MVC (Model - View - Controller):

Model - responsible for business logic and managing data of the application.

View - Responsible for presenting result data to the user.

Controller - controls the interactions between the Model and View. Receives request from the user and then works with model to prepare any data needed by the view. then send the response to the view layer.

***Component and functional differences between struts 1.x and struts 2.x***

In struts 1.x front controller is ActionServlet

In 2.x front controller is FilterDispatcher

In struts 1.x we have RequestProcessor class

In 2.x we have Interceptors instead RequestProcessor will see about this concept later just remember as of now

In struts 1.x we have multiple tag libraries like, html, logic, bean..etc

In 2.x we do not have multiple libraries, instead we have single library which includes all tags

In struts 1.x the configuration fine name can be [any name].xml and we used to place in web-inf folder

In 2.x the configuration file must be struts.xml only and this must be in classes folder

In struts 1.x we have form beans and Action classes separately

In 2.x form bean, Action classes are combinedly given as Action class only, of course we can take separately if we want ;)

In struts 1.x properties file must be configured in struts-config.xml

But in 2.x we need to configure our resource bundle(s) in struts.properties file

In struts 1.x we have programmatic and declarative validations only

In 2.x we have annotations support too along with programmatic and declarative validations

**Functional Differences**

In struts 1.x declarative validations are done by using validation frame work

In 2.x, declarative validations are done by using xwork2 frame work by webwork the reason being, its support valuations through Annotations

In struts 1.x an Action class is a single ton class, so Action class object is not a thread safe, as a programmer we need to make it as thread safe by applying synchronization

In 2.x an Action class object will be created for each request, so it is by default thread safe, so we no need to take care about safety issues here

In struts 1.x we have only jsp as a view technology

In 2.x we have support of multiple view technologies like velocity, Freemarker, jasper reports, jsp bla bla

In struts 1.x Action class is having servlet dependency, because in execute() method accepts req, res parameter right ! so.

In 2.x Action class doesn’t have any servlet dependency, because its execute() method doesn’t accepts any parameters, however we can access all servlet objects with dependency injection :-)

***Struts2 framework:***

It is used for full development life cycle.

webwork2 of opensymphont + struts1 = struts.

In Struts 2 **FilterDispatcher** does the job of Controller.

In Struts 2 the model is implemented by the **Action** component. Model contains the data and the business logic.

In Struts 2 the view is commonly implemented using **JSP, Velocity Template, Freemaker** or some other presentation-layer technology.

***Struts2 features:***

* Configurable MVC components In struts 2 framework, we provide all the components (view components and action) information in struts.xml file. If we need to change any information, we can simply change it in the xml file.
* POJO based actions In struts 2, action class is POJO (Plain Old Java Object) i.e. a simple java class.
* AJAX support Struts 2 provides support to ajax technology. So a part of the page will be changed only and page will not reload. So it makes the performace fast.
* Integration Support We can simply integrate the struts 2 application with hibernate, spring, tiles etc. frameworks.
* Various Result Types We can use JSP, freemarker, velocity etc. technology as the result in struts2.

**ActionInvocation:**

* The ActionInvocation represents the execution state of an action. It holds the action and interceptors objects.
* The struts framework provides ActionInvocation interface to deal with ActionInvocation. It provides many methods, some of them can be used to get the instance of ValueStack, ActionProxy, ActionContext, Result etc.
* Eg. public ValueStack getStack() returns the instance of ValueStack , public Action getAction() returns the instance of Action associated with this ActionInvocation.
* FilterDispatcher determine which Struts 2 action to invoke.
* The framework creates an instance of this action and associate it with the newly created instance of the ActionInvocation.
* In Struts 2 the invocation of action should pass through a series of interceptors as defined in the application's XML file.
* The framework calls the ActionInvocations invoke() method to start the execution of the action.
* Each time the invoke() method is called, ActionInvocation consults its state and executes whichever interceptor comes next.
* ActionInvocation hands control over to the interceptor in the stack by calling the interceptors intercept() method.
* The intercept() method of the interceptor intern calls the invoke() method of the ActionInvocation till all the interceptors are invoked, in the end the action itself will be called and the corresponding result will be returned back to the user.

***ActionContext***

The ActionContext is a container of objects in which action is executed. The values stored in the ActionContext are unique per thread (i.e. ThreadLocal). So we don't need to make our action thread safe.

**Configuration files:**

1.web.xml - This is the entry point of any web application and entry point of struts2 application is FilterDispatcher(PrepareAndExecute since version 2.x) which is defined in web.xml file

2.strut.xml - The struts.xml file contains the configuration information that you will be modifying as actions are developed. This file can be used to override default settings for an application , for example struts.devMode = false and other settings which are defined in property file.

