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.

Many of the features provided in the **Struts2** framework are implemented using interceptors;

Examples include exception handling, file uploading, lifecycle callbacks, etc. In fact, as Struts2 emphasizes much of its functionality on interceptors, it is not likely to have 7 or 8 interceptors assigned per action.

Struts2 Framework Interceptors

Struts 2 framework provides a good list of out-of-the-box interceptors that come preconfigured and ready to use. Few of the important interceptors are listed below −

Sr.No	Interceptor & Description
1	**alias** Allows parameters to have different name aliases across requests.
2	**checkbox** Assists in managing check boxes by adding a parameter value of false for check boxes that are not checked.
3	**conversionError** Places error information from converting strings to parameter types into the action's field errors.
4	**createSession** Automatically creates an HTTP session if one does not already exist.
5	**debugging**

	Provides several different debugging screens to the developer.
6	**execAndWait** Sends the user to an intermediary waiting page while the action executes in the background.
7	**exception** Maps exceptions that are thrown from an action to a result, allowing automatic exception handling via redirection.
8	**fileUpload** Facilitates easy file uploading.
9	**i18n** Keeps track of the selected locale during a user's session.
10	**logger** Provides simple logging by outputting the name of the action being executed.
11	**params** Sets the request parameters on the action.
12	**prepare** This is typically used to do pre-processing work, such as setup database connections.
13	**profile** Allows simple profiling information to be logged for actions.
14	**scope** Stores and retrieves the action's state in the session or application scope.
15	**ServletConfig** Provides the action with access to various servlet-based information.
16	**timer** Provides simple profiling information in the form of how long the action takes to execute.
17	**token** Checks the action for a valid token to prevent duplicate formsubmission.
18	**validation**

Provides validation support for actions

Please look into Struts 2 documentation for complete detail on the abovementioned interceptors. But I will show you how to use an interceptor in general in your Struts application.

How to Use Interceptors?

Let us see how to use an already existing interceptor to our "Hello World" program. We will use the **timer** interceptor whose purpose is to measure how long it took to execute an action method. At the same time, I'm using **params** interceptor whose purpose is to send the request parameters to the action. You can try your example without using this interceptor and you will find that **name** property is not being set because parameter is not able to reach to the action.

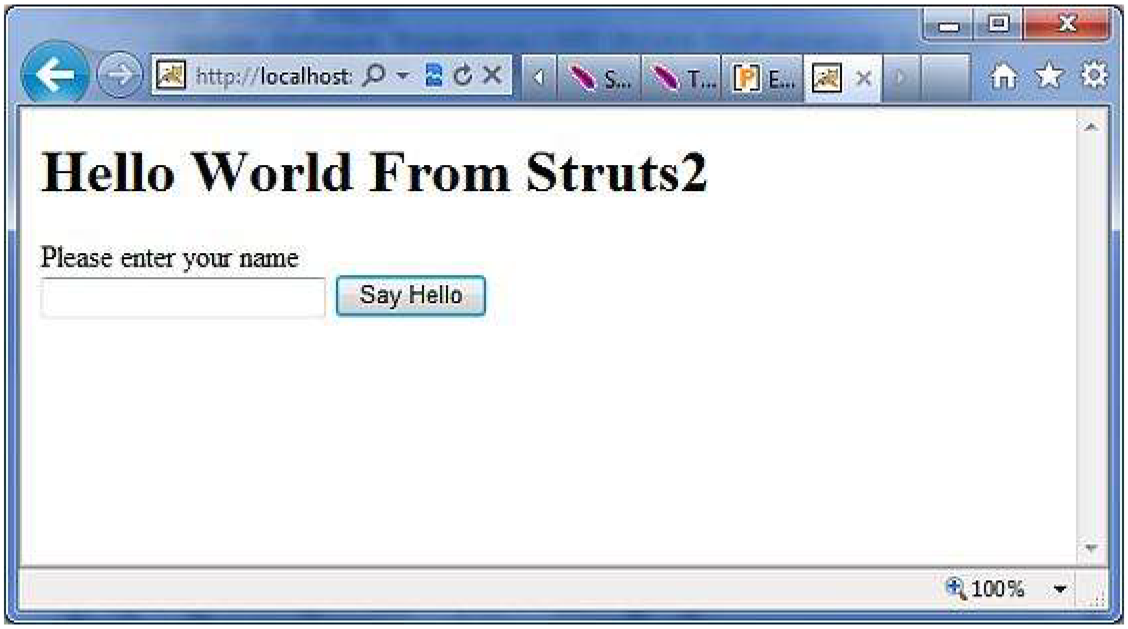
We will keep HelloWorldAction.java, web.xml, HelloWorld.jsp and index.jsp files as they have been created in **Examples** chapter but let us modify the **struts.xml** file to add an interceptor as follows −

```
<?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">
         <interceptor-ref name = "params"/>
         <interceptor-ref name = "timer" />
         <result name =
"success">/HelloWorld.jsp</result>
      </action>
   </package>
```

```
</struts>
```

Right click on the project name and click **Export > WAR File** to create a War file. Then deploy this WAR in the Tomcat's webapps directory. Finally, start Tomcat server and try to access URL **http://localhost:8080/HelloWorldStruts2/index.jsp**. This will produce the following screen −



Now enter any word in the given text box and click Say Hello button to execute the defined action. Now if you will check the log generated, you will find the following text −

```
INFO: Server startup in 3539 ms
27/08/2011 8:40:53 PM
com.opensymphony.xwork2.util.logging.commons.CommonsLogge
r info
INFO: Executed action [//hello!execute] took 109 ms.
```

Here bottom line is being generated because of **timer** interceptor which is telling that action took total 109ms to be executed.