STRUTS2 - INTERVIEW QUESTIONS

Dear readers, these **Struts2 Interview Questions** have been designed especially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **Struts2 Programming**. As per my experience, good interviewers hardly planned to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer −

What is Struts2?

Struts2 is popular and mature web application framework based on the MVC design pattern. Struts2 is not just the next version of Struts 1, but it is a complete rewrite of the Struts architecture.

Name some of the features of Struts2.

Here are some of the great features that may force you to consider Struts2 −

- **POJO forms and POJO actions** − Struts2 has done away with the Action Forms that were an integral part of the Struts framework. With Struts2, you can use any POJO to receive the form input. Similarly, you can now see any POJO as an Action class.
- **Tag support** − Struts2 has improved the form tags and the new tags allow the developers to write less code.
- **AJAX support** − Struts2 has recognised the take over by Web2.0 technologies, and has integrated AJAX support into the product by creating AJAX tags, that function very similar to the standard Struts2 tags.
- **Easy Integration** − Integration with other frameworks like Spring, Tiles and SiteMesh is now easier with a variety of integration available with Struts2.
- **Template Support** − Support for generating views using templates.
- **Plugin Support** − The core Struts2 behaviour can be enhanced and augmented by the use of plugins. A number of plugins are available for Struts2.

What are the core components of a Struct2 based application?

The Model-View-Controller pattern in Struts2 is realized with following five core components −

- Actions
- Interceptors
- Value Stack / OGNL
- Results / Result types
- View technologies

Explain the life cycle of a request in Struct2 application.

Following is the life cycle of a request in Struct2 application −

- User sends a request to the server for requesting for some resource *i. epages.*
- The FilterDispatcher looks at the request and then determines the appropriate Action.
- Configured interceptors functionalities applies such as validation, file upload etc.
- Selected action is executed to perform the requested operation.
- Again, configured interceptors are applied to do any post-processing if required.
- Finally the result is prepared by the view and returns the result to the user.

What is the purpose of struts.xml in Struct2?

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

What is the purpose of constant tag in struts.xml?

The constant tag along with name and value attributes will be used to override any of the following properties defined in default.properties, like we just set struts.devMode property. Setting struts.devMode property allows us to see more debug messages in the log file.

What is the purpose of action tag in struts.xml?

We define action tags corresponds to every URL we want to access and we define a class with execute method which will be accessed whenever we will access corresponding URL.

Results determine what gets returned to the browser after an action is executed. The string returned from the action should be the name of a result. Results are configured per-action as above, or as a "global" result, available to every action in a package. Results have optional name and type attributes. The default name value is "success".

What is the purpose of struct-config.xml in Struct2?

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

What is the purpose of form-beans tag in struct-config.xml?

This is where you map your ActionForm subclass to a name. You use this name as an alias for your ActionForm throughout the rest of the struts-config.xml file, and even on your JSP pages.

What is the purpose of global-forwards tag in struct-config.xml?

This section maps a page on your webapp to a name. You can use this name to refer to the actual page. This avoids hardcoding URLs on your web pages.

What is the purpose of action-mappings tag in struct-config.xml?

This is where you declare form handlers and they are also known as action mappings.

What is the purpose of plug-in tag in struct-config.xml?

This section tells Struts where to find your properties files, which contain prompts and error messages.

What is the purpose of struts.properties in Struct2?

This configuration file provides a mechanism to change the default behavior of the framework. Actually all of the properties contained within the struts.properties configuration file can also be configured in the web.xml using the init-param, as well using the constant tag in the struts.xml configuration file. But if you like to keep the things separate and more struts specific then you can create this file under the folder WEB-INF/classes. The values configured in this file will override the default values configured in default.properties which is contained in the struts2-core-x.y.z.jar distribution.

What are interceptors in Struts 2?

Interceptors are conceptually the same as servlet filters or the JDKs Proxy class. Interceptors allow for crosscutting functionality to be implemented separately from the action as well as the framework. You can achieve the following using interceptors −

- Providing preprocessing logic before the action is called.
- Providing postprocessing logic after the action is called.

- Catching exceptions so that alternate processing can be performed.

How can you create your custom interceptor in Struts 2?

Creating a custom interceptor is easy; the interface that needs to be extended is the Interceptor interface.

How interceptor works in Struts 2?

Actual action will be executed using the interceptor by invocation.invoke call. So you can do some pre-processing and some post-processing based on your requirement.

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

What are Result types in Struts?

The Action class manages the application's state, and the Result Type manages the view.

What is defeult result type?

Default result type is dispatcher, which is used to dispatch to JSP pages.

