**1. What is MVC?**

Model-View-Controller (MVC) is a design pattern put together to help control change. MVC decouples interface from business logic and data.

* **Model :** The model contains the core of the application's functionality. The model encapsulates the state of the application. Sometimes the only functionality it contains is state. It knows nothing about the view or controller.
* **View:** The view provides the presentation of the model. It is the *look* of the application. The view can access the model getters, but it has no knowledge of the setters. In addition, it knows nothing about the controller. The view should be notified when changes to the model occur.
* **Controller:**The controller reacts to the user input. It creates and sets the model.

**2. What is a framework?**

A framework is made up of the set of classes which allow us to use a library in a best possible way for a specific requirement.

**3. What is Struts framework?**

Struts framework is an open-source framework for developing the web applications in Java EE, based on MVC-2 architecture. It uses and extends the Java Servlet API. Struts is robust architecture and can be used for the development of application of any size. Struts framework makes it much easier to design scalable, reliable Web applications with Java.

**4. What are the components of Struts?**

Struts components can be categorize into Model, View and Controller:

* **Model:** Components like business logic /business processes and data are the part of model.
* **View:** HTML, JSP are the view components.
* **Controller:** Action Servlet of Struts is part of Controller components which works as front controller to handle all the requests.

**5. What are the core classes of the Struts Framework?**

Struts is a set of cooperating classes, servlets, and JSP tags that make up a reusable MVC 2 design.

* JavaBeans components for managing application state and behavior.
* Event-driven development (via listeners as in traditional GUI development).
* Pages that represent MVC-style views; pages reference view roots via the JSF component tree.

**6. What is ActionServlet?**

ActionServlet is a simple servlet which is the backbone of all Struts applications. It is the main Controller component that handles client requests and determines which Action will process each received request. It serves as an Action factory – creating specific Action classes based on user’s request.

**7. 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

**8. What is the ActionForm?**

ActionForm is javabean which represents the form inputs containing the request parameters from the View referencing the Action bean.

**9. What are the important methods of ActionForm?**

The important methods of ActionForm are : validate() & reset().

**10. validate() and reset() methods ?**

***validate()*** : Used to validate properties after they have been populated; Called before FormBean is handed 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)

***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.

public void reset() {}

**11. What is ActionMapping?**

Action mapping contains all the deployment information for a particular Action bean. This class is to determine where the results of the Action will be sent once its processing is complete.

**12. 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>

**13. What is role of Action Class?**

An Action Class performs a role of an adapter between the contents of an incoming HTTP request and the corresponding business logic that should be executed to process this request.

**14. 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

**15. What design patterns are used in Struts?**

Struts is based on model 2 MVC (Model-View-Controller) architecture. Struts controller uses the *command design pattern* and the action classes use the *adapter design pattern*. The process() method of the RequestProcessor uses the *template method design pattern*. Struts also implement the following J2EE design patterns.

* Service to Worker
* Dispatcher View
* Composite View (Struts Tiles)
* Front Controller
* View Helper
* Synchronizer Token

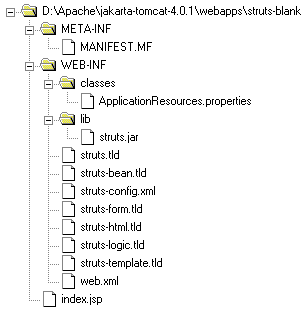
**16.Can we have more than one struts-config.xml file for a single Struts application?**

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>

**17.What is the directory structure of Struts application?**

The directory structure of Struts application :

  
  
**18.What is the difference between session scope and request scope when saving formbean ?**

when the scope is *request*,the values of formbean would be available for the current request.  
when the scope is *session*,the values of formbean would be available throughout the session.

**19. What are the important tags of struts-config.xml ?**

The five important sections are:



**20. What are the different kinds of actions in Struts?**

The different kinds of actions in Struts are:

* ForwardAction
* IncludeAction
* DispatchAction
* LookupDispatchAction
* SwitchAction

**21. What is DispatchAction?**

The DispatchAction class is used to group related actions into one class. Using this class, you can have a method for each logical action compared than a single execute method. The DispatchAction dispatches to one of the logical actions represented by the methods. It picks a method to invoke based on an incoming request parameter. The value of the incoming parameter is the name of the method that the DispatchAction will invoke.

