP or HTML file. There is no flow logic, no business logic, and no model information -- just tags. Tags are one of the things that make Struts unique compared to other frameworks like Velocity.

The view components typically employed in a struts application are
HTML
Data Transfer Objects

Struts ActionForms
JavaServer pages
Custom tags
Java resource bundles

14.2.5 ActionForms

These greatly simplify user form validation by capturing user data from the HTTP request. They act as a "firewall" between forms (Web pages) and the application (actions). These components allow the validation of user input before proceeding to an Action. If the input is invalid, a page with an error can be displayed.

There are six types of **FormBean classes** are available in struts.

ActionForm -Action form for action class.

DynaActionForm – No form bean is created. No Validations are done here. Form bean properties are specified in struts-config.xml file

ValidatorForm Uses validation framework for validating the form bean.

DynaValidatorForm – Similar to the ValidatorForm but form bean is not created and form bean properties are specified in the struts-config.xml file. Validations here are done according to the to the form bean name.

ValidatorActionForm – Validations are done according to the action class name and not according to the form bean name as in ValidatorForm. Form bean is still created.

DynaValidatorActionForm – Similar to the ValidatorActionForm but form bean is not created and form bean properties are specified in the struts-config.xml file. Validations are done according to the Action class name.

14.2.6 Model Components

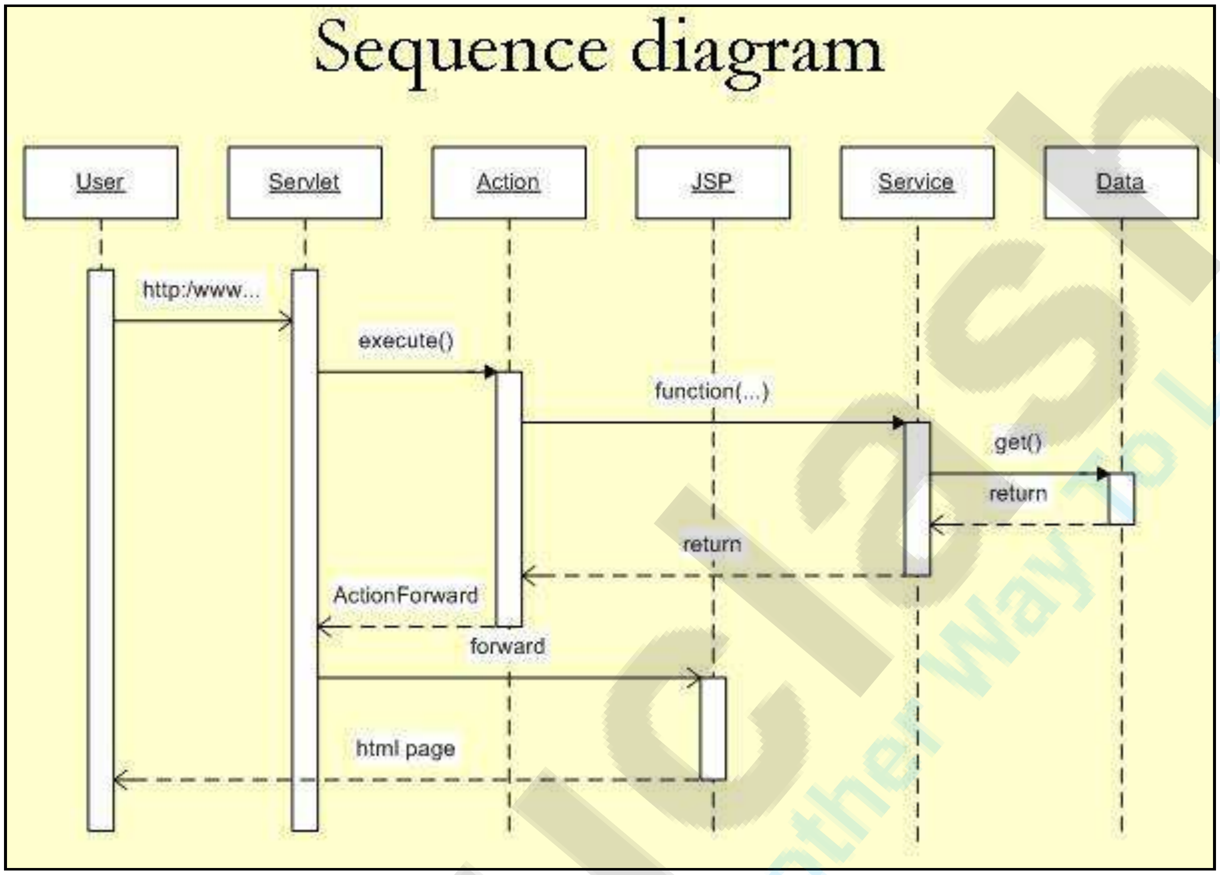
The Struts Framework has no built-in support for the Model layer. Struts supports any model components:

- JavaBeans
- EJB
- CORBA
- JDO
- any other

A model represents an application's data and contains the logic for accessing and manipulating that data. Any data that is part of the persistent state of the application should reside in the model objects. The business objects update the application state. ActionForm bean represents the Model state at a session or request level, and not at a persistent level. Model services are accessed by the controller for either querying or effecting a change in the model state. The model notifies the view when a state change occurs in the model.The JSP file reads information from the ActionForm bean using JSP tags.