What is the purpose of dispatcher result type?

The dispatcher result type is the default type, and is used if no other result type is specified. It's used to forward to a servlet, JSP, HTML page, and so on, on the server. It uses the RequestDispatcher.forward method.

What is the purpose of redirect result type?

he redirect result type calls the standard response.sendRedirect method, causing the browser to create a new request to the given location. We can provide the location either in the body of the <result...> element or as a <param name="location"> element.

What is Value Stack?

The value stack is a set of several objects which keeps the following objects in the provided order −

- **Temporary Objects** − There are various temporary objects which are created during execution of a page. For example the current iteration value for a collection being looped over in a JSP tag.
- **The Model Object** − If you are using model objects in your struts application, the current model object is placed before the action on the value stack.
- **The Action Object** − This will be the current action object which is being executed.
- **Named Objects** − These objects include #application, #session, #request, #attr and #parameters and refer to the corresponding servlet scopes.

What is OGNL?

The Object-Graph Navigation Language *OGNL* is a powerful expression language that is used to reference and manipulate data on the ValueStack. OGNL also helps in data transfer and type conversion.

Which components are available using ActionContext map?

The ActionContext map consists of the following −

- **application** − application scoped variables.
- **session** − session scoped variables.

- **root / value stack** − all your action variables are stored here.
- **request** − request scoped variables.
- **parameters** − request parameters.
- **atributes** − the attributes stored in page, request, session and application scope.

Which interceptor is responsible for file upload support?

File uploading in Struts is possible through a pre-defined interceptor called FileUpload interceptor which is available through the org.apache.struts2.interceptor.FileUploadInterceptor class and included as part of the defaultStack.

What are the Struts2 configuration properties that control file uploading process?

Following are the Struts2 configuration properties that control file uploading process −

- **struts.multipart.maxSize** − The maximum size *inbytes* of a file to be accepted as a file upload. Default is 250M.
- **struts.multipart.parser** − The library used to upload the multipart form. By default is jakarta.
- **struts.multipart.saveDir** − The location to store the temporary file. By default is javax.servlet.context.tempdir.

What are the Struts2 error message keys that can come during file uploading process?

The fileUplaod interceptor uses several default error message keys −

- **struts.messages.error.uploading** − A general error that occurs when the file could not be uploaded.
- **struts.messages.error.file.too.large** − Occurs when the uploaded file is too large as specified by maximumSize.
- **struts.messages.error.content.type.not.allowed** − Occurs when the uploaded file does not match the expected content types specified.

How to override the default error message that can come during file uploading process?

You can override the text of these messages in WebContent/WEB-INF/classes/messages.properties resource files.

What is Structs 2 validation framework?

At Struts's core, we have the validation framework that assists the application to run the rules to perform validation before the action method is executed. Action class should extend the ActionSupport class, in order to get the validate method executed.

How Struts 2 validation works?

When the user presses the submit button, Struts 2 will automatically execute the validate method and if any of the if statements listed inside the method are true, Struts 2 will call its addFieldError method. If any errors have been added then Struts 2 will not proceed to call the execute method. Rather the Struts 2 framework will return input as the result of calling the action.

So when validation fails and Struts 2 returns input, the Struts 2 framework will redisplay the view file. Since we used Struts 2 form tags, Struts 2 will automatically add the error messages just above the form filed.

These error messages are the ones we specified in the addFieldError method call. The addFieldError method takes two arguments. The first is the form field name to which the error applies and the second is the error message to display above that form field.

What is XML Based Validation in struts2?

The second method of doing validation is by placing an xml file next to the action class. Struts2 XML based validation provides more options of validation like email validation, integer range validation, form validation field, expression validation, regex validation, required validation, requiredstring validation, stringlength validation and etc.

What should be the name of xml file used for validation in struts?

The xml file needs to be named '[action-class]'-validation.xml.

What types of validations are available in xml based validation in struts2?

Following is the list of various types of field level and non-field level validation available in Struts2 −

- date validator
- double validator
- email validator
- expression validator
- int validator
- regex validator
- required validator
- requiredstring validator
- stringlength validator
- url validator

What is Internationalization?

Internationalization *i18n* is the process of planning and implementing products and services so that they can easily be adapted to specific local languages and cultures, a process called localization. The internationalization process is sometimes called translation or localization enablement.

How struts2 supports Internationalization?

Struts2 provides localization ie. internationalization *i18n* support through resource bundles, interceptors and tag libraries in the following places −

- The UI Tags.
- Messages and Errors.
- Within action classes.

What is the naming convention for a resource bundle file in struts2?

