This chapter will take you through basic configuration which is required for a **Struts 2** application. Here we will see what can be configured with the help of few important configuration files like **web.xml, struts.xml, strutsconfig.xml** and **struts.properties**

Honestly speaking, you can start working by just using **web.xml** and **struts.xml** configuration files (as you have already witnessed in our previous chapter where our example worked using these two files). However, for your knowledge we will explain regarding other files also.

The web.xml File

The web.xml configuration file is a J2EE configuration file that determines how elements of the HTTP request are processed by the servlet container. It is not strictly a Struts2 configuration file, but it is a file that needs to be configured for Struts2 to work.

As discussed earlier, this file provides an entry point for any web application. The entry point of Struts2 application will be a filter defined in deployment descriptor (web.xml). Hence we will define an entry of *FilterDispatcher* class in web.xml. The web.xml file needs to be created under the folder **WebContent/WEB-INF**.

This is the first configuration file you will need to configure if you are starting without the aid of a template or tool that generates it (such as Eclipse or Maven2).

Following is the content of web.xml file which we used in our last example.

```
<?xml version = "1.0" Encoding = "UTF-8"?>

<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"

  xmlns = "http://java.sun.com/xml/ns/javaee"

  xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

  xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee

  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"

  id = "WebApp_ID" version = "3.0">


  <display-name>Struts 2</display-name>

  <welcome-file-list>

    <welcome-file>index.jsp</welcome-file>

  </welcome-file-list>


  <filter>

    <filter-name>struts2</filter-name>

    <filter-class>

      org.apache.struts2.dispatcher.FilterDispatcher
```

```
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

</web-app>
```

Note that we map the Struts 2 filter to **/***, and not to **/*.action** which means that all urls will be parsed by the struts filter. We will cover this when we will go through the Annotations chapter.

The Struts.xml File

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**.

Let us have a look at the struts.xml file we created in the Hello World example explained in previous chapter.

```
<?xml version = "1.0" Encoding = "UTF-8"?>

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

<struts>
  <constant name = "struts.devMode" value = "true" />
  <package name = "helloworld" extends = "struts-default">

    <action name = "hello"
      class = "com.tutorialspoint.struts2.HelloWorldAction"
      method = "execute">
      <result name = "success">/HelloWorld.jsp</result>
    </action>

    <-- more actions can be listed here -->
```

```
</package>

<-- more packages can be listed here -->


</struts>
```

The first thing to note is the **DOCTYPE**. All struts configuration file needs to have the correct doctype as shown in our little example. <struts> is the root tag element, under which we declare different packages using <package> tags. Here <package> allows separation and modularization of the configuration. This is very useful when you have a large project and project is divided into different modules.

For example, if your project has three domains - business_application, customer_application and staff_application, then you could create three packages and store associated actions in the appropriate package.

The package tag has the following attributes −

Sr.No	Attribute & Description
1	**name (required)** The unique identifier for the package
2	**extends** Which package does this package extend from? By default, we use struts-default as the base package
3	**abstract** If marked true, the package is not available for end user consumption.
4	**namespace** Unique namespace for the actions

The **constant** tag along with name and value attributes should 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.

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".

Struts.xml file can grow big over time and so breaking it by packages is one way of modularizing it, but **Struts** offers another way to modularize the struts.xml file. You could split the file into multiple xml files and import them in the following fashion.

```
<?xml version = "1.0" Encoding = "UTF-8"?>

<!DOCTYPE struts PUBLIC

  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

  "http://struts.apache.org/dtds/struts-2.0.dtd">


<struts>

  <include file="my-struts1.xml"/>

  <include file="my-struts2.xml"/>

</struts>
```

The other configuration file that we haven't covered is the struts-default.xml. This file contains the standard configuration settings for Struts and you would not have to touch these settings for 99.99% of your projects. For this reason, we are not going into too much detail on this file. If you are interested, take a look into the at the **default.properties** file available in struts2-core-2.2.3.jar file.

The Struts-config.xml File

The struts-config.xml configuration file is a link between the View and Model components in the Web Client but you would not have to touch these settings for 99.99% of your projects.

The configuration file basically contains following main elements −

Sr.No	Interceptor & Description
1	**struts-config** This is the root node of the configuration file.
2	**form-beans** This is where you map your ActionForm subclass to a name. You use this name as an alias for your Ac the strutsconfig.xml file, and even on your JSP pages.
3	**global forwards** This section maps a page on your webapp to a name. You can use this name to refer to the actual pag on your web pages.
4	**action-mappings** This is where you declare form handlers and they are also known as action mappings.

5	**controller**	This section configures Struts internals and rarely used in practical situations.
6	**plug-in**	This section tells Struts where to find your properties files, which contain prompts and error message

Following is the sample struts-config.xml file −

```
<?xml version = "1.0" Encoding = "ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
  "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">

<struts-config>

  <!-- ========== Form Bean Definitions ============ -->
  <form-beans>
    <form-bean name = "login" type = "test.struts.LoginForm" />
  </form-beans>

  <!-- ========== Global Forward Definitions ========= -->
  <global-forwards>
  </global-forwards>

  <!-- ========== Action Mapping Definitions ======== -->
  <action-mappings>
    <action
      path = "/login"
      type = "test.struts.LoginAction" >

      <forward name = "valid" path = "/jsp/MainMenu.jsp" />
      <forward name = "invalid" path = "/jsp/LoginView.jsp" />
    </action>
```

```
  </action-mappings>

  <!-- ========== Controller Definitions ======== -->
  <controller contentType = "text/html;charset = UTF-8"
    debug = "3" maxFileSize = "1.618M" locale = "true" nocache = "true"/>

</struts-config>
```

For more detail on struts-config.xml file, kindly check your struts documentation.

The Struts.properties File

This configuration file provides a mechanism to change the default behavior of the framework. Actually, all 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. There are a couple of properties that you might consider changing using the **struts.properties** file −

```
### When set to true, Struts will act much more friendly for developers
struts.devMode = true

### Enables reloading of internationalization files
struts.i18n.reload = true

### Enables reloading of XML configuration files
struts.configuration.xml.reload = true

### Sets the port that the server is run on
struts.url.http.port = 8080
```

Here any line starting with **hash** (#) will be assumed as a comment and it will be ignored by **Struts 2**.