

AJAX for Java Web Applications



Session: 6

**AJAX with JavaServer
Faces (JSF) and Struts**

Objectives

- ◆ Explain the features of Struts 2
- ◆ Use AJAX with Struts 2
- ◆ Describe the features of JSF
- ◆ Use AJAX with JSF

For Aptech Centre Use Only

Introduction

- ◆ Two of the most widely used Web frameworks are Struts and JSF.
- ◆ However, both frameworks send and receive data synchronously.
- ◆ Therefore, while the server processes the user's request, the user cannot interact with the Web page. AJAX gives these frameworks the ability to send and receive data asynchronously.
- ◆ Thus, the user can interact with the Web page while the server is processing the request.
- ◆ To enable AJAX in a Web application, JavaScript code is used to send AJAX request.
- ◆ This code is written in a JSP page.
- ◆ A server-side component, such as a servlet processes the AJAX request.
- ◆ Today developers prefer using Web frameworks such as Struts and JSF to process AJAX requests.
- ◆ This is because these frameworks provide a higher-level of abstraction above the servlet API.

AJAX in Struts Application

Consider a Struts or a JSF Web application that allows downloading articles from a Web site only after the user is registered.

To become a registered user, creation of an account is required. The Web application displays the account registration page.

The registration page contains text boxes to accept username and password, and a Submit button.

When a user enters the details and clicks the Submit button, the page waits for a response from the Web server.

If the username already exists, the user will need to repeat the process of filling details in the registration page. This is because Struts and JSF follow the click-wait-refresh cycle.

If AJAX based applications are used, the users can be notified about the validation of the username and password as soon as the text box loses focus.

Features of Struts 2

- ◆ Struts 2 is a popular Web application framework and a complete rewrite of Struts 1.
- ◆ It is based on the MVC design pattern.
- ◆ The features of Struts 2 are as follows:

Support for AJAX	• Struts 2 has integrated AJAX support by creating AJAX tags into the product, that function in the same manner as the standard Struts 2 tags.
Integration simplified	• Struts 2 provide integration with other frameworks such as Tiles, Spring, and SiteMesh.
Easy Tag Modification	• In Struts 2, using Freemarker templates, tag markups can be tweaked.
Plugin support	• The behavior of core Struts 2 can be augmented and enhanced by using plugins.
POJO forms and actions	• With Struts 2, POJO can be used to receive the form input. Similarly, POJO can be used as an Action class.
Profiling	• Integrated profiling and debugging for the application is offered by Struts 2.
Promote less configuration	• Less configuration is required in Struts 2 as it uses default values for various settings.
Tag support	• The form tags are improved in Struts 2 and thus, the developers are allowed to write less code by using these tags.
Template Support	• Generating views is supported in Struts 2 using templates.
View Technologies	• Multiple view options such as JSP, Velocity, Freemarker, and XSLT are supported by Struts 2.