The Enterprise JavaBeans (EJB), Java Data Objects(JDO) and JavaBeans can be use as a model. Struts frame work doesn't limit you to one particular model implementation.

The diagram below describes the flow in more detail:



- **User** clicks on a link in an HTML page.
- **Servlet** controller receives the request, looks up mapping information in struts-config.xml, and routes to an action.
- **Action** makes a call to a Model layer service.
- **Service** makes a call to the Data layer (database) and the requested data is returned.
- **Service** returns to the action.
- **Action** forwards to a View resource (JSP page)
- **Servlet** looks up the mapping for the requested resource and forwards to the appropriate JSP page.
- **JSP** file is invoked and sent to the browser as HTML.
- **User** is presented with a new HTML page in a web browser.

14.3 installing and setting up struts

Installation:

In order to do any Java development you need the Java Developer Kit. You can find JDK1.4 at http://java.sun.com. Just execute the installation executable and follow the instructions.

You will need to setup the PATH and the JAVA HOME variables; Struts can also be found as a subproject of the Jakarta project (http://jakarta.apache.org). There is no Struts installation for the moment, just uncompress it in a convenient directory.

Copy the following JAR files, extracted from the Jakarta Struts archive, to the `/WEB-INF/lib` directory:

struts.jar
```
commons-beanutils.jar
commons-collections.jar
commons-digester.jar
commons-logging.jar
commons-validator.jar
```

Copy the tld files you want to use in the `/WEB-INF` directory:
```
struts-html.tld
struts-bean.tld
struts-logic.tld
struts-nested.tld
struts-template.tld
```

If you want to use tiles add the `struts-tiles.tld` also in the `/WEB-INF` directory.

Add `struts-config.xml` file in the `/WEB-INF` directory

setting up struts

Struts framework use two related type of configuration file, which nust be configured properly before and application will work correctly. Both configration files are XML based.

The first one is the web application deployment descriptor `web.xml` and the second one the Struts configuration file, it is comonly named `struts-config.xml`.

Configuring web.xml

There are few Struts-specific configuration options that you must configure within this file when using the struts framework.

Modify the `web.xml` file in `/WEB-INF` directory by adding the `.tld` file you want to use like if you are using `struts-html.tld` then

```
<taglib>

   <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>

 <taglib-location>/WEB-INF/struts-html.tld</taglib-location>

</taglib>
```

Configure the ActionServlet that will receive all the incoming request for the appliation. The two steps involved in configuring the Struts controller servlet in the `web.xml` are, first to use the servlet element to configure the servlet instance that can be latter mapped in the servlet mapping element.
Using servlet element to configure the servlet class in `web.xml`

```
<web-app>

   <sertvlet>

      <servlet-name>some name</servlet-name>

      <servlet-class>the class name</servlet-class>

    </sertvlet>

</web-app>
```

Configure servlet mapping

```
<web-app>
```

```
    <sertvlet>

       <servlet-name>some name</servlet-name>

       <servlet-class>the class name</servlet-class>

    </sertvlet>

    <sertvlet-mapping>

       <servlet-name>some name</servlet-name>

       <url-pattern>*.do </url-pattern>

    </sertvlet-mapping>

</web-app>
```

14.4 getting started with struts

Create Struts Application in Eclipse

Things We Need

Before we starts with our first Hello World Struts 2 Example, we will need few tools.