The simplest naming format for a resource file is −

```
bundlename_language_country.properties
```

Here bundlename could be ActionClass, Interface, SuperClass, Model, Package, Global resource properties. Next part language_country represents the country locale for example Spanish *Spain* locale is represented by es_ES and English *UnitedStates* locale is represented by en_US etc. Here you can skip country part which is optional.

In which order Struts framework searches for a message bundle?

When you reference a message element by its key, Struts framework searches for a corresponding message bundle in the following order −

- ActionClass.properties

- Interface.properties
- SuperClass.properties
- model.properties
- package.properties
- struts.properties
- global.properties

Which class of struts is responsible to converts data types from string and vice versa?

StrutsTypeConverter class tells Struts how to convert Environment to a String and vice versa by overriding two methods convertFromString and convertToString.

What inbuilt themes are provided by Struts2?

Struts 2 comes with three built-in themes −

- **simple theme** − A minimal theme with no "bells and whistles". For example, the textfield tag renders the HTML <input/> tag without a label, validation, error reporting, or any other formatting or functionality.
- **xhtml theme** − This is the default theme used by Struts 2 and provides all the basics that the simple theme provides and adds several features like standard two-column table layout for the HTML, Labels for each of the HTML, Validation and error reporting etc.
- **css_xhtml theme** − This theme provides all the basics that the simple theme provides and adds several features like standard two-column CSS-based layout, using for the HTML Struts Tags, Labels for each of the HTML Struts Tags, placed according to the CSS stylesheet.

How to handle exceptions in Structs?

Struts makes the exception handling easy by the use of the "exception" interceptor. The "exception" interceptor is included as part of the default stack, so you don't have to do anything extra to configure it. It is available out-of-the-box ready for you to use.

What is the purpose of @Results annotation?

A @Results annotation is a collection of results. Under the @Results annotation, we can have multiple @Result annotations.

```
@Results({
   @Result(name="success", value="/success.jsp"),
   @Result(name="error", value="/error.jsp")
})
public class Employee extends ActionSupport{
 ...
}
```

What is the purpose of @Result annotation?

The @result annotations have the name that correspond to the outcome of the execute method. They also contain a location as to which view should be served corresponding to return value from execute.

```
@Result(name="success", value="/success.jsp")
public class Employee extends ActionSupport{
 ...
}
```

What is the purpose of @Action annotation?

This is used to decorate the execute method. The Action method also takes in a value which is the URL on which the action is invoked.

```
public class Employee extends ActionSupport{
   private String name;
   private int age;
   @Action(value="/empinfo")
   public String execute()
   {
       return SUCCESS;
   }
}
```

What is the purpose of @After annotation?

The @After annotation marks a action method that needs to be called after the main action method and the result was executed. Return value is ignored.

```
public class Employee extends ActionSupport{
   @After
   public void isValid() throws ValidationException {
      // validate model object, throw exception if failed
   }
   public String execute() {
      // perform secure action
      return SUCCESS;
   }
}
```

What is the purpose of @Before annotation?

The @Before annotation marks a action method that needs to be called before the main action method and the result was executed. Return value is ignored.

```
public class Employee extends ActionSupport{
   @Before
   public void isAuthorized() throws AuthenticationException {
      // authorize request, throw exception if failed
   }
   public String execute() {
      // perform secure action
      return SUCCESS;
   }
}
```

What is the purpose of @BeforeResult annotation?

The @BeforeResult annotation marks a action method that needs to be executed before the result. Return value is ignored.

```
public class Employee extends ActionSupport{
   @BeforeResult
   public void isValid() throws ValidationException {
    // validate model object, throw exception if failed
   }
   public String execute() {
      // perform action
      return SUCCESS;
   }
}
```

What is the purpose of @ConversionErrorFieldValidator annotation?

This validation annotation checks if there are any conversion errors for a field and applies them if they exist.

```
public class Employee extends ActionSupport{
   @ConversionErrorFieldValidator(message = "Default message",
                        key = "i18n.key", shortCircuit = true)
```

```
    public String getName() {
        return name;
    }
}
```

What is the purpose of @DateRangeFieldValidator annotation?

This validation annotation checks that a date field has a value within a specified range.

```
public class Employee extends ActionSupport{
   @DateRangeFieldValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true,
   min = "2005/01/01", max = "2005/12/31")
   public String getDOB() {
       return dob;
   }
}
```

What is the purpose of @DoubleRangeFieldValidator annotation?

This validation annotation checks that a double field has a value within a specified range. If neither min nor max is set, nothing will be done.

```
public class Employee extends ActionSupport{
   @DoubleRangeFieldValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true,
   minInclusive = "0.123", maxInclusive = "99.987")
   public String getIncome() {
       return income;
   }
}
```

What is the purpose of @EmailValidator annotation?