**22.How to use DispatchAction?**

To use the DispatchAction, follow these steps :

* Create a class that extends DispatchAction (instead of Action)
* In a new class, add a method for every function you need to perform on the service – The method has the same signature as theexecute() method of an Action class.
* Do not override execute() method – Because DispatchAction class itself provides execute() method.
* Add an entry to struts-config.xml

[**DispatchAction Example  »**](http://www.developersbook.com/struts/dispatch-action-example.php)

**23.What is the use of ForwardAction?**

The ForwardAction class is useful when you’re trying to integrate Struts into an existing application that uses Servlets to perform business logic functions. You can use this class to take advantage of the Struts controller and its functionality, without having to rewrite the existing Servlets. Use ForwardAction to forward a request to another resource in your application, such as a Servlet that already does business logic processing or even another JSP page. By using this predefined action, you don’t have to write your own Action class. You just have to set up the struts-config file properly to use ForwardAction.

**24. What is IncludeAction?**

The IncludeAction class is useful when you want to integrate Struts into an application that uses Servlets. Use the IncludeAction class to include another resource in the response to the request being processed.

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

The difference is that you need to use the IncludeAction only if the action is going to be included by another action or jsp. UseForwardAction to forward a request to another resource in your application, such as a Servlet that already does business logic processing or even another JSP page.

**26.What is LookupDispatchAction?**

The LookupDispatchAction is a subclass of DispatchAction. 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.

**27.What is the use of LookupDispatchAction?**

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.

**28.What is difference between LookupDispatchAction and DispatchAction?**

The difference between LookupDispatchAction and DispatchAction is that the actual method that gets called in LookupDispatchAction is based on a lookup of a key value instead of specifying the method name directly.

**29.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.

**30.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.

**31.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.

**32.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

<form-bean name="loginForm"type="org.apache.struts.action.DynaActionForm" >  
    <form-property name="userName" type="java.lang.String"/>  
    <form-property name="password" type="java.lang.String" />  
</form-bean>

* In your Action subclass that uses your form bean:
  + import org.apache.struts.action.DynaActionForm
  + downcast the ActionForm parameter in execute() to a DynaActionForm
  + access the form fields with get(field) rather than getField()

import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import org.apache.struts.action.Action;  
import org.apache.struts.action.ActionForm;  
import org.apache.struts.action.ActionForward;  
import org.apache.struts.action.ActionMapping;  
import org.apache.struts.action.ActionMessage;  
import org.apache.struts.action.ActionMessages;  
  
  
**import org.apache.struts.action.DynaActionForm;**  
  
public class DynaActionFormExample extends Action {  
 public ActionForward execute(ActionMapping mapping, ActionForm form,  
 HttpServletRequest request, HttpServletResponse response)  
            throws Exception {          
 **DynaActionForm loginForm = (DynaActionForm) form;**  
                ActionMessages errors = new ActionMessages();          
        if (((String) loginForm.get("userName")).equals("")) {  
            errors.add("userName", new ActionMessage(  
                            "error.userName.required"));  
        }  
        if (((String) loginForm.get("password")).equals("")) {  
            errors.add("password", new ActionMessage(  
                            "error.password.required"));  
        }  
        ...........

**33.How to display validation errors on jsp page?**

<html:errors/> tag displays all the errors. <html:errors/> iterates over ActionErrors request attribute.

**34.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

**35.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>

**36.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"/>

**37. How the exceptions are handled in struts?**

Exceptions in Struts are handled in two ways:

* **Programmatic exception handling** :

Explicit try/catch blocks in any code that can throw exception. It works well when custom value (i.e., of variable) needed when error occurs.

* **Declarative exception handling** :You can either define <global-exceptions> handling tags in your struts-config.xml or define the exception handling tags within <action></action> tag. It works well when custom page needed when esrror occurs. This approach applies only to exceptions thrown by Actions.

