**How to write Struts1 application?**

1). Write JSP Page

2). Write Form Bean Class

Extends Action Form Class

Override reset(am,req) & validate(am,req) method

**Public void reet(ActionMapping am,HttpServletRequest req)**

**Public ActionErrors validate(ActionMapping am,HttpServletRequest req)**

Generate setter & getter methods

3). Configure the struts-config.xml file.

4). Write Action Class

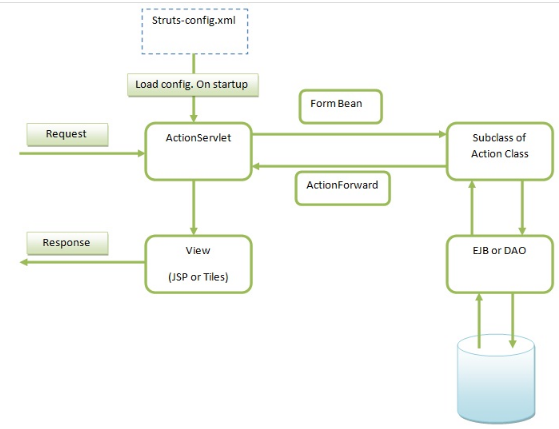
Extends Action class

Override execute() method

**Public ActionForward execute(ActionMapping am, ActionForm af,HttpServletRequest req, HttpServletResponse res)**

5). Configure Action class to struts-config.xml file.

**Flow of Struts 1 Application**



The following steps show the work flow of Struts 1 framework:

1. When first time user request comes from the browser, ActionServlet invoked which reads struts-config.xml configuration file and creates configuration objects. struts-config.xml contains information about the FormBeans, that are subclasses of ActionForm, Actions and view that can be send as response to the browser.

2. In the second step ActionServlet instantiate the specified form bean and puts the value of the form attributes, submitted by the browser, to form bean.

3. Then ActionServlet calls the Action class’s execute method, which returns ActionForward instance. ActionForward instance contains the detail of the view to be return as response.

4. ActionServlet renders the view and return to the browser as response.

**What are the main classes which are used in struts application?**

Main classes in Struts Framework are:

**Action servlet:** it’s a backbone of web application it’s a controller class responsible for handling the entire request.

**Action class:** using Action classes all the business logic is developed us call model of the application also.

**Action Form:** it’s a java bean which represents our forms and associated with action mapping. And it also maintains the session state its object is automatically populated on the server side with data entered from a form on the client side.

**Action Mapping**: using this class we do the mapping between object and Action.

**ActionForward:** this class in Struts is used to forward the result from controller to destination.

**What is Form Bean in Struts 1?**

A form bean is a type of Java bean. A form bean is an instance of a subclass of an ActionForm class and other FormBean class, which stores HTML form data from a submitted client request or that can store input data from a Struts action link that a user clicked.

**You can divide Form Beans into two types.**

Static Forms

Dynamic Forms

**Static Forms :**

There are three types of Static Forms.

ActionForm

ValidatorForm

ValidatorActionForm

When you are using the Static Form, you need to do the following things.

You need to write a Java class by extending one of the above mentioned Static Forms.

You need to declare the fields same as JSP forms.

You need to write the set/get methods for all the fields.

If you want to validate form data you need to override the validate() method in the Form Bean class.

If you want to reset your form fields ,you need to override the reset() method in the Form Bean class.

**Dynamic Forms :**

There are three types of Dynamic Forms.

DynaActionForm

DynaValidatorForm

DynaValidatorActionForm

**When you are using the Dynamic Form, you need to do the following things**

Write the field declaration inside the struts configuration file(No need to write Java class with private field and set/get method).

DynamicForms will reset automatically to default values(No need to override reset method).

