**About Struts 2.x**

Apache Struts is an open-source framework that is used for developing Java web application. Originally developed by the programmer and author Craig R. McClanahan, this was later taken over by the Apache Software Foundation in 2002. Struts have provided an excellent framework for developing application easily by organizing JSP and Servlet based on HTML formats and Java code. Strut1 with all standard Java technologies and packages of Jakarta assists to create an extensible development environment.

However, with the growing demand of web application, Strut 1 does not stand firm and needs to be changed with demand. This leads to the creation of Strut2, which is more developer friendly with features like Ajax, rapid development and extensibility.

Struts is a well-organized framework based on **MVC** architecture. In **Model-View-Controller Architecture**, Model stands for the business or database code, View represents the page design code and the Controller for navigational code.

Developing web application using struts frame work is fairly complex, but it eases things after it is setup. It encourages software development following the MVC design pattern. Many web applications are JSP-only or Servlets-only. With JSP and Servlets, Java code is embedded in the HTML code and the Java code calls println methods to generate the HTML code respectively. Both approaches have their advantages and drawbacks; Struts gathers their strengths to get the best of their association

Developing web application using struts frame work is fairly complex, but it eases things after it is setup. It encourages software development following the MVC design pattern. Many web applications are JSP-only or Servlets-only. With JSP and Servlets, Java code is embedded in the HTML code and the Java code calls println methods to generate the HTML code respectively. Both approaches have their advantages and drawbacks; Struts gathers their strengths to get the best of their association

**Why Struts 2.x Framework ?**

The new version Struts 2.0 is a combination of the Sturts action framework and Webwork. According to the Struts 2.0.1 release announcement, some key features are:

* **Simplified Design** - Programming the abstract classes instead of interfaces is one of design problem of struts1 framework that has been resolved in the struts 2 framework. Most of the Struts 2 classes are based on interfaces and most of its core interfaces are HTTP independent. Struts 2 Action classes are framework independent and are simplified to look as simple POJOs. Framework components are tried to keep loosely coupled.
* **Simplified Actions** - Actions are simple POJOs. Any java class with execute() method can be used as an Action class. Even we don't need to implement interfaces always. Inversion of Control is introduced while developing the action classes. This make the actions to be neutral to the underlying framework.
* **No more ActionForms** - ActionForms feature is no more known to the struts2 framework. Simple JavaBean flavored actions are used to put properties directly. No need to use all String properties.
* **Simplified testability** - Struts 2 Actions are HTTP independent and framework neutral. This enables to test struts applications very easily without resorting to mock objects.
* **Intelligent Defaults** - Most configuration elements have a default value which can be set according to the need. Even there are xml-based default configuration files that can be overridden according to the need.
* **Improved results** - Unlike ActionForwards, Struts 2 Results provide flexibility to create multiple type of outputs and in actual it helps to prepare the response.
* **Better Tag features** - Struts 2 tags enables to add style sheet-driven markup capabilities, so that we can create consistent pages with less code. Struts 2 tags are more capable and result oriented. Struts 2 tag markup can be altered by changing an underlying stylesheet. Individual tag markup can be changed by editing a FreeMarker template. Both JSP and FreeMarker tags are fully supported.
* **Annotations Introduced:** Applications in struts 2 can use Java 5 annotations as an alternative to XML and Java properties configuration. Annotations minimize the use of xml.
* **Stateful Checkboxes** - Struts 2 checkboxes do not require special handling for false values.
* **QuickStart** - Many changes can be made on the fly without restarting a web container.
* **Customizing Controller** - Struts 1 lets to customize the request processor per module, Struts 2 lets to customize the request handling per action, if desired.
* **Easy Spring Integration** - Struts 2 Actions are Spring-aware. Just need to add Spring beans.
* **Easy Plugins** - Struts 2 extensions can be added by dropping in a JAR. No manual configuration is required!
* **AJAX Support** - The AJAX theme gives interactive applications a significant boost. The framework provides a set of tags to help you ajaxify your applications, even on Dojo.

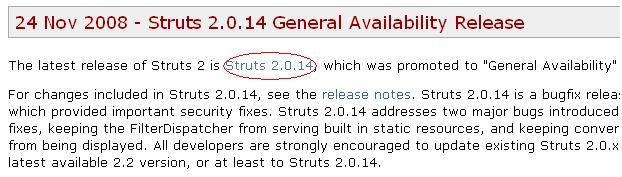
**Installation of Struts 2 Framework**

In this tutorial you will learn how to setup development enviroment for Struts 2 Framework.