<global-exceptions>  
 <exception key="some.key"  
 type="java.lang.NullPointerException"  
 path="/WEB-INF/errors/null.jsp"/>  
</global-exceptions>

**or**

<exception key="some.key"   
 type="package.SomeException"   
 path="/WEB-INF/somepage.jsp"/>

**38. What is difference between ActionForm and DynaActionForm?**

* An ActionForm represents an HTML form that the user interacts with over one or more pages. You will provide properties to hold the state of the form with getters and setters to access them. Whereas, using DynaActionForm there is no need of providing properties to hold the state. Instead these properties and their type are declared in the struts-config.xml
* The DynaActionForm bloats up the Struts config file with the xml based definition. This gets annoying as the Struts Config file grow larger.
* The DynaActionForm is not strongly typed as the ActionForm. This means there is no compile time checking for the form fields. Detecting them at runtime is painful and makes you go through redeployment.
* ActionForm can be cleanly organized in packages as against the flat organization in the Struts Config file.
* ActionForm were designed to act as a Firewall between HTTP and the Action classes, i.e. isolate and encapsulate the HTTP request parameters from direct use in Actions. With DynaActionForm, the property access is no different than using request.getParameter( .. ).
* DynaActionForm construction at runtime requires a lot of Java Reflection (Introspection) machinery that can be avoided.

**39.How can we make message resources definitions file available to the Struts framework environment?**

We can make message resources definitions file (properties file) available to Struts framework environment by adding this file to struts-config.xml.

<message-resources parameter="com.login.struts.ApplicationResources"/>

**40. 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

=====================================================================

**Struts2** is one of the famous framework for developing web application in java. Recently I have wrote a lot of [Struts2 Tutorials](http://www.journaldev.com/2310/struts2-tutorial-with-example-projects) and in this post, I am listing down some of the important **Struts2 interview questions**with answers to help you in interview.

1. [What is Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-framework)
2. [What are the differences between Struts1 and Struts2 or how Struts2 is better than Struts1?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-vs-struts2)
3. [What are Struts2 core components?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-components)
4. [What is interceptor in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-interceptors)
5. [Which design pattern is implemented by Struts2 interceptors?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#interceptor-design-pattern)
6. [What are different ways to create Action classes in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#action-classes)
7. [Does Struts2 action and interceptors are thread safe?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-thread-safety)
8. [Which class is the Front Controller in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-front-controller)
9. [What are the benefits of Interceptors in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#interceptor-benefits)
10. [What is ValueStack and OGNL?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#valuestack-ognl)
11. [Name some useful annotations introduced in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-annotations)
12. [Provide some important Struts2 constants that you have used?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-constants)
13. [What is the use of namespace in action mapping in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-namespace)
14. [Which interceptor is responsible for mapping request parameters to action class Java Bean properties?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-param-interceptor)
15. [Which interceptor is responsible for i18n support?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-i18n)
16. [What is the difference in using Action interface and ActionSupport class for our action classes, which one you would prefer?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#action-vs-actionsupport)
17. [How can we get Servlet API Request, Response, HttpSession etc Objects in action classes?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-aware-interfaces)
18. [What is the use of execAndWait interceptor?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#execAndWait-interceptor)
19. [What is the use of token interceptor in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#token-interceptor)
20. [How can we integrate log4j in Struts2 application?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#log4j-struts2-integration)
21. [What are different Struts2 tags? How can we use them?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-tags)
22. [What is Custom Type Converter in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#custom-type-converter)
23. [How can we write our own interceptor and map it for action?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#custom-interceptor)
24. [What is life cycle of an interceptor?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#interceptor-life-cycle)
25. [What is an interceptor stack?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#interceptor-stack)
26. [What is struts-default package and what are it’s benefits?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts-default-package)
27. [What is the default suffix for Struts2 action URI and how can we change it?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-url-suffix)
28. [What is the default location of result pages and how can we change it?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-result-path)
29. [How can we upload files in Struts2 application?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#file-upload-interceptor)
30. [What are best practices to follow while developing Struts2 application?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-best-practices)
31. [How can we handle exceptions thrown by application in Struts2?](http://www.journaldev.com/2354/struts2-interview-questions-and-answers#struts2-exception-handling)