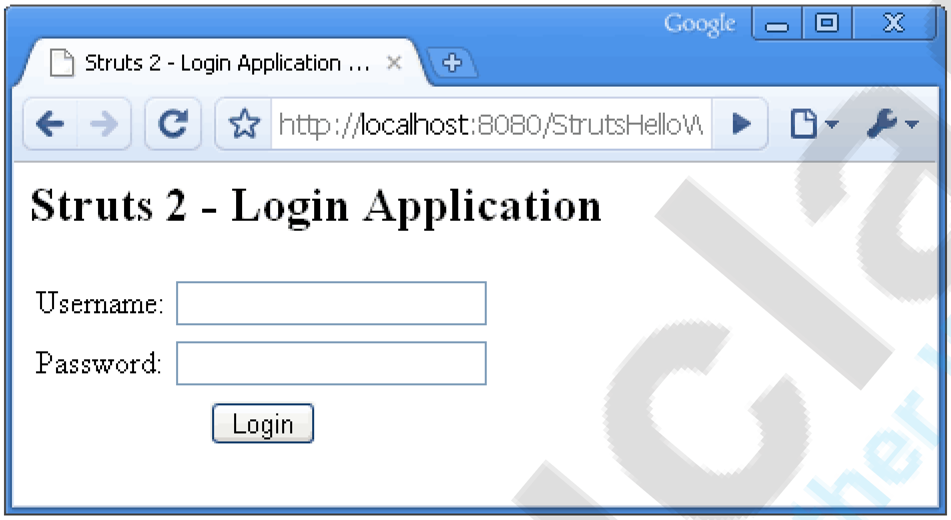
1 JDK 1.5 above
2 Tomcat 5.x above or any other container (Glassfish, JBoss, Websphere, Weblogic etc)
3 Eclipse 3.2.x above
4 Apache Struts2 JAR files: Following are the list of JAR files required for this application.

  - commons-logging-1.0.4.jar
  - freemarker-2.3.8.jar
  - ognl-2.6.11.jar
  - struts2-core-2.0.12.jar
  - xwork-2.0.6.jar

Note that depending on the current version of Struts2, the version number of above jar files may change.

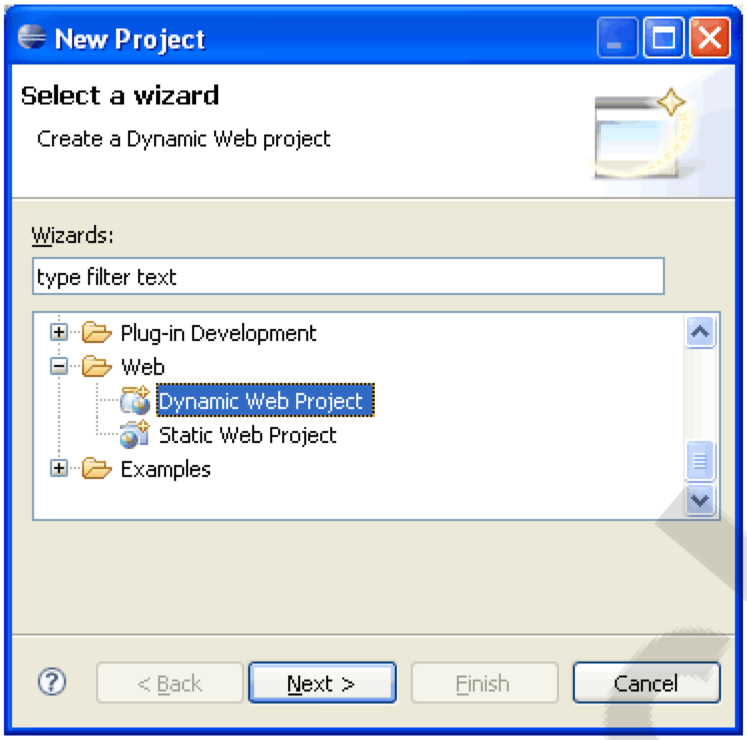
Our Goal

Our goal is to create a basic Struts2 application with a Login page. User will enter login credential and if authenticated successfully she will be redirected to a Welcome page which will display message " *Howdy, …!*". If user is not authenticated, she will be redirected back to the login page.


Struts 2 - Login Application ...
http://localhost:8080/StrutsHelloW
Struts 2 - Login Application
Username:
Password:
Login

Getting Started

Let us start with our first Struts2 based application.
Open Eclipse and goto File -> New -> Project and select **Dynamic Web Project** in the New Project wizard screen.