Dynamic forms will use validation framework for the form validations (No need to override the validate method)

**What is the Difference between DispatchAction and LookupDispatchAction in Struts Framework?**

|  |  |
| --- | --- |
| **Dispatch Action** | **LookupDispatchAction** |
| It’s a parent class of  LookupDispatchAction | Subclass of Dispatch Action |
| DispatchAction provides a mechanism for grouping a set of related functions into a single action, thus eliminating the need to create separate actions for each function. | An abstract **Action** that dispatches to the subclass mapped executes method. This is useful in cases where an HTML form has multiple submit buttons with the same name. The button name is specified by the parameter property of the corresponding ActionMapping. |
| If not using Internalization functionality then dispatch action is more useful. | Lookup Dispatch Action is useful when we are using Internalization functionality |
| DispatchAction selects the method to execute depending on the request parameter value which is configured in the XML file. | **LookupDispatchAction** looks into the resource bundle file and finds out the corresponding key name. We can map this key name to a method name by overriding the getKeyMethodMap() method. |
| **DispatchAction** is not useful for I18N | LookupDispatchAction is used for I18N. |

**What is the life cycle of ActionForm?**

The lifecycle of ActionForm invoked by the RequestProcessor is as follows:

Retrieve or Create Form Bean associated with Action

"Store" FormBean in appropriate scope (request or session)

Reset the properties of the FormBean

Populate the properties of the FormBean

Validate the properties of the FormBean

Pass FormBean to Action.

**What the Validate () and reset () method does?**

**validate**(): validate method is Used to validate properties after they have been populated, and this, method is Called before FormBean is passed to Action. Returns a collection of ActionError as ActionErrors. Following is the method signature for the validate() method.

public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {

ActionErrors errors = new ActionErrors();

if ( StringUtils.isNullOrEmpty(username) && StringUtils.isNullOrEmpty(password)){

errors.add(ActionErrors.GLOBAL\_ERROR, new ActionError("error.usernamepassword.required"));

}

return errors;

}

**reset**(): reset() method is called by Struts Framework with each request that uses the defined ActionForm. The purpose of this method is to reset all of the ActionForm's data members prior to the new request values being set.

Example :

public void reset(ActionMapping mapping, HttpServletRequest request) {

this.password = null;

this.username = null;

}

**How exceptions are handled in Struts application?**

Exceptions are handled in struts by using any one of the following two ways:

**Programmatically handling**: In this exception are handled by using try and catch block in program. Using this programmer can define how to handle the situation when exception arises.

**Declarative handling**: In this exception handling is done by using the XML file. Programmer defines the exception handling logic in the XML file. There are two ways of defining the exception handling logic in the XML file:

- Global Action Specific Exception Handler Definition.

- Local Action Specific Exception Handler Definition.

**What is the different actions available in Struts?**

The different kinds of actions in Struts are:

Action

DispatchAction

LookupDispatchAction

MappingDispatchAction (1.2)

IncludeAction

ForwardAction

LocaleAction (1.2)

DownloadAction

SwitchAction

**What is role of ActionServlet?**

ActionServlet performs the role of Controller:

Process user requests

Determine what the user is trying to achieve according to the request

Pull data from the model (if necessary) to be given to the appropriate view,

Select the proper view to respond to the user

Delegates most of this grunt work to Action classes

Is responsible for initialization and clean-up of resources

**What is DispatchAction?**

- The DispatchAction enable the programmer to combine together related function or class.

- Using Dispatch Action programmer can combine the user related action into a single UserAction. like add user, delete user and update user.

- DispatchAction execute the action based on the parameter value it receives from the user.

**How to use DispatchAction?**

We can use the Dispatch Action we executing following steps:

- Create a class that extends DispatchAction.

- In a new class, add a method: method has the same signature as the execute() method of an Action class.

- Do not override execute() method.

- Add an entry to struts-config.xml

**What is the difference between ForwardAction and IncludeAction?**

The difference between ForwardAction and InculdeAction are:

- IncludeAction is used when any other action is going to intake that action whereas ForwardAction is used move the request from one resource to another resource.

**What is LookupDispatchAction?**

- The LookupDispatchAction class is a subclass of DispatchAction.

- The LookupDispatchAction is used to call the actual method.

- For using LookupDispatchAction, first we should generate a subclass with a set of methods.

- It control the forwarding of the request to the best resource in its subclass.

- It does a reverse lookup on the resource bundle to get the key and then gets the method whose name is associated with the key into the Resource Bundle.

