SPRING - EXCEPTION HANDLING EXAMPLE

The following example show how to write a simple web based application using Spring MVC Framwork, which can handle one or more exceptions raised inside its controllers. To start with it, let us have working Eclipse IDE in place and follow the following steps to develope a Dynamic Form based Web Application using Spring Web Framework:

Step	Description
1	Create a *Dynamic Web Project* with a name *HelloWeb* and create a package *com.tutorialspoint* under the *src* folder in the created project.
2	Drag and drop below mentioned Spring and other libraries into the folder *WebContent/WEB-INF/lib*.
3	Create a Java classes *Student*, *StudentController* and *SpringException* under the *com.tutorialspoint* package.
4	Create Spring configuration files *Web.xml* and *HelloWeb-servlet.xml* under the *WebContent/WEB-INF* folder.
5	Create a sub-folder with a name *jsp* under the *WebContent/WEB-INF* folder. Create a view files *student.jsp*, *result.jsp*, *error.jsp*, and *ExceptionPage.jsp* under *jsp* sub-folder.
6	The final step is to create the content of all the source and configuration files and export the application as explained below.

Following is the content of **Student.java** file:

```
package com.tutorialspoint;

public class Student {
   private Integer age;
   private String name;
   private Integer id;

   public void setAge(Integer age) {
      this.age = age;
   }
   public Integer getAge() {
      return age;
   }

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

   public void setId(Integer id) {
      this.id = id;
   }
   public Integer getId() {
      return id;
   }
}
```

Following is the content of **SpringException.java** file:

```
package com.tutorialspoint;
```

```
public class SpringException extends RuntimeException{
   private String exceptionMsg;

   public SpringException(String exceptionMsg) {
      this.exceptionMsg = exceptionMsg;
   }

   public String getExceptionMsg(){
      return this.exceptionMsg;
   }

   public void setExceptionMsg(String exceptionMsg) {
      this.exceptionMsg = exceptionMsg;
   }
}
```

Following is the content of **StudentController.java** file. Here you need to annotate a service method using *@ExceptionHandler* where you can specify one or more exceptions to be handled. If you are specifying more than one exceptions then you can use comma separated values.

```
package com.tutorialspoint;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.ui.ModelMap;

@Controller
public class StudentController {

   @RequestMapping(value = "/student", method = RequestMethod.GET)
   public ModelAndView student() {
      return new ModelAndView("student", "command", new Student());
   }

   @RequestMapping(value = "/addStudent", method = RequestMethod.POST)
   @ExceptionHandler({SpringException.class})
   public String addStudent( @ModelAttribute("HelloWeb")Student student,
      ModelMap model) {

      if(student.getName().length() < 5 ){
         throw new SpringException("Given name is too short");
      }else{
       model.addAttribute("name", student.getName());
      }

      if( student.getAge() < 10 ){
         throw new SpringException("Given age is too low");
      }else{
       model.addAttribute("age", student.getAge());
      }
      model.addAttribute("id", student.getId());
      return "result";
   }
}
```

Following is the content of Spring Web configuration file **web.xml**

```
<web-app
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
```

```
    <display-name>Spring Exception Handling</display-name>

    <servlet>
        <servlet-name>HelloWeb</servlet-name>
        <servlet-class>
           org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloWeb</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

</web-app>
```

Following is the content of another Spring Web configuration file **HelloWeb-servlet.xml**

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
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-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:component-scan base-package="com.tutorialspoint" />

<bean >
   <property name="prefix" value="/WEB-INF/jsp/" />
   <property name="suffix" value=".jsp" />
</bean>

<bean class="org.springframework.web.servlet.handler.
      SimpleMappingExceptionResolver">
   <property name="exceptionMappings">
      <props>
         <prop key="com.tutorialspoint.SpringException">
            ExceptionPage
         </prop>
      </props>
   </property>
   <property name="defaultErrorView" value="error"/>
</bean>

</beans>
```

Here you specified *ExceptionPage* as an exception view in case SpringException occurs, if there is any other type of exception then a generic view *error* will take place.

Following is the content of Spring view file **student.jsp**:

```
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<html>
<head>
    <title>Spring MVC Exception Handling</title>
</head>
<body>

<h2>Student Information</h2>
<form:form method="POST" action="/HelloWeb/addStudent">
   <table>
   <tr>
   <td><form:label path="name">Name</form:label></td>
   <td><form:input path="name" /></td>
   </tr>
```

```
   <tr>
   <td><form:label path="age">Age</form:label></td>
   <td><form:input path="age" /></td>
   </tr>
   <tr>
   <td><form:label path="id">id</form:label></td>
   <td><form:input path="id" /></td>
   </tr>
   <tr>
   <td colspan="2">
   <input type="submit" value="Submit"/>
   </td>
   </tr>
   </table>
</form:form>
</body>
</html>
```

Following is the content of Spring view file **error.jsp**:

```
<html>
<head>
    <title>Spring Error Page</title>
</head>
<body>

<p>An error occured, please contact webmaster.</p>

</body>
</html>;
```

Following is the content of Spring view file **ExceptionPage.jsp**. Here you will access the exception instance via ${exception}.