===========================================================

### 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.

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

Struts2 is designed to overcome the shortcomings of Struts1 and to make it more flexible, extendable. Some of the noticeable differences are:

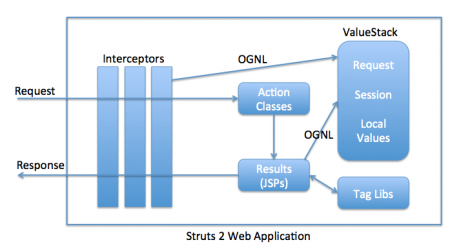
|  |  |  |
| --- | --- | --- |
| **Components** | **Struts1** | **Struts2** |
| Action Classes | Struts1 action classes are forced to extend an Abstract Class that makes it not extendable. | Struts2 action classes flexible and we can create them by implementing Action interface, extending ActionSupport class or just by having execute() method. |
| Thread Safety | Struts1 Action Classes are Singleton and not thread safe, that makes extra care on developer side to avoid any side effects because of multithreading. | Struts2 action classes gets instantiated per request, so there is no multithreading and makes them thread safe. |
| Servlet API coupling | Struts1 APIs are tightly coupled with Servlet API and Request and Response objects are passed to action classes execute() method. | Struts2 API is loosely coupled with Servlet API and automatically maps the form bean data to action class java bean properties that we mostly use. If however we need reference to Servlet API classes, there are \*Aware interfaces for that. |
| Testing | Struts1 action classes are hard to test because of Servlet API coupling. | Struts2 Action classes are like normal java classes and we can test them easily by instantiating them and setting their properties. |
| Request Parameters mapping | Struts1 requires us to create ActionForm classes to hold request params and we need to configure it in the struts configuration file. | Struts2 request params mapping is done on the fly and all we need is to have java bean properties in action classes or implement ModelDriven interface to provide the java bean class name to be used for mapping. |
| Tag Support | Struts1 uses JSTL Tags and hence are limited. | Struts2 uses OGNL and provide different kinds of UI, Control and Data Tags. It’s more versatile and easy to use. |
| Validation | Struts1 supports validation through manual validate() method | Struts2 support both manual validation as well as Validation framework integration. |
| Views Rendering | Struts1 uses standard JSP technology for providing bean values to JSP pages for views. | Struts2 uses ValueStack to store request params and attributes and we can use OGNL and Struts2 tags to access them. |
| Modules support | Struts1 modules are complex to design and looks like separate projects | Struts2 provides “namespace” configuration for packages to easily create modules. |

### 

### What are Struts2 core components?

Struts2 core components are:

* 1. Action Classes
  2. Interceptors
  3. Result Pages, JSP of FreeMarker templates
  4. ValueStack, OGNL and Tag Lisbraries

[](http://1988780851.rsc.cdn77.org/wp-content/uploads/2013/09/Struts-2-Architecture-Diagram.png)

### 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.

### 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.

* 1. By implementing Action interface
  2. Using Struts2 @Action annotation
  3. By extending ActionSupport class
  4. Any normal java class with execute() method returning String can be configured as Action class.

### 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 usesorg.apache.struts2.dispatcher.FilterDispatcher as Front Controller class.

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

Some of the benefits of interceptors are:

* 1. Interceptor plays a crucial role in achieving high level of separation of concerns.
  2. Struts2 interceptors are configurable, we can configure it for any action we want.
  3. 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.
  4. We can create interceptors stack to use with different actions.