Select a wizard

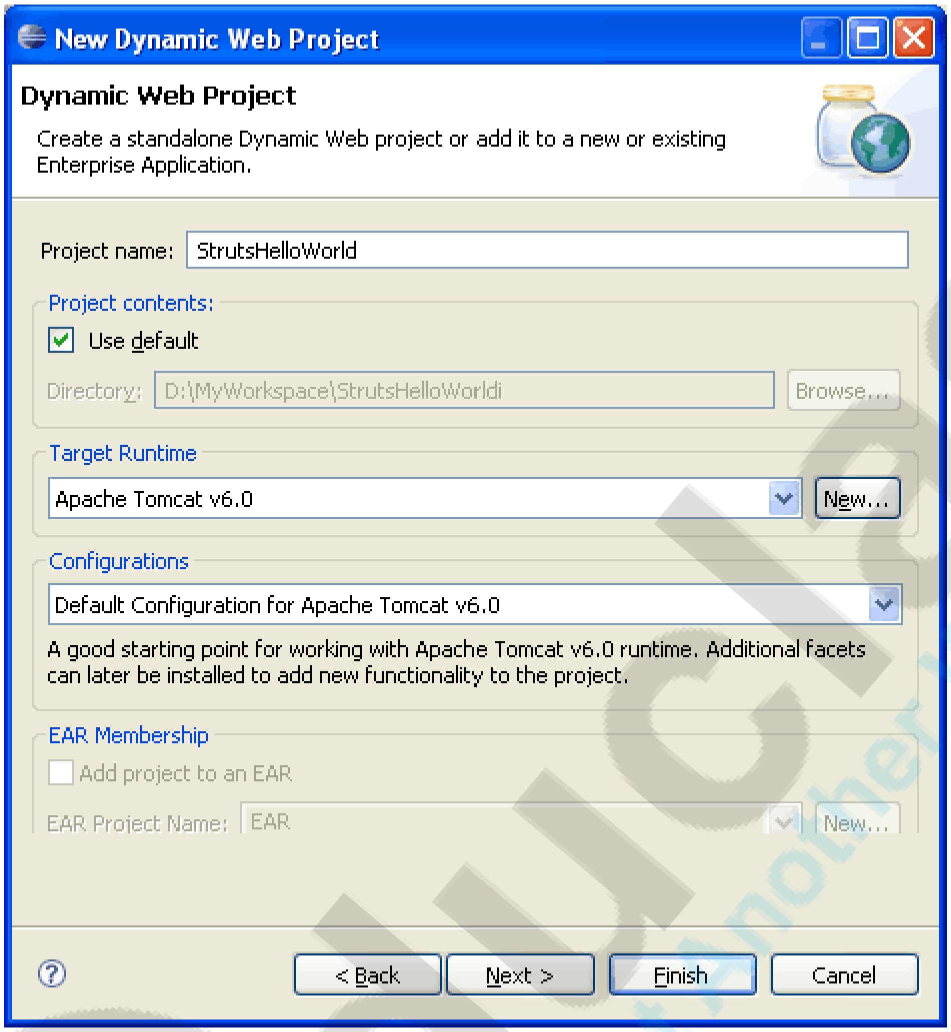
Create a Dynamic Web project

Wizards:

type filter text

- Plug-in Development
- Web
  - Dynamic Web Project
  - Static Web Project
- Examples

After selecting Dynamic Web Project, press **Next**.



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: StrutsHelloWorld

Project contents:

☑ Use default

Directory: D:\MyWorkspace\StrutsHelloWorldi

Target Runtime

Apache Tomcat v6.0

Configurations

Default Configuration for Apache Tomcat v6.0

A good starting point for working with Apache Tomcat v6.0 runtime. Additional facets can later be installed to add new functionality to the project.

