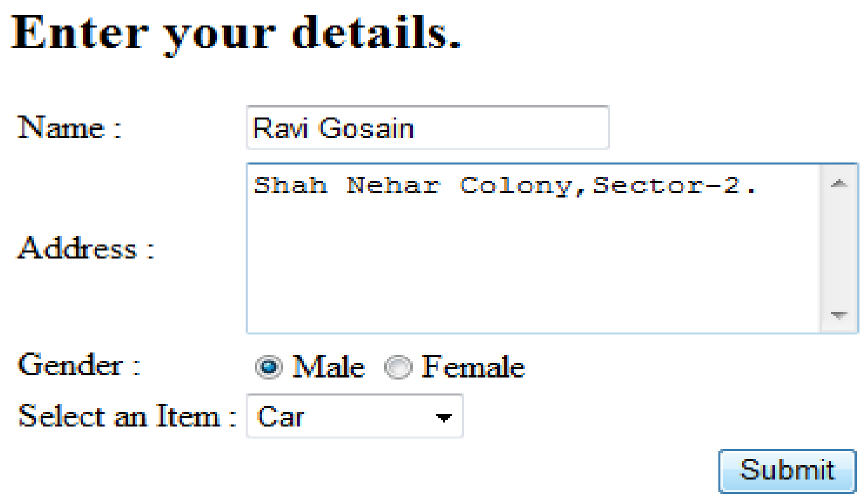
Struts2 Spring Hibernate Integration Example :

This example shows how to integrate Struts2 with Spring and Hibernate . User is shown a web page where he enters his details and clicks on the submit button. He is shown a list of all saved users in another page. See figures below.

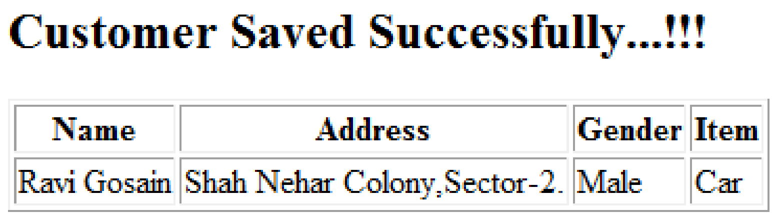
First page : User enters his details and clicks on submit button.



Enter your details.

Name :	Ravi Gosain
Address :	Shah Nehar Colony,Sector-2.
Gender :	[✓ Male] Female
Select an Item :	Car

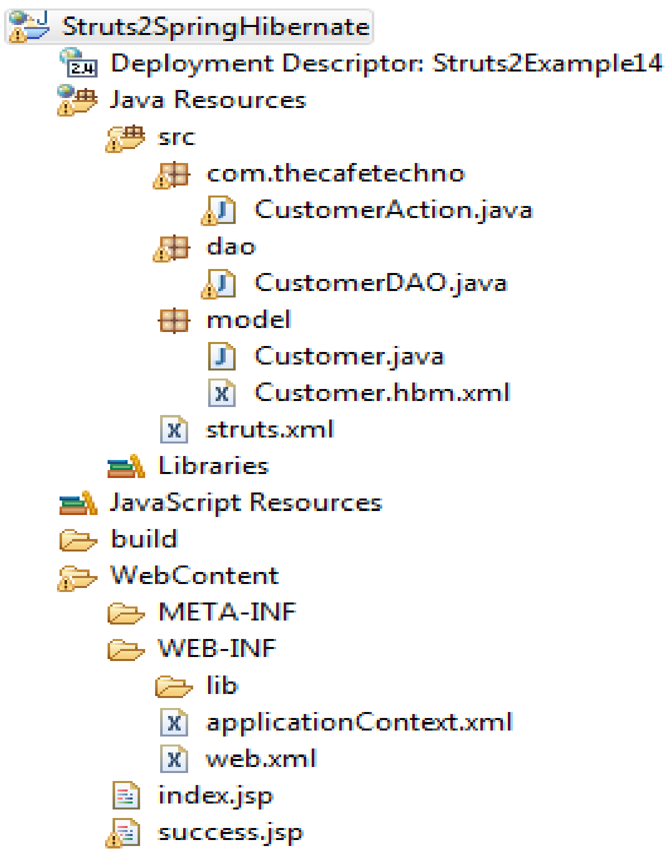
Second page : Here he is shown list of saved users.