**Struts 2 Framework** - A quick tutorial to install Struts 2 Framework on your development environment.

**Downloading Struts 2 Framework:** It is available for downloading straight form its official site. The latest version of this framework for this day is 2.2.1

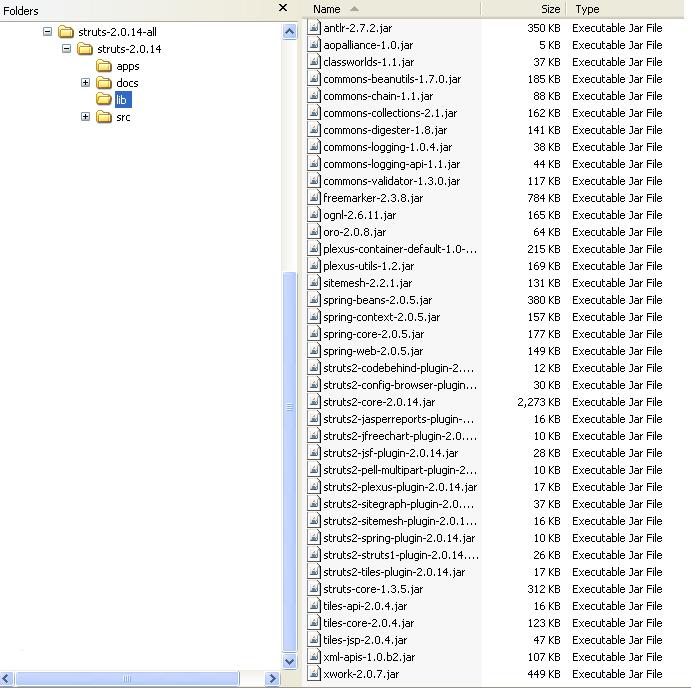
* Simply go to <http://struts.apache.org/>
* Choose ***Struts-2.0.14*** Download option as shown on the picture below:



* Choose ***Struts-2.0.14*** Framework download, click it with your right mouse button and choose "save as" option. as shown on the picture.
* **Extracting Struts 2 download:** Now extract the downloaded file (***struts-2.0.14-all.zip***) into your favorite directory with Winzip/Winrar or an other extracting program u have installed on your computer. Depending upon the system speed it will take around 3 minutes.

Once extract go to the extracted folder and you will find many files.

Here is the screen shot of the directory:



**Struts 2 Annotation Tutorial 1**

In this example you will learn how to use Annotations in Struts 2 and see how the Convention plug-in helps us create an application with minimal configuration.

We will learn annotations in struts 2 using the hello user example. In this example we will get the user name and display a welcome message to the user. There are two versions of this example, in the first one we will see how to do this by using the intelligent defaults provided by the struts 2 framework. We will not do any configuration in this example except the deployment descriptor.

The example is created using ecilpse. The war file of this example is also provided at the end of this tutorial so that you can try it yourself.

So lets start, you need to have the following jar files in the WEB-INF/lib directory.

commons-fileupload-1.2.1

commons-io-1.3.2

commons-logging-1.1

freemarker-2.3.13

junit-3.8.1

ognl-2.6.11

spring-test-2.5.6

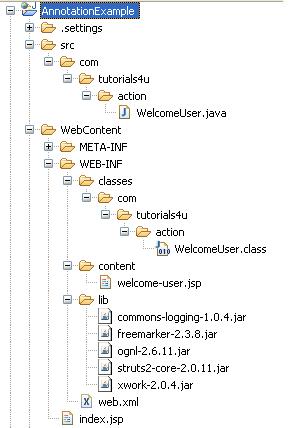
struts2-convention-plugin-2.1.6

struts2-core-2.1.6

xwork-2.1.2

You can definitely save a lot of time by having the correct versions of these jar files in the lib directory. The *struts2-convention-plugin-2.1.6* jar file is needed if your are using annotations.

This is the directory structure of the hello user example.



Now we will create the index page. This page is simple, we use the struts tags to create the page. The *textfield* tag is used to create the textfiled and the *submit* tag is used to create the submit button. The index.jsp page contains the following code.

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

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

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Hello World</title>

</head>

<body>

<s:form action="welcome-user" >

<s:textfield name="userName" label="User Name" />

<s:submit />

</s:form>

</body>

</html>

Note the URL value of the action attribute in the form tag. In the end we will see how everything relates together.