**LookupDispatchAction is useful** if the method name in the Action is not driven by its name in the front end, but by the Locale independent key into the resource bundle. Since the key is always the same, the LookupDispatchAction shields your application from the side effects of I18N.

**What is the use of ForwardAction?**

- The ForwardAction is used when we want to combine Struts with existing application.

- Used when we want to transfer the control form JSP to local server.

- Used to integrate with struts in order to take benefit of struts functionality, without writing the Servlets again.

- Use to forward a request to another resource in your application.

**What is IncludeAction?**

The IncludeAction is used to integrate the one action file in another action file.

- It is same as ForwardAction but the only difference is that the resource is present in HTTP response.

- Is used to combine the Struts functionality and control into an application that uses Servlets.

- Use the IncludeAction class to include another resource in the response to the request being processed.

**What is SwitchAction?**

The SwitchAction class provides a means to switch from a resource in one module to another resource in a different module. SwitchAction is useful only if you have multiple modules in your Struts application. The SwitchAction class can be used as is, without extending.

**What if <action> element has <forward> declaration with same name as global forward?**

In this case the global forward is not used. Instead the <action> element’s <forward>takes precendence.

**What is DynaActionForm?**

A specialized subclass of ActionForm that allows the creation of form beans with dynamic sets of properties (configured in configuration file), without requiring the developer to create a Java class for each type of form bean.

**What are the steps need to use DynaActionForm?**

Using a DynaActionForm instead of a custom subclass of ActionForm is relatively straightforward. You need to make changes in two places:

· In struts-config.xml: change your <form-bean> to be an org.apache.struts.action.DynaActionForm instead of some subclass of ActionForm

**What are the various Struts tag libraries?**

The various Struts tag libraries are:

· HTML Tags

· Bean Tags

· Logic Tags

· Template Tags

· Nested Tags

· Tiles Tags

**What is the use of <logic:iterate>?**

<logic:iterate> repeats the nested body content of this tag over a specified collection.

<table border=1>

<logic:iterate id="customer" name="customers">

<tr>

<td><bean:write name="customer"

property="firstName"/></td>

<td><bean:write name="customer" property="lastName"/></td>

<td><bean:write name="customer" property="address"/></td>

</tr>

</logic:iterate>

</table>

**What are differences between <bean:message> and <bean:write>**

<bean:message>: is used to retrive keyed values from resource bundle. It also supports the ability to include parameters that can be

substituted for defined placeholders in the retrieved string.

<bean:message key="prompt.customer.firstname"/>

<bean:write>: is used to retrieve and print the value of the bean property. <bean:write> has no body.

<bean:write name="customer" property="firstName"/>

**What is the life cycle of ActionForm?**

The lifecycle of ActionForm invoked by the RequestProcessor is as follows:

· Retrieve or Create Form Bean associated with Action

· "Store" FormBean in appropriate scope (request or session)

· Reset the properties of the FormBean

· Populate the properties of the FormBean

· Validate the properties of the FormBean

· Pass FormBean to Action

**How is the Action Mapping specified ?**  
We can specify the action mapping in the configuration file called struts-config.xml. Struts framework creates ActionMappingobject  
from <ActionMapping> configuration element of struts-config.xml file  
<action-mappings>  
<action path="/submit"   
type="submit.SubmitAction"  
name="submitForm"  
input="/submit.jsp"  
scope="request"  
validate="true">  
<forward name="success" path="/success.jsp"/>  
<forward name="failure" path="/error.jsp"/>  
</action>  
</action-mappings>

**In which method of Action class the business logic is executed ?**

In the execute() method of Action class the business logic is executed.

**public ActionForward execute(ActionMapping mapping,ActionForm form,**

**HttpServletRequest request, HttpServletResponse response) throws Exception ;**

**execute() method of Action class:**

· Perform the processing required to deal with this request

· Update the server-side objects (Scope variables) that will be used to create the next page of the user interface

· Return an appropriate ActionForward object

**What is the difference between RequestAware and ServletRequestAware interface?**

The difference between RequestAware and ServletRequestAware are:

- RequestAware enables programmer with the attributes in the Servlet Request as a map whereas ServletRequestAware enables programmer with HttpServletRequest object.