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

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"<http://struts.apache.org/dtds/struts-2.0.dtd>">

<struts>

<constant name="struts.enable.DynamicMethodInvocation" value="false" />

<constant name="struts.devMode" value="false" />

<constant name="struts.custom.i18n.resources" value="myapp" />

<package name="default" extends="struts-default" namespace="/">

<action name="login" class="com.dineshonjava.struts2.login.LoginAction">

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

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

</action>

</package>

</struts>

***1.Package element:***

Package tag in struts is used for separation and modularization of the configuration. This is very useful when you have large project and project is divided into different modules.

You can have or or more packages based on the modules and store associated actions in the appropriate package.

Attributes of package elements:

* name name is must for defining any package.
* namespace It is an optional attribute of package. If namespace is not present, / is assumed as the default namespace. In such case, to invoke the action class, you need this URI: /actionName.action If you specify any namespace, you need this URI: /namespacename/actionName.action (Struts 2 Namespace is a new concept to handle the multiple modules by given a namespace to each module. In addition, it can used to avoid conflicts between same action names located at different modules.)
* extends The package element mostly extends the struts-default package where interceptors and result types are defined. If you extend struts-default, all the actions of this package can use the interceptors and result-types defined in the struts-default.xml file.

***2) action element:***

The action is the sub-element of package and represents an action.

Attributes of action element-

* name name is must for defining any action.
* class class is the optional attribute of action. If you omit the class attribute, ActionSupport will be considered as the default action.
* method It is an optional attribute. If you don't specify method attribute, execute method will be considered as the method of action class.

***3) result element:***

It is the sub element of action that specifies where to forward the request for this action.

Attributes of result element-

* name is the optional attribute. If you omit the name attribute, success is assumed as the default result name.
* type is the optional attribute. If you omit the type attribute, dispatcher is assumed as the default result type.

3.struts-config.xml - The struts-config.xml configuration file is a link between the View and Model components in the Web Client

4.struts.properties - The values configured in this file will override the default values configured in default.properties which is contained in the struts2-core-x.y.z.jar distribution. Actually all of the properties contained within the struts.properties configuration file can also be configured in the web.xml using the init-param, as well using the constant tag in the struts.xml configuration file.

***Interceptors:***

it is used in the following scenarios:

1.Providing preprocessing logic before the action is called.

2.Providing postprocessing logic after the action is called.

Many of the features provided in the Struts2 framework are implemented using interceptors; examples include exception handling, file uploading, lifecycle callbacks and validation etc.

Custom interceptor:

To create the custom interceptor u need to extend the interceptor interface.

public interface Interceptor extends Serializable{

void destroy();

void init();

String intercept(ActionInvocation invocation)

throws Exception;

}

If you have no need for initialization or cleanup code, the AbstractInterceptor class can be extended to write a custom interceptor.

public class MyInterceptor extends AbstractInterceptor {

public String intercept(ActionInvocation invocation)throws Exception{

/\* let us do some pre-processing \*/

String output = "Pre-Processing";

System.out.println(output);

/\* let us call action or next interceptor \*/

String result = invocation.invoke();

/\* let us do some post-processing \*/

output = "Post-Processing";

System.out.println(output);

return result;

}

}

The framework itself starts the process by making the first call to the ActionInvocation object's invoke(). Each time invoke() is called, ActionInvocation consults its state and executes whichever interceptor comes next. When all of the configured interceptors have been invoked, the invoke() method will cause the action itself to be executed.

***Result types:***

1,Dispatcher result type:

Default result type of struts2 is dispatcher. It is used to forword servlet, JSP, HTML page, and so on to the View.It uses the RequestDispatcher.forward() method.

<result name="success" type="dispatcher">

<param name="location">

/HelloWorld.jsp

</param >

</result>

2.Freemaker result type

3.redirect result type

The redirect result type calls the standard response.sendRedirect() method, causing the browser to create a new request to the given location.

***ValueStack:***

For every request , the framework creates one action object. ValueStack is simply a stack used to store the action objects and other model objects.

***ActionContext:***

It is used to store all the other objects. for example session, request, application objects.

***OGNL(Object-Graph Navigation Language) :***

The topmost object dealing with OGNL is a Map called as context map or context.

The framework sets the OGNL context to be our ActionContext, and the value stack to be the OGNL root object.

OGNL uses # notation to access the actioncontext objects.

OGNL has direct access to the value stack objects since it is the root object of OGNL.