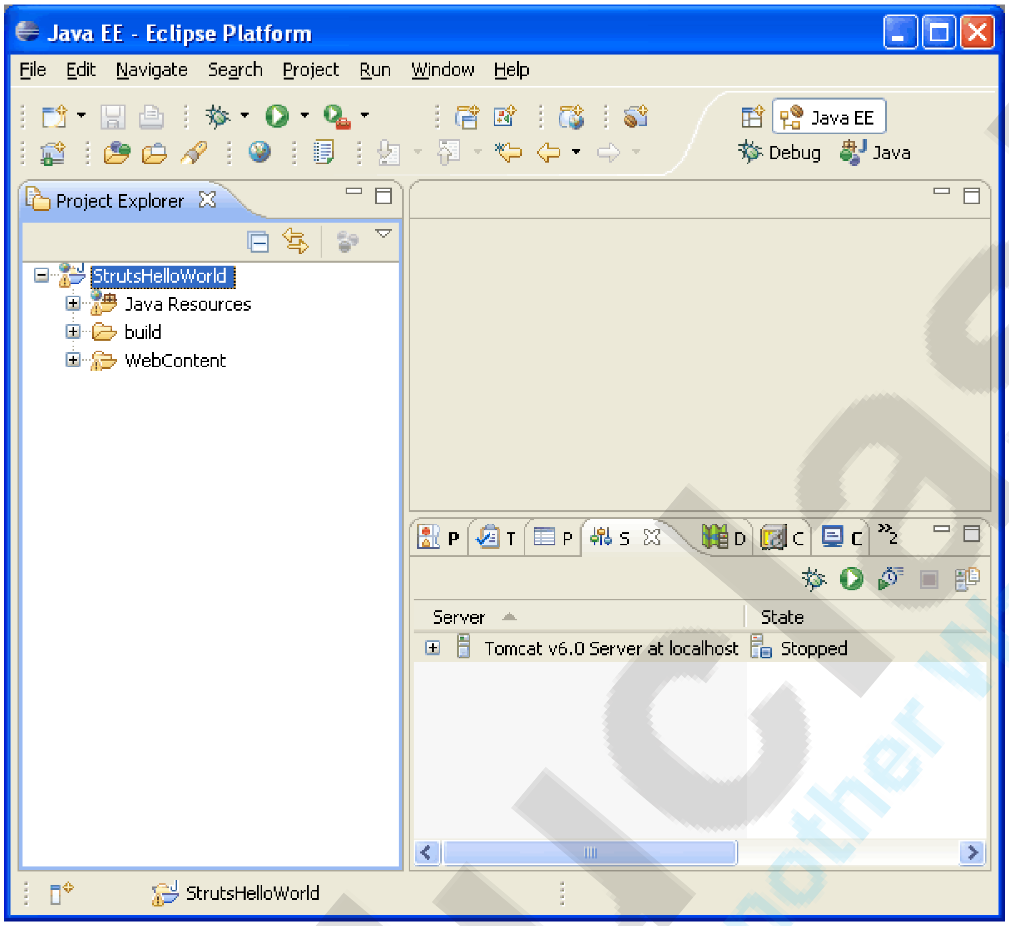
EAR Membership

☐ Add project to an EAR

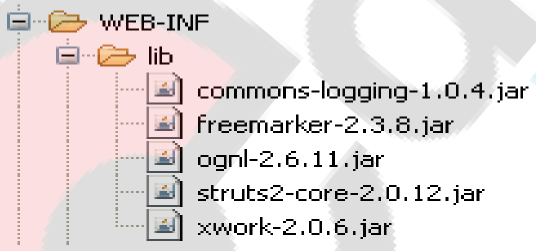
EAR Project Name: EAR

Write the name of the project. For example **StrutsHelloWorld**. Once this is done, select the target runtime environment (e.g. Apache Tomcat v6.0). This is to run the project inside Eclipse environment. After this press **Finish**.

Once the project is created, you can see its structure in Project Explorer.



Now copy all the required JAR files in WebContent -> WEB-INF -> lib folder. Create this folder if it does not exists.



Mapping Struts2 in WEB.xml

As discussed above the entry point of Struts2 application will be the Filter define in deployment descriptor (web.xml). Hence we will define an entry of `org.apache.struts2.dispatcher.FilterDispatcher` class in `web.xml`.

Open `web.xml` file which is under `WEB-INF` folder and copy paste following code.

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

<web-appid="WebApp_9"version="2.4"xmlns="http://java.sun.com/xml/
ns/j2ee"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:
schemalocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <display-name>Struts2 Application</display-name>

    <filter>

        <filter-name>struts2</filter-name>

        <filter-class>

            org.apache.struts2.dispatcher.FilterDispatcher

        </filter-class>

    </filter>

    <filter-mapping>

        <filter-name>struts2</filter-name>

        <url-pattern>/*</url-pattern>

    </filter-mapping>

    <welcome-file-list>

        <welcome-file>Login.jsp</welcome-file>

    </welcome-file-list>

</web-app>
```

The above code in `web.xml` will map Struts2 filter with url `/*`. The default url mapping for struts2 application will be `/*`.action. Also note that we have define `Login.jsp` as welcome file.

Note: The `FilterDispatcher` filter is deprecated since Struts version 2.1.3. If you are using latest version of Struts2 (> 2.1.3) use `StrutsPrepareAndExecuteFilter` class instead.

```
<filter>

   <filter-name>struts2</filter-name>


<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepa
reAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

    <filter-name>struts2</filter-name>

    <url-pattern>/*</url-pattern>

</filter-mapping>
```

The Action Class

We will need an Action class that will authenticate our user and holds the value for username and password. For this we will create a package **net.test.struts2** in the source folder. This package will contain the action file.

Create a class called **LoginAction** in net.test.struts2 package with following content.

```
packagenet.test.struts2;

publicclassLoginAction {

    privateString username;

    privateString password;
    publicString execute() {

        if(this.username.equals("admin")

                &&this.password.equals("admin123")) {

            return"success";

        }else{

            return"error";

        }

    }

    publicString getUsername() {

        returnusername;

    }

    publicvoidsetUsername(String username) {

        this.username = username;

    }

    publicString getPassword() {

        returnpassword;

    }

    publicvoidsetPassword(String password) {

        this.password = password;