- ServletRequestAware are more flexible than RequestAware.

- ServletRequestAware makes action class highly coupled with Servlet environment which is not possible in RequestAware.

**Can we have more than one struts config.xml file for a single Struts application?**  
Yes, we can have more than one struts-config.xml for a single Struts application. They can be configured as follows:  
  
<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,  
/WEB-INF/struts-admin.xml,  
/WEB-INF/struts-config-forms.xml  
</param-value>  
</init-param>  
.....  
<servlet>

**What are the sections into which a strut configuration file can be divided?**

Strut configuration file directly relates to struts-config. The five sections into which strut configuration file is divided are:

1. Form bean definition section: This is the section that is required to provide the definition of the form bean that is used to create the form and allow the user to communicate with the server using it.

2. Global forward definition section: it allows the components to be forwarded as global and provide the server with the information about the global variables.

3. Action mapping definition section: this allows the mapping of various actions that can be used to provide the tool for the user to their help.

4. Controller configuration section: this allows the configuration settings to be controlled and filled by the user. The modifications can be done to change the settings that are to be used for controlling the resources.

5. Application Resources definition section: this allows the resources to be defined that are used in the application.

**List the important attribute and elements of action mapping under struts.**

The important attribute and elements of action mapping under struts are as follows :

1. Path: It is the URL path for which the action mapping is used. The path must be unique and can be for either path mapping or suffix mapping.

2. Type: It is the fully qualified class name of the Action.

3. Name: It is the logical name of the Form bean. It helps in deciding which action mappings should use which ActionForms for strut application.

4. Scope: The Scope of the Form bean is either session or request.

5. Validate: Is either true or false. For true, the Form bean is validated on submission, for false, the validation is skipped.

6. Input: When validation error exists the physical page to which the control should be passed is called the input.

7. Forward: When action forward with respect to the name is selected in execute method of action class the control is passed to this page.

**What is Struts2?**

Apache Struts2 is an open source framework to build web applications in Java. Struts2 is based on OpenSymphony WebWork framework. It’s highly improved from Struts1 and that makes it more flexible, easy to use and extend. The core components of Struts2 are Action, Interceptors and Result pages.

Struts2 provides many ways to create Action classes and configure them via struts.xml or through annotations. We can create our own interceptors for common tasks. Struts2 comes with a lot of tags and uses OGNL expression language. We can create our own type converters to render result pages. Result pages can be JSPs and FreeMarker templates.

Struts 2 provides many features that were not in struts 1.

**The important features of struts 2 framework are as follows:**

Configurable MVC components

POJO based actions

AJAX support

Integration support

Various Result Types

Various Tag support

Theme and Template support

**What are the differences between Struts1 and Struts2 or how Struts2 is better than Struts1?**

1). In struts 1.x front controller is ActionServlet

In 2.x front controller is FilterDispatcher

2). 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

3). 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

4). 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

5). 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 ;)

6). 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

7). 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**

8). 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

9). 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

10). 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

11). 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

**What are Struts2 core components?**

Struts2 core components are:

Action Classes

Interceptors

Result Pages, JSP of FreeMarker templates

ValueStack, OGNL and Tag Libraries.

**Which design pattern is implemented by Struts2 interceptors?**

Struts2 interceptors are based on intercepting filters design pattern. The invocation of interceptors in interceptor stack closely resembles Chain of Responsibility design pattern.

**What are different ways to create Action classes in Struts2?**

Struts2 provide different ways to create action classes.

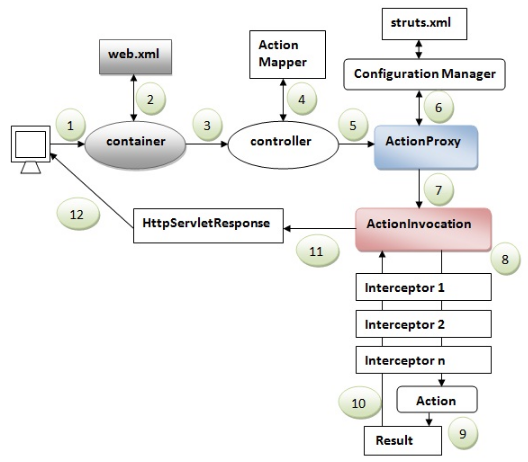
By implementing Action interface

Using Struts2 @Action annotation

By extending ActionSupport class

Any normal java class with execute() method returning String can be configured as Action class.