We compose the welcome message in the *execute() method* of the WelcomeUser class and we return "*success*". The WelcomeUser class contains the following code.

package com.tutorials4u.action;

import com.opensymphony.xwork2.ActionSupport;

public class WelcomeUser extends ActionSupport{

private String userName;

private String message;

public String execute() {

message = "Welcome " + userName;

return SUCCESS;

}

public void setUserName(String userName) {

this.userName = userName;

}

public void setMessage(String message) {

this.message = message;

}

public String getUserName() {

return userName;

}

public String getMessage() {

return message;

}

}

Note the class name, can you find any similarities between the action URL and the class name? if yes got the concept, if no don't worry you will learn what it is in the coming pages.

We display the welcome message to the user using the *welcome-user.jsp* page. The content of the page is very simple, we just display the message. The important thing to note here is the name of the page.

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Welcome User</title>

</head>

<body>

<h1>${message}</h1>

</body>

</html>

Now we will configure the web.xml for the struts 2 framework. We need to specify the filter and the filter mapping here. Except this there is no need to have any other XML configuration files.

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" id="WebApp\_ID" version="2.5">

<display-name>Struts2Example1</display-name>

<filter>

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

<filter-class>

org.apache.struts2.dispatcher.ng.filter. StrutsPrepareAndExecuteFilter

</filter-class>

</filter>

<filter-mapping>

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

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

</filter-mapping>

<welcome-file-list>

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

</welcome-file-list>

</web-app>

Now the coding part is complete. You can run the example by using the following URL "***http://localhost:8080/AnnotationExample/***"



Enter a user name and submit the form, you will see the following welcome-user.jsp page.



The example works fine. Now lets see how the example works.

The **Convention plug-in** is the one which does everything in the background. The Convention plug-in does the following things.

* By default the Convention plug-in looks for the action classes inside the following packages **strut, struts2, action or actions**. Here our package name is com.tutorials4u.**action**. Any package that matches these names will be considered as the root package for the Convention plug-in.
* The action class should either implement **com.opensymphony.xwork2.Action** interface or the name of the action class should end with **Action**. Here we extend our WelcomeUser class from **com.opensymphony.xwork2.ActionSupport** which inturn implements com.opensymphony.xwork2.Action.
* The Convention plug-in uses the action class name to map the action URL. Here our action class name is *WelcomeUser* and the URL is *welcome-user*. The plug-in converts the camel case class name to dashes to get the request URL.
* Now the Convention plug-in knows which Action class to call for a particular request. The next step is to find which result to forward based on the return value of the execute method. By default the Convention plug-in will look for result pages in **WEB-INF/content** directory.
* Now the Convention plug-in knows where to look for results, but it doesn't know which file to display inside the content directory. The Convention plug-in finds this with the help of the result code returned by the Action class. If "*success*" is returned then the Convention plug-in will look for a file name *welcome-user-success.jsp or welcome-user.jsp . It need not be a jsp file it can be even a velocity or freemaker files.* If the result value is "input" it will look for a file name *welcome-user-input.jsp or welcome-user-input.vm or welcome-user-input.ftl.*

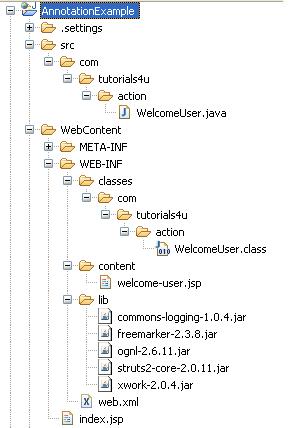
**Struts 2 Annotation Tutorial 2**

In this annotation example you will learn how to use the Action and Result annotations and see how to change the default location of the result pages.

This example is the continuation of the previous annotation example. If you are new to Struts 2 annotations then go through that example first ( [Struts 2 Annotations - Part 1](http://www.tutorials4u.net/struts2-tutorial/struts-2-annotation-example-1.html) ). Here we will see the same hello user example with the following changes.

* Our Action class ends with the world **Action** and does not implement com.opensymphony.xwork2.Action.
* We use **/results** directory for storing our result pages instead of WEB-INF/content.

The directory structure of the hello user example is shown below.



Our WelcomeUserAction class is a simple pojo class. The important thing to note here is that our Action classs name ends with the word **Action**.

01.package com.tutorials4u.action;

02.

03.import org.apache.struts2.convention.annotation.Action;

04.import org.apache.struts2.convention.annotation.Result;

05.

06.public class WelcomeUserAction {

07.    private String userName;

08.    private String message;

09.

10.