    }

}
```

Note that, above action class contains two fields, username and password which will hold the values from form and also contains an execute() method that will authenticate the user. In this simple example, we are checking if **username is *admin*** and **password is *admin123***.

Also note that unlike Action class in Struts1, Struts2 action class is a simple POJO class with required attributes and method.

The execute() method returns a String value which will determine the result page. Also, in Struts2 the name of the method is not fixed. In this example we have define method execute(). You may want to define a method `authenticate()` instead.

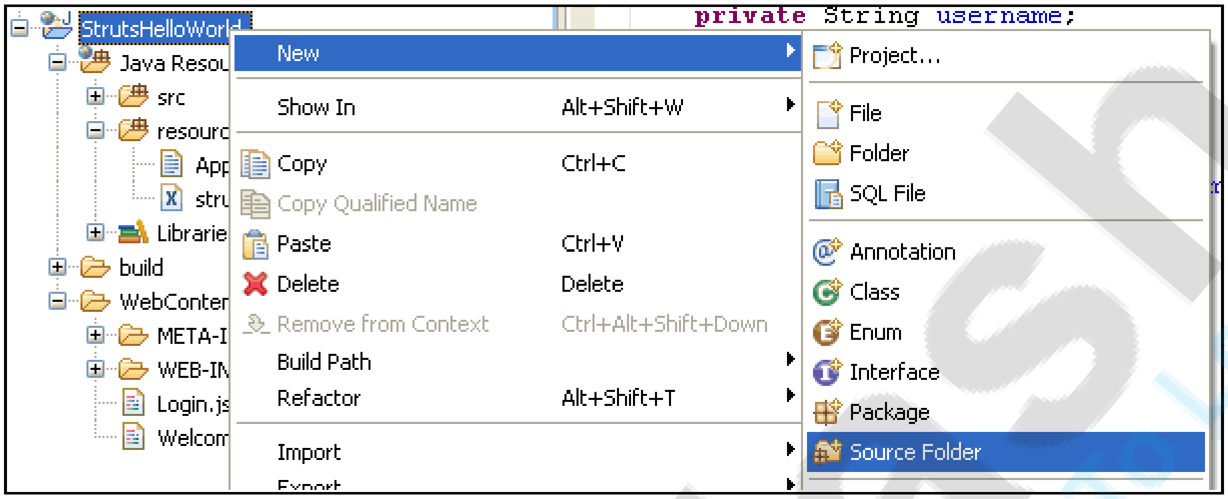
The ResourceBundle

ResourceBundle is very useful Java entity that helps in putting the static content away from the source file. Most of the application define a resource bundle file such as `ApplicationResources.properties` file which contains static messages such as Username or Password and include this with the application.

ResourceBundle comes handy when we want to add Internationalization (I18N) support to an application.

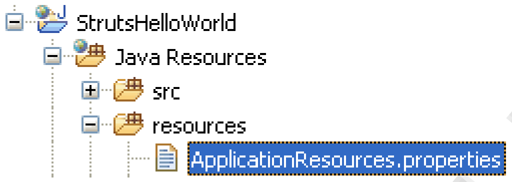
We will define an `ApplicationResources.properties` file for our application. This property file should be present in `WEB-INF/`classes folders when the source is compiled. Thus we will create a source folder called **resources** and put the `ApplicationResources.properties` file in it.

To create a source folder, right click on your project in Project Explorer and select **New -> Source Folder**.



Specify folder name **resources** and press **Finish**.

Create a file `ApplicationResources.properties` under resources folder.



Copy following content in `ApplicationResources.properties`.

```
label.username= Username

label.password= Password

label.login= Login
```

The JSP

We will create two JSP files to render the output to user. `Login.jsp` will be the starting point of our application which will contain a simple login form with username and password. On successful authentication, user will be redirected to `Welcome.jsp` which will display a simple welcome message.

Create two JSP files `Login.jsp` and `Welcome.jsp` in WebContent folder of your project. Copy following content into it.

Login.jsp

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

<%@ taglib prefix="s" uri="/struts-tags"%>

<title>Struts 2 - Login Application..</title>

<h2>Struts 2 - Login Application</h2>

<s:actionerror>

<s:formaction="login.action"method="post">

    <s:textfieldname="username"key="label.username"size="20">

    <s:passwordname="password"key="label.password"size="20">

    <s:submitmethod="execute"key="label.login"align="center">

</s:submit></s:password></s:textfield></s:form>

 </s:actionerror>
```

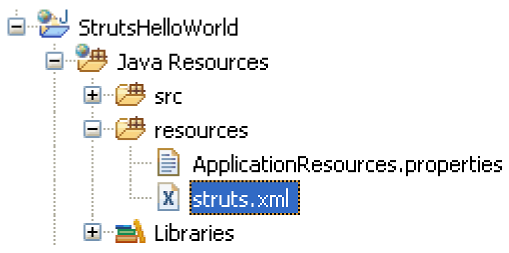
Welcome.jsp

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

<%@ taglib prefix="s" uri="/struts-tags"%>

<title>Welcome</title>

<h2>Howdy, <s:propertyvalue="username">...!</s:property></h2>
```

Note that we have used struts2 tag to render the textboxes and labels. Struts2 comes with a powerful built-in tag library to render UI elements more efficiently.

The struts.xml file

Struts2 reads the configuration and class definition from an xml file called `struts.xml`. This file is loaded from the classpath of the project. We will define `struts.xml` file in the **resources** folder. Create file `struts.xml` in resources folder.