**Struts 2 Flow**



User sends a request for the action

Container maps the request in the web.xml file and gets the class name of controller.

Container invokes the controller (StrutsPrepareAndExecuteFilter or FilterDispatcher). Since struts2.1, it is StrutsPrepareAndExecuteFilter. Before 2.1 it was FilterDispatcher.

Controller gets the information for the action from the ActionMapper

Controller invokes the ActionProxy

ActionProxy gets the information of action and interceptor stack from the configuration manager which gets the information from the struts.xml file.

ActionProxy forwards the request to the ActionInvocation

ActionInvocation invokes each interceptors and action

A result is generated

The result is sent back to the ActionInvocation

A HttpServletResponse is generated

Response is sent to the user

**What is interceptor in Struts2?**

Interceptors are the backbone of Struts2 Framework. Struts2 interceptors are responsible for most of the processing done by the framework, such as passing request params to action classes, making Servlet API request, response, session available to Action classes, validation, i18n support, etc.

ActionInvocation is responsible to incapsulate Action classes and interceptors and to fire them in order. The most important method for use in ActionInvocation is invoke() method that keeps track of the interceptor chain and invokes the next interceptor or action. This is one of the best example of Chain of Responsibility pattern in Java EE frameworks.

**Does Struts2 action and interceptors are thread safe?**

Struts2 Action classes are thread safe because an object is instantiated for every request to handle it.

Struts2 interceptors are singleton classes and a new thread is created to handle the request, so it’s not thread safe and we need to implement them carefully to avoid any issues with shared data.

**Which class is the Front Controller in Struts2?**

org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter is the Front Controller class in Struts2 and every request processing starts from this class. Earlier versions of Struts2 uses org.apache.struts2.dispatcher.FilterDispatcher as Front Controller class.

**What are the benefits of Interceptors in Struts2?**

**Some of the benefits of interceptors are:**

Interceptor plays a crucial role in achieving high level of separation of concerns.

Struts2 interceptors are configurable, we can configure it for any action we want.

We can create our own custom interceptors to perform some common tasks such as request params logging, authentication etc. This helps us in taking care of common tasks at a single location, achieving low maintenance cost.

We can create interceptors stack to use with different actions.

**What is ValueStack and OGNL(Object-Graph Navigation Language)?**

ValueStack is the storage area where the application data is stored by Struts2 for processing the client requests. The data is stored in ActionContext objects that use ThreadLocal to have values specific to the particular request thread.

**Object-Graph Navigation Language** (OGNL) is a powerful Expression Language that is used to manipulate data stored on the ValueStack. As you can see in architecture diagram, both interceptors and result pages can access data stored on ValueStack using OGNL.

**Name some useful annotations introduced in Struts2?**

Some of the important annotations introduced in Struts2 are:

@Action to create action class

@Actions to configure single class for multiple actions

@Namespace and @Namespaces for creating different modules

@Result for result pages

@ResultPath for configuring result pages location

**What is the use of namespace in action mapping in Struts2?**

Struts2 namespace configuration allows us to create modules easily. We can use namespace to separate our action classes based on their functionality, for example admin, user, customer etc.

**Which interceptor is responsible for mapping request parameters to action class Java Bean properties?**

com.opensymphony.xwork2.interceptor.ParametersInterceptor interceptor is responsible for mapping request parameters to the Action class java bean properties. This interceptor is configured in struts-default package with name “params”. This interceptor is part of basicStack and defaultStack interceptors stack.

**Which interceptor is responsible for i18n support?**

com.opensymphony.xwork2.interceptor.I18nInterceptor interceptor is responsible for i18n support in Struts2 applications. This interceptor is configured in struts-default package with name “i18n” and it’s part of i18nStack and defaultStack.