AJAX in Struts 2

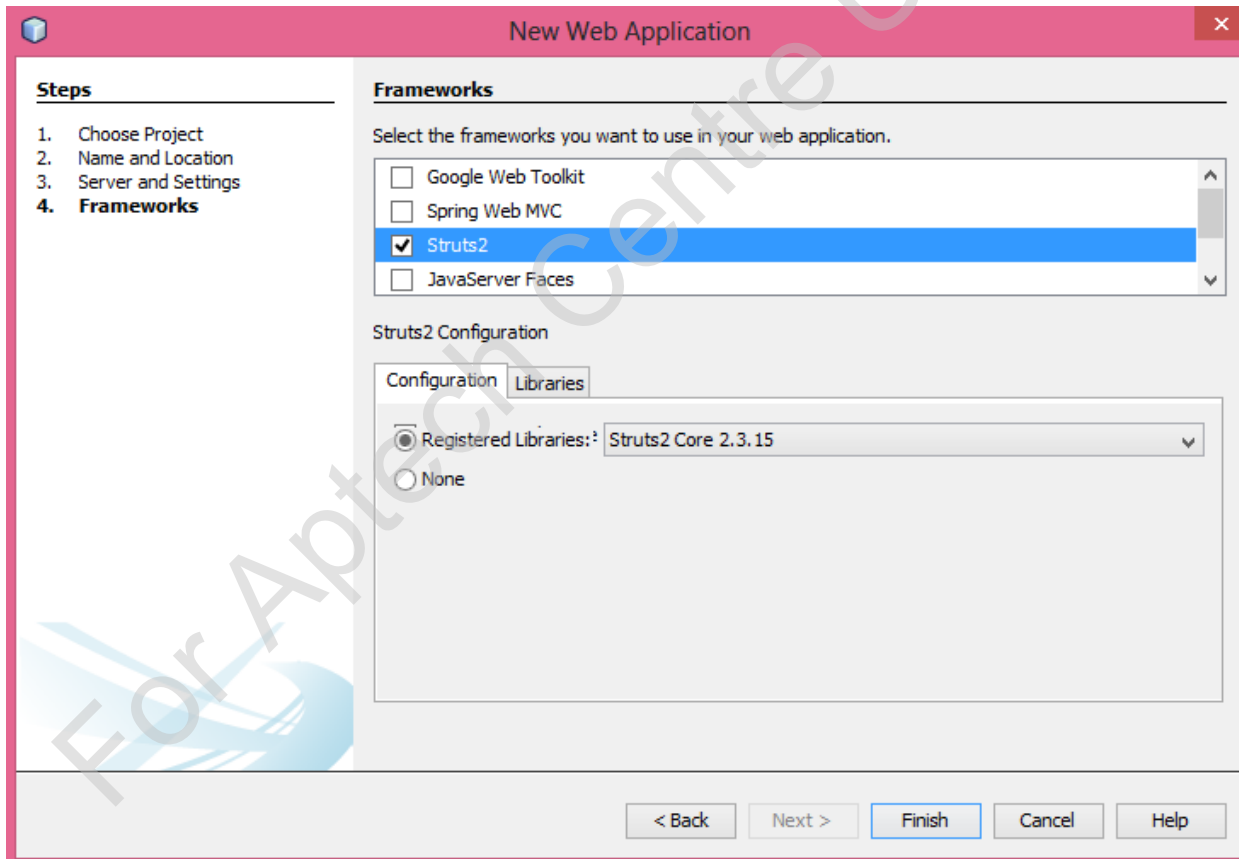
To add the Struts 2 plugin, perform the following steps:

1.

- Download the Struts 2 plugin from <http://sourceforge.net/projects/struts2nbplugin> and install it using Tools → Plugins in NetBeans IDE 8.0.

2.

- Create a Web application and use the Struts2 Framework for it as shown in the following figure:



AJAX Implementation in Struts 2 Using JQuery and JSON 1-7

Following files need to be created for the application:

- ◆ **web.xml**

- ◆ The configuration settings, filter-name, filter-class, and mapping details are specified here.

Following Code Snippet demonstrates the web.xml file:

```
...
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>
org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-
class>
    </filter>
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>Registration.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```


AJAX Implementation in Struts 2 Using JQuery and JSON 2-7

◆ **struts.xml**

Following Code Snippet depicts the struts.xml file that connects the html code and Java code by using action tags:

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
  <!-- Configuration for the default package. -->
  <package name="default" extends="struts-default,json-default">
    <action name="blurAction" class="ajax.TestAjaxCall" method="check">
      <result type="json"></result>
    </action>
  </package>
</struts>
```

- ◆ The action named 'blurAction' is called and the TestAjaxCall class is bounded with the action.
- ◆ The output received is in the form of JSON so it is required to add the Struts2-JSON jar file to the Libraries folder in the project.

AJAX Implementation in Struts 2 Using JQuery and JSON 3-7

◆ Registration.jsp

- ◆ The Registration.jsp page takes input from the user and if the username already exists, it displays a message to the user in the span tag.
- ◆ The message is retrieved from the TestAjaxCall.java class.