```
StrutsHelloWorld
    Java Resources
        src
        resources
            ApplicationResources.properties
            struts.xml
        Libraries
```

Copy following content into `struts.xml`.

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

<struts>
<constantname="struts.enable.DynamicMethodInvocation"value="false
">

<constantname="struts.devMode"value="false">


<constantname="struts.custom.i18n.resources"value="ApplicationRes
ources">

    <packagename="default"extends="struts-default"namespace="/">

        <actionname="login"class="net.test.struts2.LoginAction">

            <resultname="success">Welcome.jsp</result>

            <resultname="error">Login.jsp</result>

        </action>
    </package>

</constant></constant></constant></struts>
```

Note that in above configuration file, we have defined Login action of our application. Two result paths are mapped with LoginAction depending on the outcome of `execute()` method. If `execute()` method returns success, user will be redirected to `Welcome.jsp` else to `Login.jsp`.

Also note that a constant is specified with name `struts.custom.i18n.resources`. This constant specify the resource bundle file that we created in above steps. We just have to specify name of resource bundle file without extension (`ApplicationResources without .properties`).

Our LoginAction contains the method `execute()` which is the default method getting called by Sturts2. If the name of method is different, e.g. `authenticate();` then we should specify the method name in tag.

```
<actionname="login"method="authenticate"class="net.test.struts2.L
oginAction">

</action>
```

Almost Done

We are almost done with the application. You may want to run the application now and see the result yourself. I assume you have already configured Tomcat in eclipse. All you need to do:

Open Server view from Windows -> Show View -> Server.

Right click in this view and select New -> Server and add your server details.

To run the project, right click on Project name from Project Explorer and select Run as -> Run on Server (Shortcut: Alt+Shift+X, R)

But there is one small problem. Our application runs perfectly fine at this point. But when user enters wrong credential, she is redirected to `Login page`. But no error message is displayed. User does not know what just happened. A good application always show proper error messages to user. So we must display an error message `Invalid Username/Password`. *Please try again* when user authentication is failed.

Final Touch

To add this functionality first we will add the error message in our `ResourceBundle` file. Open `ApplicationResources.properties` and add an entry for `error.login` in it. The final `ApplicationResources.properties` will look like:

```
label.username= Username

label.password= Password

label.login= Login

error.login= Invalid Username/Password. Please try again.
```

Also we need to add logic in `LoginAction` to add error message if user is not authenticated. But there is one problem. Our error message is specified in `ApplicationResources.properties` file. We must specify key `error.login` in `LoginAction` and the message should be displayed on JSP page.

For this we must implement `com.opensymphony.xwork2.TextProvider` interface which provides method `getText()`. This method returns String value from resource bundle file. We just have to pass the key value as argument to `getText()` method. The TextProvider interface defines several method that we must implement in order to get hold on `getText()` method. But we don’t want to spoil our code by adding all those methods which we do not intend to use. There is a good way of dealing with this problem.

Struts2 comes with a very useful class `com.opensymphony.xwork2.ActionSupport`. We just have to extend our `LoginAction` class with this class and directly use methods such as `getText()`, `addActionErrors()` etc. Thus we will extend the `LoginAction` class with `ActionSupport class` and add the logic for error reporting into it. The final code in `LoginAction` must look like:

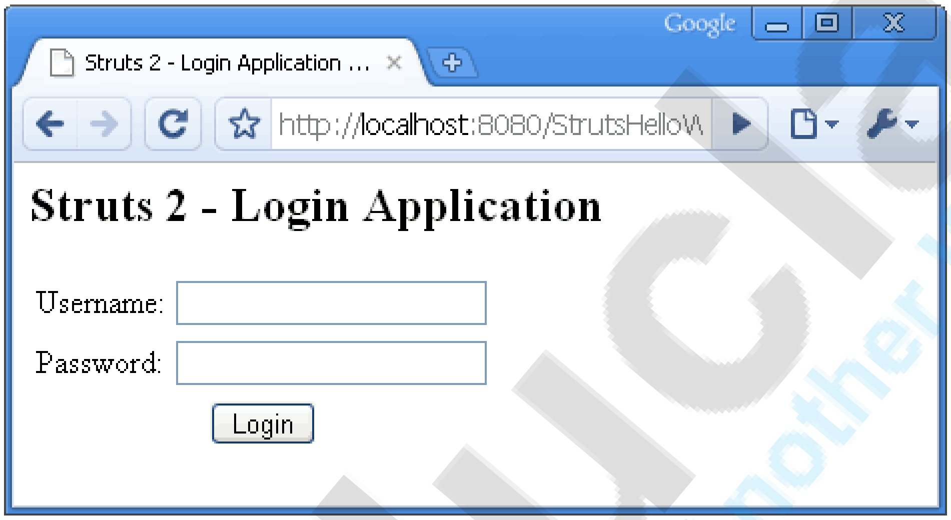
```
packagenet.test.struts2;

importcom.opensymphony.xwork2.ActionSupport;
```

```
publicclassLoginActionextendsActionSupport {

    privateString username;

    privateString password;

    publicString execute() {

        if(this.username.equals("admin")

                &&this.password.equals("admin123")) {

            return"success";

        }else{

            addActionError(getText("error.login"));

            return"error";

        }

    }

    publicString getUsername() {

        returnusername;

    }

    publicvoidsetUsername(String username) {

        this.username = username;

    }

    publicString getPassword() {

        returnpassword;

    }

    publicvoidsetPassword(String password) {
            this.password = password;

        }

}
```