**What is the difference in using Action interface and ActionSupport class for our action classes, which one you would prefer?**

We can implement Action interface to create our action classes. This interface has a single method execute() that we need to implement. The only benefit of using this interface is that it contains some constants that we can use for result pages, these constants are SUCCESS, ERROR, NONE, INPUT and LOGIN.

ActionSupport class is the default implementation of Action interface and it also implements interfaces related to Validation and i18n support. ActionSupport class implements Action, Validateable, ValidationAware, TextProvider and LocaleProvider interfaces. We can override validate() method of ActionSupport class to include field level validation login in our action classes.

Depending on the requirements, we can use any of the approaches to create struts 2 action classes, my favorite is ActionSupport class because it helps in writing validation and i18n logic easily in action classes.

**How can we get Servlet API Request, Response, HttpSession etc Objects in action classes?**

Struts2 action classes doesn’t provide direct access to Servlet API components such as Request, Response and Session. However sometimes we need these access in action classes such as checking HTTP method or setting cookies in response.

Thats why Struts2 API provides a bunch of \*Aware interfaces that we can implement to access these objects. Struts2 API uses dependency injection to inject Servlet API components in action classes. Some of the important Aware interfaces are SessionAware, ApplicationAware, ServletRequestAware and ServletResponseAware.

**What is the use of execAndWait interceptor?**

Struts2 provides execAndWait interceptor for long running action classes. We can use this interceptor to return an intermediate response page to the client and once the processing is finished, final response is returned to the client. This interceptor is defined in the struts-default package and implementation is present in ExecuteAndWaitInterceptor class.

**What is the use of token interceptor in Struts2?**

One of the major problems with web applications is the double form submission. If not taken care, double form submission could result in charging double amount to customer or updating database values twice. We can use token interceptor to solve the double form submission problem. This interceptor is defined in struts-default package but it’s not part of any interceptor stack, so we need to include it manually in our action classes.

**What are different Struts2 tags? How can we use them?**

Struts2 provides a lot of custom tags that we can use in result pages to create views for client request. These tags are broadly divided into three categories- Data tags, Control tags and UI tags.

We can use these tags by adding these in JSP pages using taglib directive.

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

**What is Custom Type Converter in Struts2?**

Struts2 support OGNL expression language and it performs two important tasks in Struts 2 – data transfer and type conversion.

OGNL is flexible and we can easily extend it to create our own custom converter class. Creating and configuring custom type converter class is very easy, first step is to fix the input format for the custom class. Second step is to implement the converter class. Type converter classes should implement com.opensymphony.xwork2.conversion.TypeConverter interface. Since in web application, we always get the request in form of String and send response in the form of String, Struts 2 API provides a default implementation of TypeConverter interface, StrutsTypeConverter. StrutsTypeConverter contains two abstract methods – convertFromString to convert String to Object and convertToString to convert Object to String.

**How can we write our own interceptor and map it for action?**

We can implement com.opensymphony.xwork2.interceptor.Interceptor interface to create our own interceptor. Once the interceptor class is ready, we need to define that in struts.xml package where we want to use it. We can also create interceptor stack with our custom interceptor and defaultStack interceptors. After that we can configure it for action classes where we want to use our interceptor.

**What is life cycle of an interceptor?**

Interceptor interface defines three methods – init(), destroy() and intercept(). init and destroy are the life cycle methods of an interceptor. Interceptors are Singleton classes and Struts2 initialize a new thread to handle each request. init() method is called when interceptor instance is created and we can initialize any resources in this method. destroy() method is called when application is shutting down and we can release any resources in this method.

intercept() is the method called every time client request comes through the interceptor.

**What is an interceptor stack?**

An interceptor stack helps us to group together multiple interceptors in a package for further use. struts-default package creates some of the mostly used interceptor stack – basicStack and defaultStack. We can create our own interceptor stack at the start of the package and then configure our action classes to use it.

**What is difference between Interceptors and Filters?**

The difference between Interceptors and filter are below:

- Filters are based on Servlet Specification whereas Interceptors are based on Struts2.