Following Code Snippet demonstrates the Registration.jsp page:

```
...
<script type="text/javascript" src="jquery-1.11.1.js"> </script>
<script type="text/javascript">
    $(document).ready(function() {

        $(".requiredF").blur(function() {
            $.ajax({type: "POST",
                url: 'blurAction',
                data: {
                    uname : $("#username").val(),
                    pwd : $("#password").val()
                },
                success: function(data) {
                    console.log(data.message);
                    document.getElementById("message").innerHTML=data.message;
                    $("#username").text="";
                    $("#password").text="";
                },
                error: function(error) {
                    console.log(error); }
            });
        });
    });

```

AJAX Implementation in Struts 2 Using JQuery and JSON 4-7

```
        });  
    });  
});  
</script>  
</head>  
<body>  
    <s:form action="registerAction" theme="simple" method="post"  
validate="true">  
        <table>  
            <tr>  
                <td>  
                    UserName:<s:textfield cssClass="requiredF" name="username"  
id="username" label="Username"></s:textfield>  
<span id="message" style="color:red"></span><br/>  
                </td>  
            </tr>  
            <tr>  
                <td>  
                    Password: &nbsp;<s:textfield cssClass="requiredF"  
name="password" id="password" label="Password" />  
                </td>  
            </tr>  
            <tr>  
                <td>  
                    <input type="button" id="btn" value="Submit">  
                </td>  
            </tr>  
        </table>  
    </s:form>  
</body> </html>
```

AJAX Implementation in Struts 2 Using JQuery and JSON 5-7

◆ TestAjaxCall.java

- ◆ The TestAjaxCall.java class consists of the 'check' method which is called for verifying the username.

The class is created under a package named 'ajax' as demonstrated in the following Code Snippet:

```
package ajax;
import static com.opensymphony.xwork2.Action.SUCCESS;
public class TestAjaxCall {
    private String uname;
    private String pwd;
    private String message;

    public String check() throws Exception {
        String[] unames= new String[4];
        unames[0]="jsmith";
        unames[1]="ashandrews";
        unames[2]="johnsmith";
        unames[3]="maryjoe";
        for (String name : unames) {
            if(name.equalsIgnoreCase(uname)){
                message= "Username already exists";
                break;
            }
            else
                message= "Available";
        }
    }
}
```

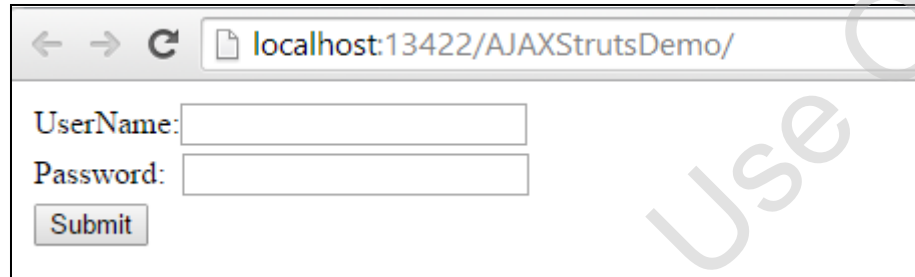
AJAX Implementation in Struts 2 Using JQuery and JSON 6-7

```
return SUCCESS;
}
public String getMessage() {
    return message;
}
public void setMessage(String message) {
    this.message = message;
}
public String getUsername() {
    return uname;
}
public void setUsername(String uname) {
    this.uname = uname;
}
public String getPwd() {
    return pwd;
}
public void setPwd(String pwd) {
    this.pwd = pwd;
}
}
```

- ◆ The 'check' method is called through AJAX and Struts 2 framework.
- ◆ It checks for the contents in the username field and returns the message accordingly.
- ◆ The message is directly accessed in the Registration.jsp page.

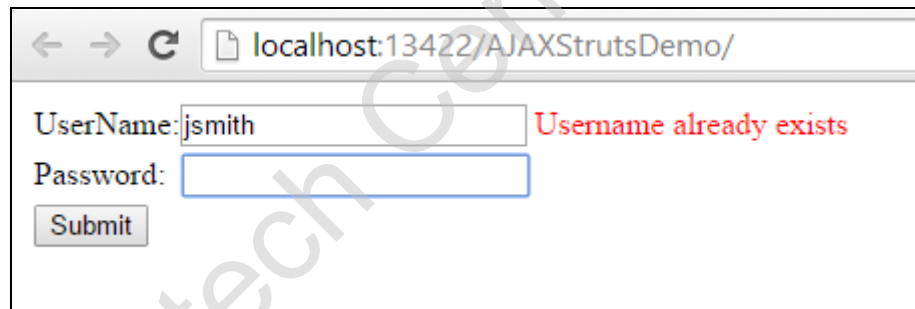
AJAX Implementation in Struts 2 Using JQuery and JSON 7-7

