The Value Stack

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

Sr.No	Objects & Description
1	**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.
2	**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.
3	**The Action Object** This will be the current action object which is being executed.
4	**Named Objects** These objects include #application, #session, #request, #attr and #parameters and refer to the corresponding servlet scopes.

The value stack can be accessed via the tags provided for JSP, Velocity or Freemarker. There are various tags which we will study in separate chapters, are used to get and set struts 2.0 value stack.
You can get valueStack object inside your action as follows −

```
ActionContext.getContext().getValueStack()
```

Once you have a ValueStack object, you can use the following methods to manipulate that object −

Sr.No	ValueStack Methods & Description
1	**Object findValue(String expr)** Find a value by evaluating the given expression against the stack in the default search order.
2	**CompoundRoot getRoot()** Get the CompoundRoot which holds the objects pushed onto the stack.
3	**Object peek()** Get the object on the top of the stack without changing the stack.

4	**Object pop()** Get the object on the top of the stack and remove it from the stack.
5	**void push(Object o)** Put this object onto the top of the stack.
6	**void set(String key, Object o)** Sets an object on the stack with the given key so it is retrievable by findValue(key,...)
7	**void setDefaultType(Class defaultType)** Sets the default type to convert to if no type is provided when getting a value.
8	**void setValue(String expr, Object value)** Attempts to set a property on a bean in the stack with the given expression using the default search order.
9	**int size()** Get the number of objects in the stack.

The 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.

The OGNL is very similar to the JSP Expression Language. OGNL is based on the idea of having a root or default object within the context. The properties of the default or root object can be referenced using the markup notation, which is the pound symbol.

As mentioned earlier, OGNL is based on a context and Struts builds an ActionContext map for use with OGNL. 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

It is important to understand that the Action object is always available in the value stack. So, therefore if your Action object has properties **“x”** and **“y”** there are readily available for you to use.

Objects in the ActionContext are referred using the pound symbol, however, the objects in the value stack can be directly referenced.

For example, if **employee** is a property of an action class, then it can be referenced as follows −

<s:property value = "name"/>

instead of

<s:property value = "#name"/>

If you have an attribute in session called "login" you can retrieve it as follows −

<s:property value = "#session.login"/>

OGNL also supports dealing with collections - namely Map, List and Set. For example to display a dropdown list of colors, you could do −

<s:select name = "color" list = "{'red','yellow','green'}" />

The OGNL expression is clever to interpret the "red","yellow","green" as colours and build a list based on that.

The OGNL expressions will be used extensively in the next chapters when we will study different tags. So rather than looking at them in isolation, let us look at it using some examples in the Form Tags / Control Tags / Data Tags and Ajax Tags section.

ValueStack/OGNL Example

Create Action

Let us consider the following action class where we are accessing valueStack and then setting few keys which we will access using OGNL in our view, i.e., JSP page.

```
package com.tutorialspoint.struts2;

import java.util.*;

import com.opensymphony.xwork2.util.ValueStack;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;

public class HelloWorldAction extends ActionSupport {
   private String name;

   public String execute() throws Exception {
      ValueStack stack = ActionContext.getContext().getValueStack();
      Map<String, Object> context = new HashMap<String, Object>();

      context.put("key1", new String("This is key1"));
      context.put("key2", new String("This is key2"));
      stack.push(context);

      System.out.println("Size of the valueStack: " + stack.size());
      return "success";
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }
}
```

Actually, Struts 2 adds your action to the top of the valueStack when executed. So, the usual way to put stuff on the Value Stack is to add getters/setters for the values to your Action class and then use <s:property> tag to access the values. But I'm showing you how exactly ActionContext and ValueStack work in struts.

Create Views

Let us create the below jsp file **HelloWorld.jsp** in the WebContent folder in your eclipse project. This view will be displayed in case action returns success –

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>

<html>
   <head>
      <title>Hello World</title>
   </head>

   <body>
      Entered value : <s:property value = "name"/><br/>
      Value of key 1 : <s:property value = "key1" /><br/>
      Value of key 2 : <s:property value = "key2" /> <br/>
   </body>
</html>
```

We also need to create **index.jsp** in the WebContent folder whose content is as follows –

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
   pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <title>Hello World</title>
   </head>

   <body>
      <h1>Hello World From Struts2</h1>
      <form action = "hello">
         <label for = "name">Please enter your name</label><br/>
         <input type = "text" name = "name"/>
         <input type = "submit" value = "Say Hello"/>
      </form>
   </body>
</html>
```

Configuration Files

Following is the content of **struts.xml** file −

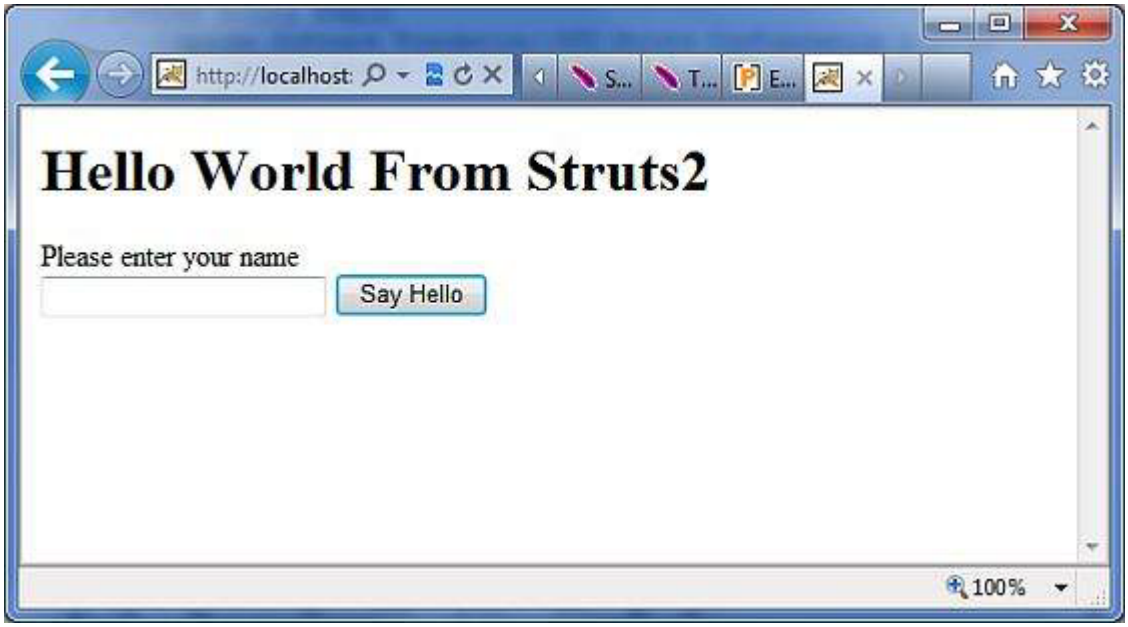
```
<?xml version = "1.0" Encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">

<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>

   </package>
</struts>
```

Following is the content of **web.xml** file −

```
<?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>
```

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 at the bottom −

Size of the valueStack: 3

This will display the following screen, which will display whatever value you will enter and value of key1 and key2 which we had put on ValueStack