And that's it. Our first Hello World Struts2 Application is now ready.

Execute the application in Eclipse and run it in your favorite browser.

Login page

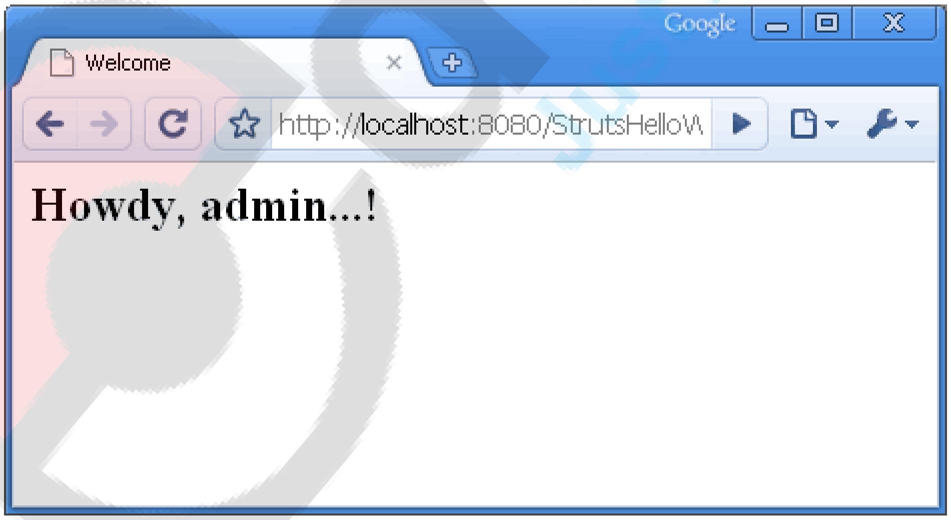

Struts 2 - Login Application ...
http://localhost:8080/StrutsHelloW
Login

Struts 2 - Login Application

Username:

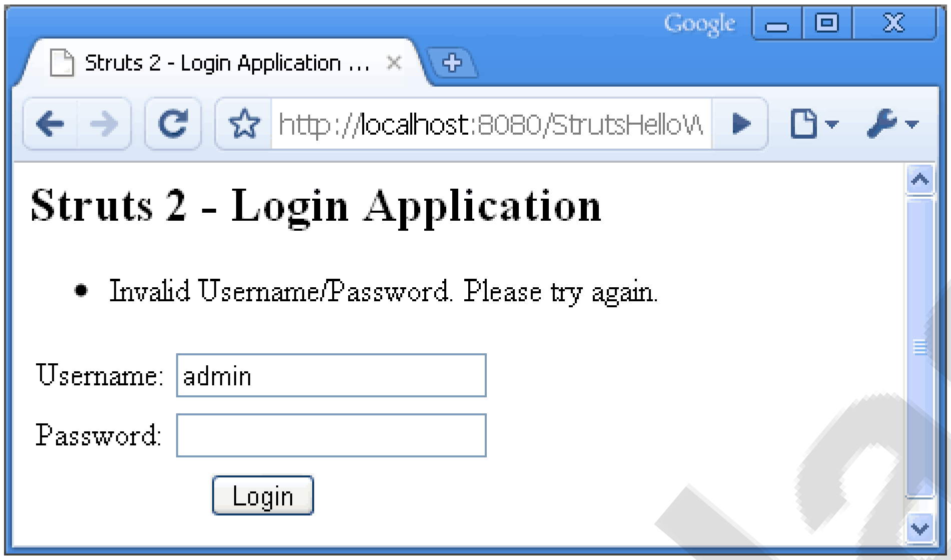
Password:

Welcome page



Howdy, admin...!

Login page with error



Struts 2 - Login Application ...
http://localhost:8080/StrutsHelloW

Struts 2 - Login Application

- Invalid Username/Password. Please try again.

Username: admin

Password:

Login