- Filters are executed only when patter matches whereas Interceptors executes for all request qualifies for a front controller.

- Filters are not Configurable method calls whereas Interceptors methods can be configured.

**What is struts-default package and what are it’s benefits?**

struts-default is an abstract package that defines all the Struts2 interceptors and commonly used interceptor stack. It is advisable to extend this package while configuring our application package to avoid configuring interceptors again. This is provided to help developers by eliminating the trivial task of configuring interceptor and result pages in our application.

**What is the default suffix for Struts2 action URI and how can we change it?**

The default URI suffix for Struts2 action is .action, in Struts1 default suffix was .do. We can change this suffix by defining struts.action.extension constant value in our Struts2 configuration file as:

<constant name="struts.action.extension" value="action,do"></constant>

**What is the default location of result pages and how can we change it?**

By default Struts2 looks for result pages in {WEBAPP-ROOT}/{Namespace}/ directory but sometimes we want to keep result pages in another location, we can provide struts.convention.result.path constant value in Struts2 configuration file to change the result pages location.

Another way is to use @ResultPath annotation in action classes to provide the result pages location.

**How can we upload files in Struts2 application?**

File Upload is one of the common task in a web application. Thats why Struts2 provides built in support for file upload through FileUploadInterceptor. This interceptor is configured in struts-default package and provide options to set the maximum size of a file and file types that can be uploaded to the server.

**How can we handle exceptions thrown by application in Struts2?**

Struts2 provides a very robust framework for exception handling. We can specify global results in packages and then map specific exceptions to these result pages. The exception mapping can be done at the global package level as well as action level.

It’s a good idea to have exception result pages to provide some information to user when some unexpected exception occurs that is not processed by the application. Sample configuration in struts.xml file looks like below.

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

<global-results>

<result name="exception">/exception.jsp</result>

<result name="runtime\_exception">/runtime\_exception.jsp</result>

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

</global-results>

<global-exception-mappings>

<exception-mapping exception="java.lang.Exception" result="exception"></exception-mapping>

<exception-mapping exception="java.lang.Error" result="error"></exception-mapping>

<exception-mapping exception="java.lang.RuntimeException" result="runtime\_exception"></exception-mapping>

</global-exception-mappings>

<action name="myaction" class="com.journaldev.struts2.exception.MyAction">

</action>

<action name="myspecialaction" class="com.journaldev.struts2.exception.MySpecialAction">

<exception-mapping exception="java.io.IOException" result="login"></exception-mapping>

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

</action>

</package>

**What are best practices to follow while developing Struts2 application?**

Some of the best practices while developing Struts2 application are:

Always try to extend struts-default package while creating your package to avoid code redundancy in configuring interceptors.

For common tasks across the application, such as logging request params, try to use interceptors.

Always keep action classes java bean properties in a separate bean for code reuse and implement ModelDriven interface.

If you have custom interceptor that you will use in multiple actions, create interceptor stack for that and then use it.

Try to divide your application in different modules with namespace configuration based on functional areas.

Try to use Struts2 tags in result pages for code clarify, if needed create your own type converters.

Use development mode for faster development, however make sure production code doesn’t run in dev mode.

Use Struts2 i18n support for resource bundles and to support localization.

Struts2 provides a lot of places where you can have resource bundles but try to keep one global resource bundle and one for action class to avoid confusion.

struts-default package configures all the interceptors and creates different interceptor stacks. Try to use only what is needed, for example if you don’t have localization requirement, you can avoid i18n interceptor.

**Validation in Struts 2**

**Struts 2 provide two types of validations**

**Basic Validation**

**Validation Framework Validations**

**Basic Validation**

**Steps :** 1) YourActionclass must implement interface called Validateable.

2) You have to override the validate() method in the Action Class.

a). public void validate()

3). Workflow interceptor is responsible

a). To check whether Action class is implementing Validateable interface or not.

b). If the Action class is implementing Validateable interface then calls validate() method.

3). You should specify the result with name “Input”.

**Validation Framework Validations**

**a). XML based Validation**

**b). Annotation based Validation**