Struts2 Interview Questions and Answers

Struts2 is one of the famous framework for developing web application in java. Recently I have wrote a lot of Struts2 Tutorials 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?

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.

2. 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:

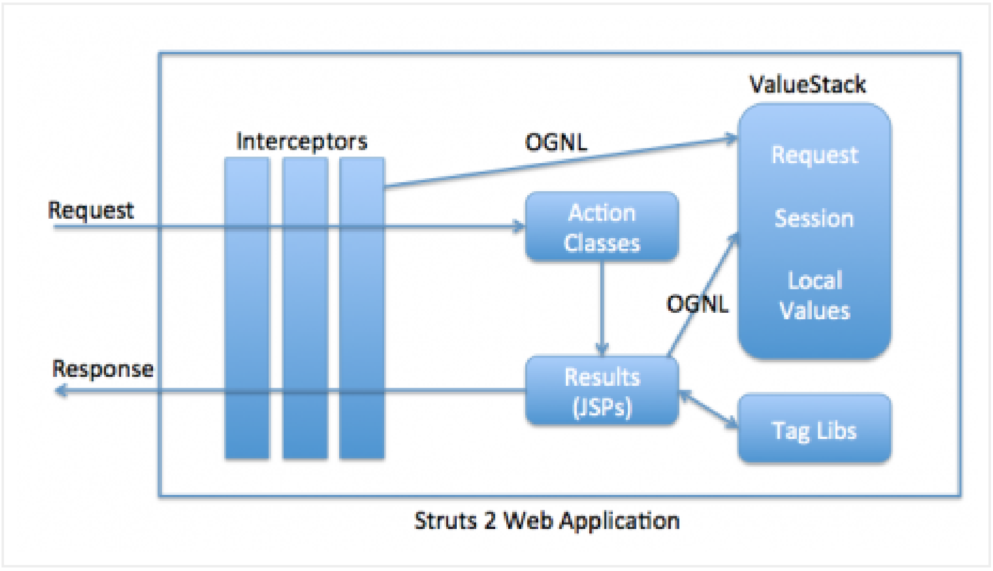
Components	Struts1	Struts2
Action Classes	Struts1 action classes are forced to extend an Abstract Class that makes it not extendable.	Struts2 action classes flexible and we can create them by implementing Action interface, extending ActionSupport class or just by having execute() method.
Thread Safety	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	Struts1 requires us to create	Struts2 request params mapping is done on the

Parameters mapping	ActionForm classes to hold request params and we need to configure it in the struts configuration file.	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.

3. What are Struts2 core components?

Struts2 core components are:

A. Action Classes
B. Interceptors
C. Result Pages, JSP of FreeMarker templates
D. ValueStack, OGNL and Tag Libraries



4. ## 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.

5. ## 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.

6. ## What are different ways to create Action classes in Struts2?

   Struts2 provide different ways to create action classes.

   A. By implementing Action interface
   B. Using Struts2 @Action annotation
   C. By extending ActionSupport class
   D. Any normal java class with execute() method returning String can be configured as Action class.

7. ## 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.

8. ## 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.

9. 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.

10. 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.

11. Name some useful annotations introduced in Struts2?

Some of the important annotations introduced in Struts2 are:

A. @Action to create action class
B. @Actions to configure single class for multiple actions
C. @Namespace and @Namespaces for creating different modules
D. @Result for result pages
E. @ResultPath for configuring result pages location

12. Provide some important Struts2 constants that you have used?

Some of the Struts2 constants that I have used are:

A. **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.
B. **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.

C. **struts.custom.i18n.resources** to define global resource bundle for i18n support.
D. **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.

```
<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>
```

13. 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.

14. 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.

15. 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.

16. 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.

17. 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 tutorial.

18. 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 to learn more about this interceptor and how to use it.

19. 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 example.

20. ## 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.

21. ## 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" %>
```

Some of the important Data tags are property, set, push, bean, action, include, i18n and text tag. Read more at Struts2 Data Tags.

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.

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.

22. ## 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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>
```

28. 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.

29. 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.

30. What are best practices to follow while developing Struts2 application?

Some of the best practices while developing Struts2 application are:

A. Always try to extend struts-default package while creating your package to avoid code redundancy in configuring interceptors.
B. For common tasks across the application, such as logging request params, try to use interceptors.
C. Always keep action classes java bean properties in a separate bean for code reuse and implement ModelDriven interface.
D. If you have custom interceptor that you will use in multiple actions, create interceptor stack for that and then use it.
E. Try to divide your application in different modules with namespace configuration based on functional areas.
F. Try to use Struts2 tags in result pages for code clarify, if needed create your own type converters.

G. Use development mode for faster development, however make sure production code doesn’t run in dev mode.
H. Use Struts2 i18n support for resource bundles and to support localization.
I. 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.
J. 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.

31. 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"></excepti
</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>
```

Read more at Struts2 Exception Handling Example.

Thats all for the Struts2 interview question and answers, if you come across any important question that I have missed, please let me know through comments.