OGNL uses valuestack for getting and setting properties of java objects and for getting and setting the value of a property.

* The Object Graph Navigation Language (OGNL) is an expression language. It simplifies the accessibility of data stored in the ActionContext.
* Object-Graph Navigation Language (OGNL) is a powerful expression language that is used to reference and manipulate data on the ValueStack.
* OGNL help in data transfer and type conversion.
* OGNL expression language provides simplified syntax to reference java objects.
* OGNL is used to bind the java-side data properties to the string-based view layer.

The struts framework sets the ValueStack as the root object of OGNL. Notice thaxt action object is pushed into the ValueStack. We can direct access the action property.

Objects in the ActionContext are referred using the pound symbol, however, the objects in the value stack can be directly referenced, for example if employee is a property of an action class then it can get referenced as follows:

<s:property value="name"/>

instead of

<s:property value="#name"/>

If you have an attribute in session called "login" you can retrieve it as follows:

<s:property value="#session.login"/>

***ResourceBundle:***

Resource bundle is a flat file that contains key-value pair.

Message resources(Resource bundle) provide a simple way

\* to put text in a view page that is the same through out your application

\* to create form field labels

\* and mainly to change text to a specific language based on the user's locale (i18n). It is mainly used to adapting the locale language. The key values resource bundle files need to be developed for each of the supporting locale.

***Internationalization(i18n):***

(a.k.a. Globalization, a.k.a. Enabling) Designing and developing a software product to function in multiple locales. This process involves identifying the locales that must be supported, designing features which support those locales, and writing code that functions equally well in any of the supported locales.

***Localization(l10n):***

Modifying or adapting a software product to fit the requirements of a particular locale. This process includes (but may not be limited to) translating the user interface, documentation and packaging, changing dialog box geometries, customizing features (if necessary), and testing the translated product to ensure that it still works (at least as well as the original).

Sample prop file:

personBean.firstName=First name

personBean.lastName=Last name

personBean.age=Age

personBean.email=Email

thankyou=Thank you for registering %{personBean.firstName}.

We can retreive the resource bundle values in the following ways.

1. Action class

In Action class, you can extends the ActionSupport and get the resource bundle via getText(‘key’) function.

public class LoginAction extends ActionSupport{

...

public void validate(){

if("".equals(getUsername())){

addFieldError("username", getText("username.required"));

}

}

}

2. property tag

In property tag, use the getText(‘key’).

<s:property value="getText('username')" />

3. text tag

In text tag, set the key in “name” attribute.

<s:text name="username" />

4. Key attribute

The Key attribute of UI component has special function, check detail in this key attribute example.

<s:textfield key="username" />

5. I18n tag

This i18n tag can get the message from a specified resource bundle that declared in the “name” attribute. In this example, it ask to get the ‘username’ message from com/mkyong/user/package.properties file.

<s:i18n name="com.mkyong.user.package" >

<s:text name="username" />

</s:i18n>

Resource bundle is searched in the following order:

**1.ActionClass.properties** - you may associate a message resource property file with each Struts 2 Action class by creating a properties file with the same name as the Action class and having the .properties extension

2.Interface.properties

3.BaseClass.properties

4.ModelDriven’s model

**5.package.properties -** if you want a properties file with keys and values that can be referenced from multiple view pages

6.Search up the i18n message key hierarchy itself

**7.Global resource properties -** You can also specify a global property file in struts.xml. \* The keys and values defined in that property file will be available to all the view pages that are rendered after executing an Action class.

\* To inform the Struts 2 framework about the global.properties file add the follow node to struts.xml after the constant name="struts.devmode" node.

Specify Global Property File In struts.xml

<constant name="struts.custom.i18n.resources" value="global" />

Main searching order is : ActionClass.properties, package.properties and Global resource properties.

***For internalization:***

Step 1:pass the locale value from the view

<s:a href="changelang?lang=en" >English</s:a>

<s:a href="changelang?lang=fr" >France</s:a>

<s:a href="changelang?lang=ja" >Japanese</s:a>

step 2: set that locale value in action context

Locale locale = new Locale(lang);

ActionContext.getContext().setLocale(locale);

**step 3:** then the struts framework will look for the property file bassed on locale.

if it is es(english) it will look for default prop file. for eg: global.properties.

if it is fr(france) it will look for prop file named global\_fr.properties.

Based on the locale we have to create our properties file content for localization.

***Exception Handlers:***

Exception handlers allows us to define the exception handling procedure on global and local basis. Framework catches the exception and then displays the page of our choice with appropriate message and exception details.