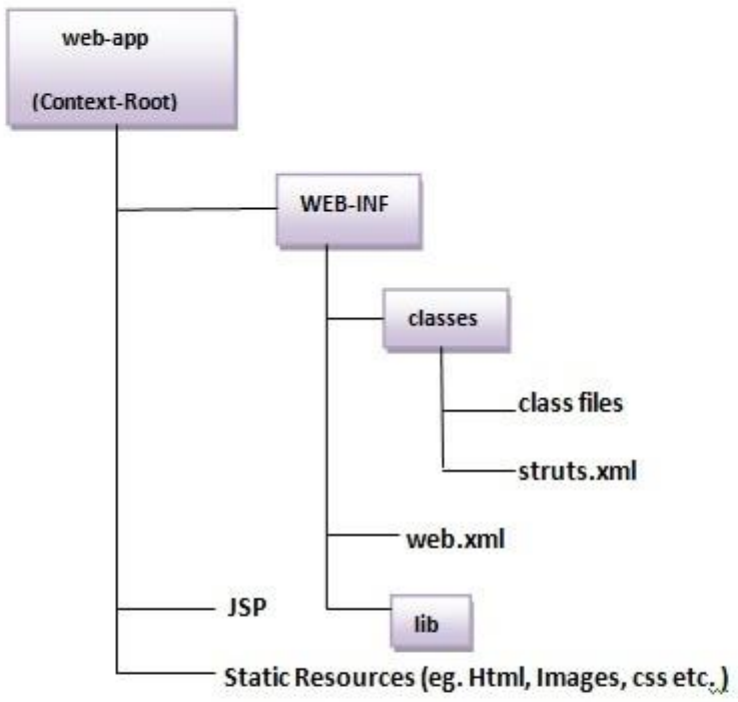
Simple struts Application creation

1. Create the directory structure
2. Create input page (index.jsp)
3. Provide the entry of Controller in (web.xml) file
4. Create the action class (Product.java)
5. Map the request with the action in (struts.xml) file and define the view components
6. Create view components (welcome.jsp)
7. load the jar files
8. start server and deploy the project

1) Create the directory structure

The directory structure of struts 2 is same as servlet/JSP. Here, struts.xml file must be located in the classes folder.



2) Create input page (index.jsp)

This jsp page creates a form using struts UI tags. To use the struts UI tags, you need to specify uri /struts-tags. Here, we have used s:form to create a form, s:textfield to create a text field, s:submit to create a submit button.

index.jsp

```
<%@ taglib uri="/struts-tags" prefix="s" %>
<s:form action="product">
<s:textfield name="id" label="Product Id"></s:textfield>
<s:textfield name="name" label="Product Name"></s:textfield>
<s:textfield name="price" label="Product Price"></s:textfield>
<s:submit value="save"></s:submit>
</s:form>
```

3) Provide the entry of Controller in (web.xml) file

In struts 2, **StrutsPrepareAndExecuteFilter** class works as the controller. As we know well, struts 2 uses filter for the controller. It is implicitly provided by the struts framework.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <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>
</web-app>
```

4) Create the action class (Product.java)

This is simple bean class. In struts 2, action is POJO (Plain Old Java Object). It has one extra method **execute** i.e. invoked by struts framework by default.

Product.java

```java
package com.javatpoint;

public class Product {
private int id;
private String name;
private float price;
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public float getPrice() {
    return price;
}
public void setPrice(float price) {
    this.price = price;
}

public String execute(){
    return "success";
}
}
```

5) Map the request in (struts.xml) file and define the view components

It is the important file from where struts framework gets information about the action and decides which result to be invoked. Here, we have used many elements such as struts, package, action and result.

struts element is the root elements of this file. It represents an application.

package element is the sub element of struts. It represents a module of the application. It generally extends the **struts-default** package where many interceptors and result types are defined.

action element is the sub element of package. It represents an action to be invoked for the incoming request. It has name, class and method attributes. If you don't specify name attribute by default execute() method will be invoked for the specified action class.

result element is the sub element of action. It represents an view (result) that will be invoked. Struts framework checks the string returned by the action class, if it returns success, result page for the action is invoked whose name is success or has no name. It has **name** and **type** attributes. Both are optional. If you don't specify the result name, by default success is assumed as the result name. If you don't specify the type attribute, by default **dispatcher** is considered as the default result type. We will learn about result types later.

struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
<package name="default" extends="struts-default">


<action name="product" class="com.javatpoint.Product">
<result name="success">welcome.jsp</result>
</action>
</package>
</struts>
```

6) Create view components (welcome.jsp)

It is the view component the displays information of the action. Here, we are using struts tags to get the information.

The s:property tag returns the value for the given name, stored in the action object.

welcome.jsp

```
<%@ taglib uri="/struts-tags" prefix="s" %>

Product Id:<s:property value="id"/><br/>
Product Name:<s:property value="name"/><br/>
Product Price:<s:property value="price"/><br/>
```

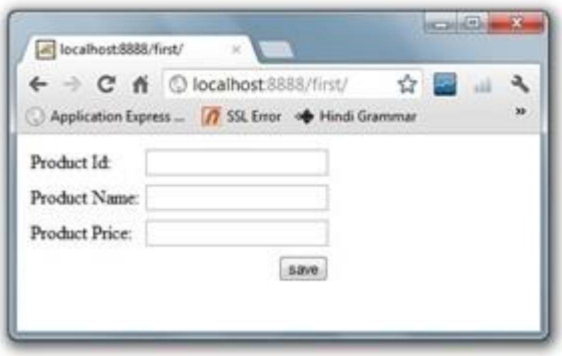
7) Load the jar files

To run this application, you need to have the struts 2 jar files. Here, we are providing all the necessary jar files for struts 2. Download it and put these jar files in the lib folder of your project.

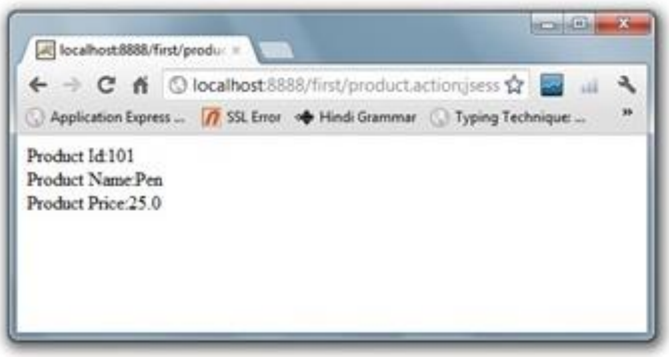
download the struts2 jar files

8) start server and deploy the project

Finally, start the server and deploy the project and access it.



Product Id:
Product Name:
Product Price:


localhost:8888/first/produ
localhost:8888/first/product.action;jsess
Application Express ...
SSL Error
Hindi Grammar
Typing Technique ...

```
Product Id:101
Product Name:Pen
Product Price:25.0
```