The output for the Registration.jsp is as shown in the following figure:



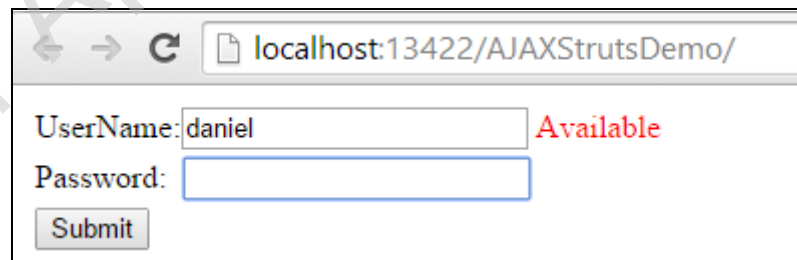
A screenshot of a web browser window showing the Registration.jsp page. The address bar displays 'localhost:13422/AJAXStrutsDemo/'. The form contains two input fields: 'UserName:' and 'Password:', both of which are empty. Below the fields is a 'Submit' button.

When the user enters a username which already exists, the output of onblur event after entering the data is as shown in the following figure:



A screenshot of the same web browser window. The 'UserName:' field now contains the text 'jsmith'. To the right of the field, a red error message 'Username already exists' is displayed. The 'Password:' field remains empty, and the 'Submit' button is still present.

The output when an unused username is specified is as shown in the following figure:



A screenshot of the same web browser window. The 'UserName:' field now contains the text 'daniel'. To the right of the field, a red message 'Available' is displayed. The 'Password:' field remains empty, and the 'Submit' button is still present.

JSF and AJAX

JSF is a component UI framework. It provides a rich set of UI components, validators, and converters to create Web applications.

However, JSF also follows the click-wait-refresh cycle. That is, JSF pages do not allow interaction with the page until the response is received from the server.

AJAX can be used in JSF applications to send and receive data asynchronously. There are several approaches to include AJAX functionality in JSF applications.

The easiest approach is to include the AJAX JavaScript code in JSF pages and use a servlet or a phase listener to process the AJAX request.

However, this requires the page author to have sound knowledge of JavaScript. JSF provides the option of creating custom components.

All the JavaScript code can be included in the custom component. The page author only needs to know how to use the custom component in the page.

New Features in JSF 2.0 and 2.2 1-2

Following are the new features in JSF 2.0:

- ◆ AJAX support within tags
- ◆ Use of annotations instead of XML.
- ◆ Composite components
- ◆ Partial State Saving
- ◆ View Parameters
- ◆ Exception Handling
- ◆ Expression Language (EL) Enhancements
- ◆ Validation
- ◆ FacesContext
- ◆ New scopes – JSF 2.0 also provides a mechanism to define custom scopes.

New Features in JSF 2.0 and 2.2 2-2

Following are the new features in JSF 2.2:

- ◆ Stateless Views for improved performance.
- ◆ HTML5 friendly markup passed straight to the browser. JSF Renderer handles decode from browser.
- ◆ JavaServer Faces has a new bean scope that is wider than request scope, narrower than session scope and different from view scope: Flow Scope.
 - ◆ Faces flows helps to modularize behavior.
 - ◆ It is built on navigation concepts.
 - ◆ Navigation is no longer just between pages but between flow 'nodes'.
 - ◆ Multiple node types include View, Method call, Switch, Flow call, and Flow return.
- ◆ Resource Library Contracts to modularize appearance.
 - ◆ It is built on facelets concepts.
 - ◆ The faces-config.xml file controls which parts of the application are allowed to use which contracts.