11.    @Action(value="/welcome",results={ @Result(name="success",location="/results/successPage.jsp")})

12.    public String execute() {

13.        message = "Welcome " + userName + " !";

14.        return "success";

15.    }

16.

17.    public void setUserName(String userName) {

18.        this.userName = userName;

19.    }

20.

21.    public void setMessage(String message) {

22.        this.message = message;

23.    }

24.

25.    public String getUserName() {

26.        return userName;

27.    }

28.

29.    public String getMessage() {

30.        return message;

31.    }

32.}

Here we use Action and Result annotations just to show you how to use them, for simple example like this you can use the intelligent defaults provided by the Convention plug-in.

The Convention plug-in uses the Action class name to map the action URL. The Convention plug-in first removes the world Action at the end of the class name and then converts the camel case name to dashes. So by default our *WelcomeUserAction* will be invoked for the request URL *welcome-user*. But if you want the Action to be invoked for a different URL then you can do this by using the **Action annotation**.

The value of the Action annotation is *"/welcome*", this means that the action will be invoked for the request URL "*/welcome*". You can change the default action and URL mapping using the Action annotation.

Now based on the result code from action the Convention plug-in will look for the result name **welcome-resultcode** in the directory WEB-INF/content. You can change this to a different location by setting the property **struts.convention.result.path** to a new value in the Struts properties file. In this example we store the result pages in /results directory.

1.struts.properties file

2.-----------------------

3.struts.convention.result.path=/results

Our result page name is *successPage.jsp*, the Convention plug-in will look for a page like welcome.jsp ( the file can even be a freemaker or velocity file ) since our URL is "*/welcome*". In this case it will give an error, if we are not specifying which result it should invoke when the result is "*success*". To do this we use the Result annotation.

The **Result annotation** maps the result code with the result page. Here the result code "*success*" is mapped to the result "*/results/successPage.jsp*".

The annotations needs to be specified only when you are not using the default naming conventions, if you use them you can keep writing action classes and result pages without any configuration and the framework know exactly when to invoke them.