### What is ValueStack and OGNL?

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:

* 1. @Action to create action class
  2. @Actions to configure single class for multiple actions
  3. @Namespace and @Namespaces for creating different modules
  4. @Result for result pages
  5. @ResultPath for configuring result pages location

### Provide some important Struts2 constants that you have used?

Some of the Struts2 constants that I have used are:

* 1. **struts.devMode** to run our application in development mode. This mode does reload properties files and provides extra logging and debugging feature. It’s very useful while developing our application but we should turn it off while moving our code to production.
  2. **struts.convention.result.path** to configure the location of result pages. By default Struts2 look for result pages at {WEBAPP-ROOT}/{Namespace}/ and we can change the location with this constant.
  3. **struts.custom.i18n.resources** to define global resource bundle for i18n support.
  4. **struts.action.extension** to configure the URL suffix to for Struts2 application. Default suffix is .action but sometimes we might want to change it to .do or something else.

We can configure above constants in struts.xml file like below.

|  |  |
| --- | --- |
| 1  2  3  4 | <constant name="struts.devMode" value="true"></constant>  <constant name="struts.action.extension" value="action,do"></constant>  <constant name="struts.custom.i18n.resources" value="global"></constant>  <constant name="struts.convention.result.path" value="/"></constant> |

### 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.

You can read more about them in How to get [Servlet API Session in Struts2 Action Classes](http://www.journaldev.com/2203/how-to-get-servlet-session-request-response-context-attributes-in-struts-2-action) tutorial.

### 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.

Check out [Struts2 execAndWait interceptor example](http://www.journaldev.com/2296/struts2-execandwait-interceptor-example-for-long-running-actions) to learn more about this interceptor and how to use it.

### 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.

Read more at [Struts2 token interceptor](http://www.journaldev.com/2281/struts2-token-interceptor-to-handle-double-form-submission-problem) example.

### How can we integrate log4j in Struts2 application?

Struts2 provides easy integration of log4j API for logging purpose, all we need to have is log4j configuration file in the WEB-INF/classes directory.

You can check out the sample project at [Struts2 Log4j integration](http://www.journaldev.com/2247/struts2-and-log4j-integration-example-project).

### 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.

|  |  |
| --- | --- |
| 1 | <%@ taglib uri="/struts-tags" prefix="s" %> |

Some of the important Data tags are property, set, push, bean, action, include, i18n and text tag. Read more at [Struts2 Data Tags](http://www.journaldev.com/2230/struts-2-data-tags-example-tutorial).

Control tags are used for manipulation and navigation of data from a collection. Some of the important Control tags are if-elseif-else, iterator, append, merge, sort, subset and generator tag. Read more at [Struts2 Control Tags](http://www.journaldev.com/2237/struts-2-control-tags-example-tutorial).

Struts2 UI tags are used to generate HTML markup language, binding HTML form data to action classes properties, type conversion, validation and i18n support. Some of the important UI tags are form, textfield, password, textarea, checkbox, select, radio and submit tag. Read more about them at [Struts2 UI Tags](http://www.journaldev.com/2266/struts2-ui-tags-example-tutorial).

### 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.

For implementation details, read [Struts2 OGNL Example Tutorial](http://www.journaldev.com/2221/struts-2-ognl-tutorial-with-custom-type-converter-example).

### 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.

One of the best example of using custom interceptor is to validate session, read more about it at[Struts2 Interceptor Tutorial](http://www.journaldev.com/2210/struts-2-interceptor-tutorial-with-custom-authentication-interceptor-example).

### 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 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:

|  |  |
| --- | --- |
| 1 | <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.

Read more about FileUpload interceptor at [Struts2 File Upload Example](http://www.journaldev.com/2192/struts-2-single-and-multiple-file-upload-example).

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

Some of the best practices while developing Struts2 application are:

* 1. Always try to extend struts-default package while creating your package to avoid code redundancy in configuring interceptors.
  2. For common tasks across the application, such as logging request params, try to use interceptors.
  3. Always keep action classes java bean properties in a separate bean for code reuse and implement ModelDriven interface.
  4. If you have custom interceptor that you will use in multiple actions, create interceptor stack for that and then use it.
  5. Try to divide your application in different modules with namespace configuration based on functional areas.
  6. Try to use Struts2 tags in result pages for code clarify, if needed create your own type converters.
  7. Use development mode for faster development, however make sure production code doesn’t run in dev mode.
  8. Use Struts2 i18n support for resource bundles and to support localization.
  9. 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.
  10. 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.

### 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>

s