Customer Saved Successfully...!!!

Name	Address	Gender	Item
Ravi Gosain	Shah Nehar Colony,Sector-2.	Male	Car

Project Structure :



```
Struts2SpringHibernate
    Deployment Descriptor: Struts2Example14
    Java Resources
        src
            com.thecafetechno
                CustomerAction.java
            dao
                CustomerDAO.java
            model
                Customer.java
                Customer.hbm.xml
            struts.xml
        Libraries
    JavaScript Resources
    build
    WebContent
        META-INF
        WEB-INF
            lib
            applicationContext.xml
            web.xml
        index.jsp
        success.jsp
```

web.xml : In a Struts 2 application, the most important element in deployment descriptor is the filter and filter-mapping which configure the Struts 2 FilterDispatcher. Basically this servlet filter `StrutsPrepareAndExecuteFilter` is the implementation of Struts 2 Framework. This filter processes all the incoming requests and looks for Struts 2 actions for processing the requests. URL pattern '/*' is maaped with this filter .

```
<?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_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>Struts2Example14</display-name>
       <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>
       <context-param>
```

```
            <param-name>contextConfigLocation</param-name>
            <param-value>WEB-INF/applicationContext.xml</param-value>
        </context-param>

        <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</
listener-class>
        </listener>
        <welcome-file-list>
              <welcome-file>index.jsp</welcome-file>
        </welcome-file-list>
</web-app>
```

index.jsp : This is the index page shown to user where he enters his details and submits the request. Here *'customerAction'* is the value of 'action' attribute of 's:form' tag. This value is used by Struts2 filter to find the action class to process this request.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
</head>
<body>
<h2>Enter your details.</h2>
<s:form action="customerAction" >
       <s:textfield name="custName" label="Name " value=""/>
       <s:textarea name="address" label="Address " value="" cols="30" rows="5" />
       <s:radio name="gender" label="Gender " list="{'Male','Female'}" />
     <s:select name="item" list="{'Laptop','Car','Phone'}" headerKey=""
         headerValue="Select-Item" label="Select an Item " />
<s:submit name="submit"/>
</s:form>
 </body>
</html>
```

struts.xml : All action mappings are defined in struts.xml within 'struts' root tag. Requests are mapped using name attribute of 'action' tag. Here we have defined one action *'customerActionBean' which is mapped with request generated at index.jsp because name attribute's value (*name="*customerAction*" *in struts.xml) is same as the action attribute's value (*action="*customerAction*" *in index.jsp) . Note that here we have given bean name (customerActionBean) of action class not the actual action class name (i.e com.thecafetechno.***CustomerAction***).Actual class name is given in spring configuration file applicationContext.xml.*

The 'result' tag declared here specifies the view that can be called by the enclosing action. Multiple 'result' tags can be declared within an action but which view will be called by the action is decided by the String value returned from execute() method defined in the action class.In our example String "successView" is returned so the corresponding 'success.jsp' is called by the action calss.

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
```

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

<struts>
    <package name="myPack" extends="struts-default">
        <action name="customerAction" class="customerActionBean">
            <result name="successView">/success.jsp</result>
        </action>
    </package>
</struts>
```

applicationContext.xml : It is the Spring configuration file where all the beans are defined.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <bean id="dataSourceBean" class="org.apache.commons.dbcp.BasicDataSource" >
          <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
          <property name="url" value="jdbc:mysql://localhost/devdb"/>
          <property name="username" value="dev"/>
          <property name="password" value="dev"/>
      </bean>

      <bean id="sessionFactoryBean"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
          <property name="dataSource" ref="dataSourceBean"/>
          <property name="mappingResources">
              <list>
                  <value>model/Customer.hbm.xml</value>
              </list>
          </property>
          <property name="hibernateProperties">
              <value>hibernate.dialect=org.hibernate.dialect.MySQLDialect
              hibernate.hbm2ddl.auto=update
              hibernate.show_sql=true
              </value>
           </property>
      </bean>

<bean id="hibernateTemplateBean"
class="org.springframework.orm.hibernate3.HibernateTemplate">
          <property name="sessionFactory">
              <ref bean="sessionFactoryBean"/>
          </property>
      </bean>
      <bean id="customerDAOBean" class="dao.CustomerDAO" >
             <property name="hibernateTemplate" ref="hibernateTemplateBean" />
      </bean>

      <bean id="customerActionBean" class="com.thecafetechno.CustomerAction" >
```

```
		<property name="customerDAO" ref="customerDAOBean" />
	</bean>
</beans>
```