```
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<html>
<head>
    <title>Spring MVC Exception Handling</title>
</head>
<body>

<h2>Spring MVC Exception Handling</h2>

<h3>${exception.exceptionMsg}</h3>

</body>
</html>
```

Following is the content of Spring view file **result.jsp**:

```
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<html>
<head>
    <title>Spring MVC Form Handling</title>
</head>
<body>

<h2>Submitted Student Information</h2>
   <table>
   <tr>
   <td>Name</td>
   <td>${name}</td>
   </tr>
   <tr>
   <td>Age</td>
   <td>${age}</td>
   </tr>
```

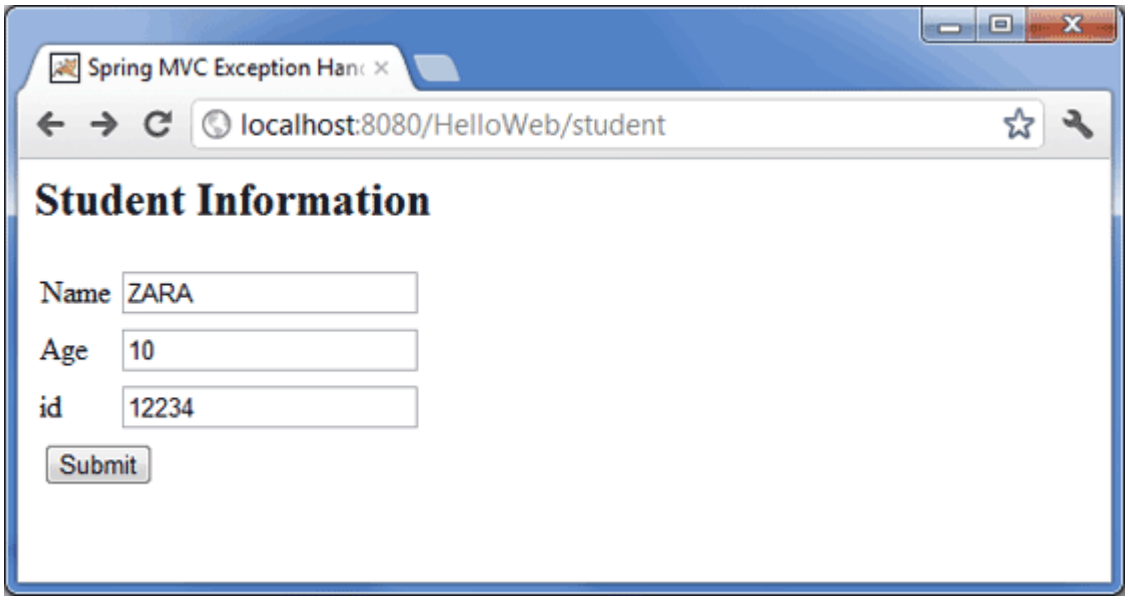
```
    <tr>
    <td>ID</td>
    <td>${id}</td>
    </tr>
    </table>
</body>
</html>
```

Finally, following is the list of Spring and other libraries to be included in your web application. You simply drag these files and drop them in **WebContent/WEB-INF/lib** folder.

- commons-logging-x.y.z.jar
- org.springframework.asm-x.y.z.jar
- org.springframework.beans-x.y.z.jar
- org.springframework.context-x.y.z.jar
- org.springframework.core-x.y.z.jar
- org.springframework.expression-x.y.z.jar
- org.springframework.web.servlet-x.y.z.jar
- org.springframework.web-x.y.z.jar
- spring-web.jar

Once you are done with creating source and configuration files, export your application. Right click on your application and use **Export > WAR File** option and save your **HelloWeb.war** file in Tomcat's *webapps* folder.

Now start your Tomcat server and make sure you are able to access other web pages from webapps folder using a standard browser. Now try to access the URL **http://localhost:8080/HelloWeb/student**. If everything is fine with your Spring Web Applicationand, you should see the following result :



Enter the values as shown above and click submit buttom. If everything is fine with your Spring Web Application, you should see the following result:

Given name is too short