|  |  |
| --- | --- |
| **Struts 2 Hello World Application - Getting started with Struts 2**  In this example you will see how to create a simple Hello World Application in Struts 2.  In this tutorial we will see how to create a simpe Struts 2 Hello World Application. The following files are needed to create a Hello World Application.   * *web.xml* * *struts.xml* * *HelloWorld.java* * *index.jsp* * *success.jsp* | |
| The following picture shows the directory structure of the Hello World application.  http://www.tutorials4u.net/struts2-tutorial/images/HelloWorldPic3.gif   * Create New Project: Dynamic Web Project and give a name to your project and the location to save your project. For this example, I gave HelloWorld as my project name. * Second step will be including JAR files required by Struts 2 framework to our project's WEB-INF/lib folder. You can either download below JARs separately or simple copy them from the lib folder of struts2-blank-application provided by Struts 2 website. Note that the version numbers in the JAR files are the latest ones when this article written. You may use the latest JARs if they are available.   + *commons-logging-1.0.4.jar*   + *freemarker-2.3.8.jar*   + *ognl-2.6.11.jar*   + *struts2-core-2.0.11.jar*   + *xwork-2.0.4.jar* * Next step will be configuring struts 2 filter in web.xml file. Have a look at the below sample configuration.   **web.xml**  web.xml is used to configure the servlet container properties of the hello world appliation. The filter and the filter-mapping elements are used to setup the Struts 2 FilterDispatcher. The filter is mapped to the URL pattern "*/\**". This means all the incoming request that targets to the Struts 2 action will be handled by FilterDispatcher class.  01.<filter>  02.    <filter-name>struts2</filter-name>  03.    <filter-class>org.apache.struts2.dispatcher.FilterDispatcher </filter-class>  04.</filter>  05.<filter-mapping>  06.  07.    <filter-name>struts2</filter-name>  08.    <url-pattern>/\*</url-pattern>  09.</filter-mapping>  10.<welcome-file-list>  11.    <welcome-file>index.jsp</welcome-file>  12.  13.</welcome-file-list>   * The gateway for our hello world application is index.jsp file. The index.jsp file should be mentioned in web.xml as shown above. * Now we will create the very important struts.xml file which glues our action class with the presentation JSP file. Since Struts 2 requires struts.xml to be present in classes folder, we will create stuts.xml file inside the source folder, so that when building the WAR file, struts.xml will be put in classes folder. Below is the code for struts.xml file.   **struts.xml**  01.<struts>  02.    <package name="default" extends="struts-default">  03.  04.        <action name="HelloWorld" class="com.tutorials4u.helloworld.HelloWorld">  05.            <result name="SUCCESS">/success.jsp</result>  06.        </action>  07.    </package>  08.  09.</struts>   * Now we will create a struts action class HelloWorld.java. To do this, create a new package com.tutorials4u.helloworld under the project's source folder and inside the helloworld package, create a new class file and name it *HelloWorld.java*.   **HelloWorld.java**  As you see the HelloWorld class is very simple. It contains two properties one for the user name and the other for displaying the message. Below is the source code for this class file.  01.import com.opensymphony.xwork2.ActionSupport;  02.  03.public class HelloWorld extends ActionSupport {  04.  05.    private String message;  06.  07.    private String userName;  08.  09.    public HelloWorld() {  10.    }  11.  12.    public String execute() {  13.        setMessage("Hello " + getUserName());  14.        return "SUCCESS";  15.    }  16.  17.    public String getMessage() {  18.        return message;  19.    }  20.  21.    public void setMessage(String message) {  22.        this.message = message;  23.    }  24.  25.    public String getUserName() {  26.        return userName;  27.    }  28.  29.    public void setUserName(String userName) {  30.        this.userName = userName;  31.    }  32.  33.}  In the execute() method of the HelloWorld action we compose the message to be displayed. Note we need not have a seperate form bean like struts 1 to access the form data. We can have a simple java class as action. The action need not extend any class or implement any interface. The only obligation is that you need to have an execute() method which returns a String and has a public scope.   * We have our action class is ready. Now it's the time to create the presentation page, i.e JSP. Create a new JSP file index.jsp inside the WebContent folder and type in below code in this JSP file.   **index.jsp**  The Struts 2 UI tags are simple and powerful. To use the struts tags in the jsp page the following taglib directive should be included.  01.<%@taglib uri="/struts-tags" prefix="s" %>  02.  03.<html>  04.<head>  05.<title>Hello World</title>  06.</head>  07.<body>  08.    <s:form action="HelloWorld" >  09.  10.        <s:textfield name="userName" label="User Name" />  11.        <s:submit />  12.    </s:form>  13.</body>  14.  15.</html>   * **success.jsp**   In the success page we display the "*Hello User*" message using the property tag.  01.<%@taglib uri="/struts-tags" prefix="s" %>  02.<html>  03.<head>  04.  05.<title>Hello World</title>  06.</head>  07.<body>  08.    <h1><s:property value="message" /></h1>  09.</body>  10.  11.</html>   * Now we are good to go. Right click on the project name and select Run As --> Run on Server. Select the Server on which you want to deploy our HelloWorld Application. Now point your browser to the URL "***http://localhost:8080/HelloWorldExample/*** "Your first struts 2 application is delivering the JSP page with a hello message to you. The index page will be displayed.   http://www.tutorials4u.net/struts2-tutorial/images/HelloWorldPic1.JPG  Enter the user name and submit the form. Hello user name message will be displayed.  http://www.tutorials4u.net/struts2-tutorial/images/HelloWorldPic2.JPG  **How the Code Works**  Your browser sends a request to the web server for the URL ***http://localhost:8080/HelloWorldExample/***   * The container receives from the web server a request for the resource HelloWorld.action. According to the settings loaded from the web.xml, the container finds that all requests are being routed to org.apache.struts2.dispatcher.FilterDispatcher, including the \*.action requests. The FilterDispatcher is the entry point into the framework. * The framework looks for an action mapping named "HelloWorld", and it finds that this mapping corresponds to the class "HelloWorld". The framework instantiates the Action and calls the Action's execute method. * The execute method sets the message and returns SUCCESS. The framework checks the action mapping to see what page to load if SUCCESS is returned. The framework tells the container to render as the response to the request, the resource success.jsp. * As the page success.jsp is being processed, the ***<s:property value="message" />*** tag calls the getter getMessage of the HelloWorld Action, and the tag merges into the response the value of the message. * A pure HMTL response is sent back to the browser.   **What to Remember**  The framework uses Actions to process HTML forms and other requests. The Action class returns a result-name such as SUCCESS, ERROR, or INPUT. Based on the mappings loaded from the struts.xml, a given result-name may select a page (as in this example), another action, or some other web resource (image, PDF).  When a server page is rendered, most often it will include dynamic data provided by the Action. To make it easy to display dynamic data, the framework provides a set of tags that can be used along with HTML markup to create a server page. |  |