CustomerAction.java : Struts action class extends ActionSupport class. In this action class an object **'customer' (Pojo)** is created which is automatically set by Struts2 framework since its (customer) properties name matches with the request parameters specified in index.jsp. Another property **'customerDAO' is injected by spring.**Here execute() method returns a string value which will be used by the action class to select a view to display.

[code lang=java]

```java
package com.thecafetechno;
import java.util.ArrayList;
import java.util.List;

import model.Customer;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

import dao.CustomerDAO;

public class CustomerAction extends ActionSupport implements ModelDriven<Customer> {
	private Customer customer=new Customer();
	private CustomerDAO customerDAO;
	List<Customer> savedCustomerList = new ArrayList<Customer>();

	public Customer getModel() {
		return customer;
	}

	public String execute() throws Exception {
		customerDAO.saveOrUpdate(customer);
		savedCustomerList=customerDAO.fetchAllCustomer();
		return "successView";
	}
	public List<Customer> getSavedCustomerList() {
		return savedCustomerList;
	}

	public void setSavedCustomerList(List<Customer> savedCustomerList) {
		this.savedCustomerList = savedCustomerList;
	}

	public void setCustomerDAO(CustomerDAO customerDAO) {
		this.customerDAO = customerDAO;
	}

	public Customer getCustomer() {
		return customer;
	}

	public void setCustomer(Customer customer) {
		this.customer = customer;
```

```
        }
}
```

success.jsp :

[code lang=text]

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
</head>
<body>
<h2>Customer Saved Successfully...!!! </h2>

<table border="1px" >
        <tr>
                <th>Name</th>
                <th>Address</th>
                <th>Gender</th>
                <th>Item</th>
        </tr>
        <s:iterator value="savedCustomerList">
                <tr>
                        <td><s:property value="custName" /></td>
                        <td><s:property value="address" /></td>
                        <td><s:property value="gender" /></td>
                        <td><s:property value="item" /></td>
                </tr>
        </s:iterator>
</table>
 </body>
</html>
```

CustomerDAO.java :

```
package dao;

import java.util.List;
import model.Customer;
import org.springframework.orm.hibernate3.HibernateTemplate;

public class CustomerDAO  {
        private HibernateTemplate hibernateTemplate;

        public HibernateTemplate getHibernateTemplate() {
                return hibernateTemplate;
        }

        public void setHibernateTemplate(HibernateTemplate hibernateTemplate) {
                this.hibernateTemplate = hibernateTemplate;
        }

        public void saveOrUpdate(Customer customer) {
```

```
		hibernateTemplate.save(customer);
	}

	public List<Customer> fetchAllCustomer() {
		return hibernateTemplate.loadAll(Customer.class);
	}
}
```

Customer.java :

```
package model;

public class Customer {
	private long id;
	private String custName;
	private String address;
	private String gender;
	private String item;
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getCustName() {
		return custName;
	}
	public void setCustName(String custName) {
		this.custName = custName;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	public String getGender() {
		return gender;
	}
	public void setGender(String gender) {
		this.gender = gender;
	}
	public String getItem() {
		return item;
	}
	public void setItem(String item) {
		this.item = item;
	}

}
```

Customer.hbm.xml :

```xml
<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
 <class name="model.Customer" table="CUSTOMER">
    <id name="id" column="Id">
     <generator class="native">
    </generator>
   </id>
   <property name="custName" />
    <property name="address" />
       <property name="gender" />
    <property name="item" />
 </class>
</hibernate-mapping>
```