Custom Component

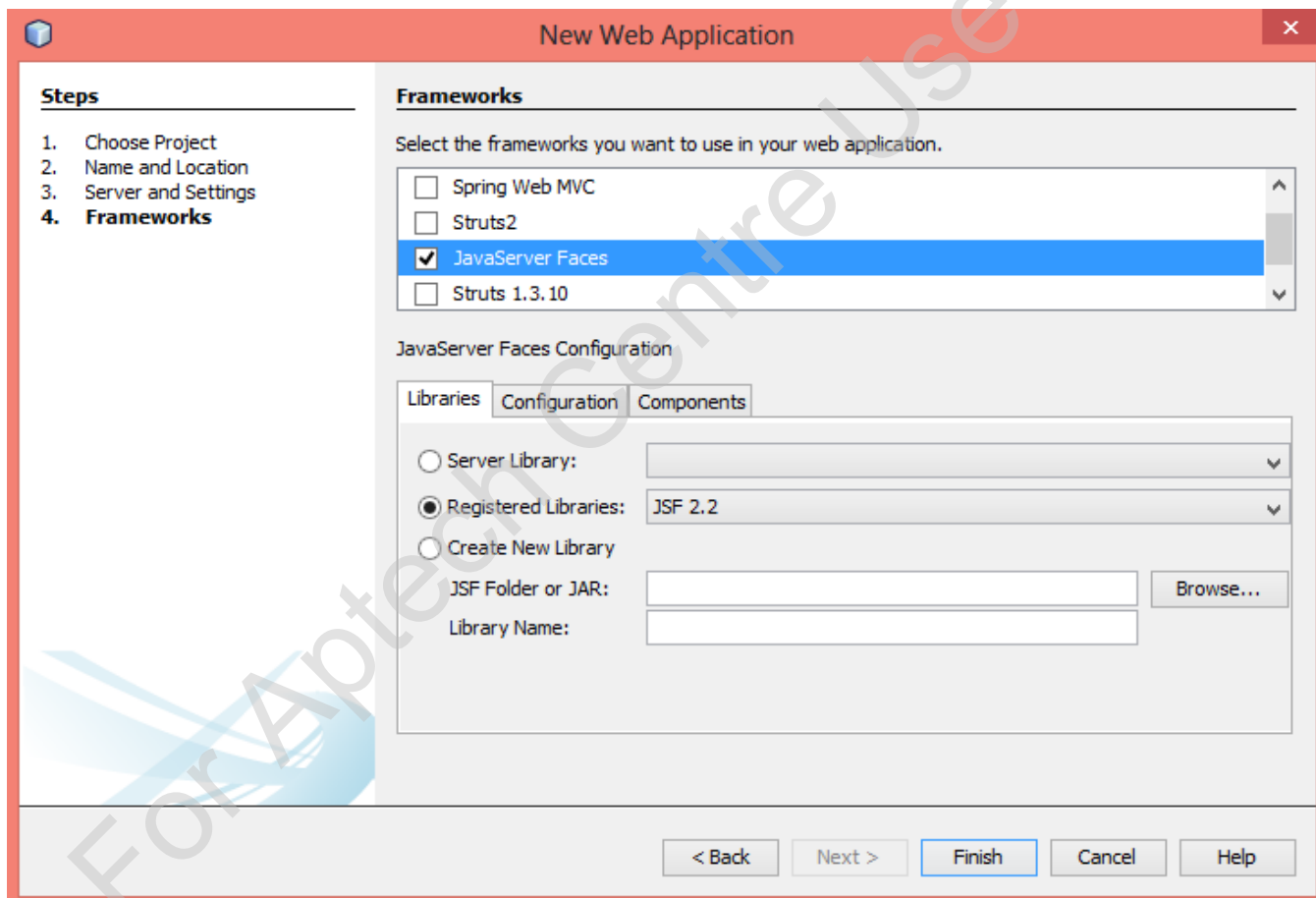
To create a JSF custom component, create or use the files shown in the following figure:



- ◆ **UI Component Class** – contains the core logic of the component.
- ◆ **Renderer Class** – contains the code to render the markup of the component.
- ◆ **UI Component Tag Class** – is the tag handler class and associates the custom tag with the component and renderer classes.
- ◆ **Tag Library Descriptor File** – describes the usage of custom component in JSP pages and associates the tag in JSP page with the UI Component tag class.
- ◆ **faces-config.xml** – specifies the name of custom component and the renderer.