This validation annotation checks that a field is a valid e-mail address if it contains a non-empty String.

```
public class Employee extends ActionSupport{
   @EmailValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true)
   public String getEmail() {
       return email;
   }
}
```

What is the purpose of @ExpressionValidator annotation?

This non-field level validator validates a supplied regular expression.

```
@ExpressionValidator(message = "Default message", key = "i18n.key",
shortCircuit = true, expression = "an OGNL expression" )
```

What is the purpose of @IntRangeFieldValidator annotation?

This validation annotation checks that a numeric field has a value within a specified range. If neither min nor max is set, nothing will be done.

```
public class Employee extends ActionSupport{
   @IntRangeFieldValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true,
   min = "0", max = "42")
   public String getAge() {
       return age;
   }
}
```

What is the purpose of @RegexFieldValidator annotation?

This annotation validates a string field using a regular expression.

```
@RegexFieldValidator( key = "regex.field", expression = "yourregexp")
```

What is the purpose of @RequiredFieldValidator annotation?

This validation annotation checks that a field is non-null. The annotation must be applied at method level.

```
public class Employee extends ActionSupport{
   @RequiredFieldValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true)
   public String getAge() {
       return age;
   }
}
```

What is the purpose of @RequiredStringValidator annotation?

This validation annotation checks that a String field is not empty $i.e. non-null with a length > 0$.

```
public class Employee extends ActionSupport{
   @RequiredStringValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true, trim = true)
   public String getName() {
       return name;
   }
}
```

What is the purpose of @StringLengthFieldValidator annotation?

This validator checks that a String field is of the right length. It assumes that the field is a String. If neither minLength nor maxLength is set, nothing will be done.

```
public class Employee extends ActionSupport{
   @StringLengthFieldValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true,
   trim = true, minLength = "5",  maxLength = "12")
   public String getName() {
       return name;
   }
}
```

What is the purpose of @UrlValidator annotation?

This validator checks that a field is a valid URL.

```
public class Employee extends ActionSupport{
   @UrlValidator(message = "Default message",
   key = "i18n.key", shortCircuit = true)
   public String getURL() {
       return url;
   }
}
```

What is the purpose of @Validations annotation?

If you want to use several annotations of the same type, these annotation must be nested within the @Validations annotation.

```
public class Employee extends ActionSupport{
  @Validations(
   requiredFields =
      {@RequiredFieldValidator(type = ValidatorType.SIMPLE,
      fieldName = "customfield",
      message = "You must enter a value for field.")},
```

```
   requiredStrings =
      {@RequiredStringValidator(type = ValidatorType.SIMPLE,
      fieldName = "stringisrequired",
      message = "You must enter a value for string.")}
   )
   public String getName() {
       return name;
   }
}
```

What is the purpose of @CustomValidator annotation?

This annotation can be used for custom validators. Use the ValidationParameter annotation to supply additional params.

```
@CustomValidator(type ="customValidatorName", fieldName = "myField")
```

What is the purpose of @Conversion Annotation annotation?

This is a marker annotation for type conversions at Type level. The Conversion annotation must be applied at Type level.

```
@Conversion()
   public class ConversionAction implements Action {
}
```

What is the purpose of @CreateIfNull Annotation annotation?

This annotation sets the CreateIfNull for type conversion. The CreateIfNull annotation must be applied at field or method level.

```
@CreateIfNull( value = true )
private List<User> users;
```

What is the purpose of @Element Annotation annotation?

This annotation sets the Element for type conversion. The Element annotation must be applied at field or method level.

```
@Element( value = com.acme.User )
private List<User> userList;
```

What is the purpose of @Key Annotation annotation?

This annotation sets the Key for type conversion. The Key annotation must be applied at field or method level.

```
@Key( value = java.lang.Long.class )
private Map<Long, User> userMap;
```

What is the purpose of @KeyProperty Annotation annotation?

This annotation sets the KeyProperty for type conversion. The KeyProperty annotation must be applied at field or method level.

```
@KeyProperty( value = "userName" )
protected List<User> users = null;
```

What is the purpose of @TypeConversion Annotation annotation?

This annotation annotation is used for class and application wide conversion rules. The TypeConversion annotation can be applied at property and method level.

```
@TypeConversion(rule = ConversionRule.COLLECTION,
converter = "java.util.String")
public void setUsers( List users ) {
```

```
    this.users = users;
}
```

What is Next?

Further, you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)