Using AJAX with JSF 1-5

- ◆ Create a Web Application with the JSF (2.0 or 2.2) framework. Following figure shows the use of JavaServer Faces framework:



Using AJAX with JSF 2-5

The files used in the JSF projects are as follows:

- ◆ **index.xhtml**

- ◆ Following Code Snippet demonstrates the index.xhtml JSF page:

```
...
<h:body>
    <h1>JSF with AJAX</h1>
    <h:form>
        <h:panelGrid columns="2">
            Select a country:
            <h:selectOneMenu value="#{country.localeCode}" >
                <f:selectItems value="#{country.countryInMap}" />
                <f:ajax event="change"
listener="#{country.countryLocaleCodeChanged()}" render="countryname" />
            </h:selectOneMenu>
            Selected country is:
            <h:outputText id="countryname" value="#{country.message}" >
            </h:outputText>
        </h:panelGrid>
    </h:form>
</h:body>
</h:html>
```

- ◆ The index.xhtml page includes a drop-down which will be filled with the names of countries from a JSF Managed Bean named CountryBean.
- ◆ <h:ajax> is the tag used for implementing AJAX functionality in JSF.

Using AJAX with JSF 3-5

◆ CountryBean.java

Following Code Snippet demonstrates the JSF Managed Bean named CountryBean. Note that while creating the bean, the scope should be set to Session.

```
...
@ManagedBean(name = "country")
@SessionScoped
public class CountryBean implements Serializable {

    private static Map<String, String> countries;
    private String message;

    private String localeCode = "en"; //default value

    static {
        countries = new LinkedHashMap<String, String>();
        countries.put("United Kingdom", "en"); //label, value
        countries.put("French", "fr");
        countries.put("German", "de");
        countries.put("China", "zh_CN");
    }
}
```

Using AJAX with JSF 4-5

```
...
public void
countryLocaleCodeChanged() {

    //      localeCode =
    e.getNewValue().toString();
    if (localeCode.equals("en")) {
        message = "United
Kingdom";
    } else if (localeCode.equals("fr"))
    {
        message = "France";
    } else if (localeCode.equals("de"))
    {
        message = "Germany";
    } else {
        message = "China";
    }
}
```

```
public Map<String, String>
getCountryInMap() {
    return countries;
}

public String getLocaleCode() {
    return localeCode;
}

public void setLocaleCode(String
localeCode) {
    this.localeCode = localeCode;
}

public String getMessage() {
    return message;
}
}
```

- ◆ The method `countryLocaleCodeChanged` is accessed on the change event of the `selectOneMenu` component.
- ◆ In the method, the property 'message' is set according to the value selected in the `selectOneMenu`.

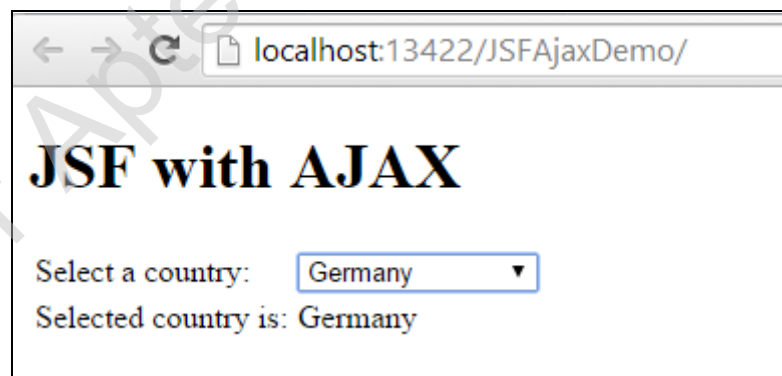
Using AJAX with JSF 5-5

Following figure shows the output on page load:



- ◆ On selecting a country from the drop-down menu, an asynchronous request is sent to the Managed Bean that returns the appropriate country name.

The name is displayed in the outputText tag as shown in the following figure:



Summary

- ◆ Web frameworks such as Struts and JSF are used to process AJAX requests as they provide higher-level of abstraction above the servlet API.
- ◆ AJAX enables improvement of the response time and provides rich look and feel to Struts and JSF applications.
- ◆ Struts 2 is a complete rewrite of the Struts architecture and not just next version of Struts 1.
- ◆ Struts 2 uses a struts.xml file for configuration of the JSP page and the Action class.
- ◆ JSF is a component UI framework with rich set of UI components, validators, and converters to create Web applications.
- ◆ Navigation in JSF 2.2 can be done without using the faces-config.xml, due to the annotations provided by it.