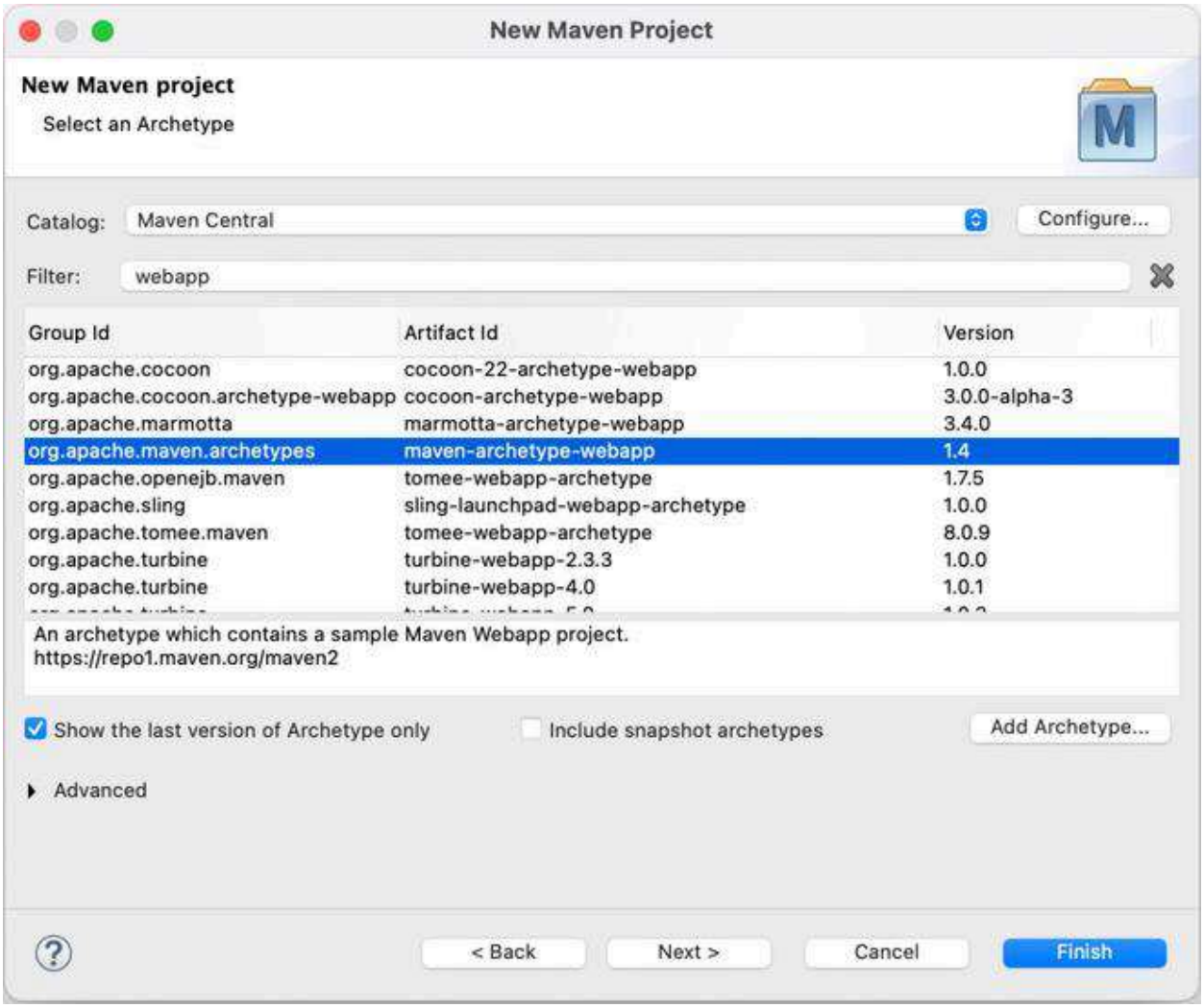
Spring MVC - Exception Handling

Last Updated : 07 Aug, 2025

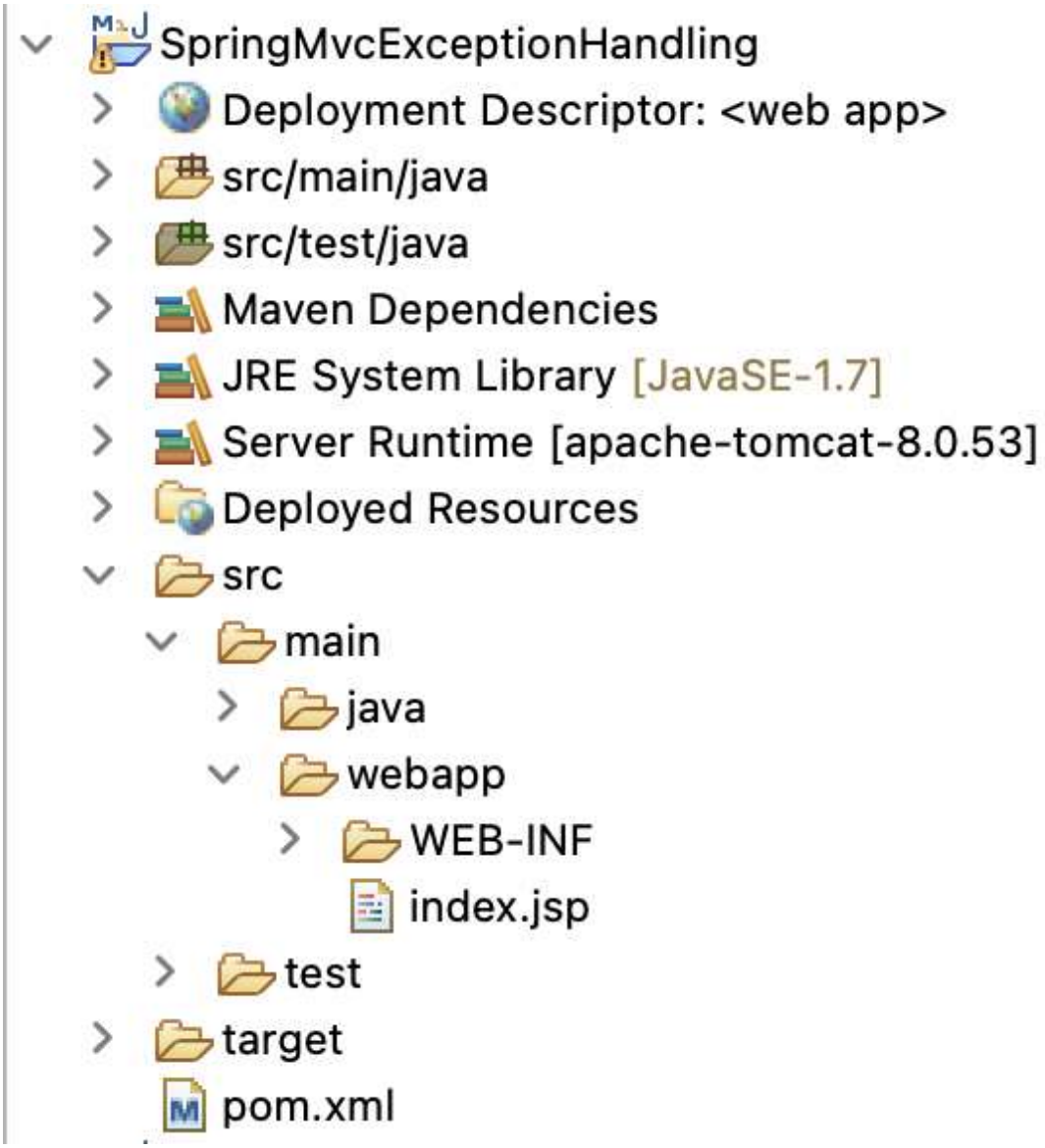
When something goes wrong in your app, the server shows a default error page, which isn't user-friendly. Spring MVC lets you handle exceptions in a cleaner way using the @ExceptionHandler annotation. It allows you to show custom error pages based on specific exceptions, either at the method or class level, improving the user experience.

Steps to Create the Application

First, create a maven project, we are using **Eclipse IDE** for this project. Now, search for webapp, as we are creating a web application. Choose to create maven while creating a new project and add a maven webapp archetype. Enter the group id and the artifact id for your project and click finish.



After clicking finish your project structure would look something like this:



The **pom.xml** is auto-created with any maven project, it defines all the dependencies required for the project. Make sure to add all the dependencies mentioned in this file.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
                               https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.gfg</groupId>
  <artifactId>SpringMvcExceptionHandling</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>SpringMvcExceptionHandling Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>
```

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.7</maven.compiler.source>
  <maven.compiler.target>1.7</maven.compiler.target>
</properties>

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.1.1.RELEASE</version>
  </dependency>
  <dependency>
      <groupId>org.apache.tomcat</groupId>
      <artifactId>tomcat-jasper</artifactId>
      <version>9.0.12</version>
  </dependency>
  <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>3.0-alpha-1</version>
  </dependency>
  <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
  </dependency>
</dependencies>

<build>
  <finalName>SpringMvcExceptionHandling</finalName>
  <pluginManagement>
    <plugins>
      <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>3.1.0</version>
      </plugin>
      <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <version>3.0.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
      <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.1</version>
      </plugin>
      <plugin>
```

```
            <artifactId>maven-war-plugin</artifactId>
            <version>3.2.2</version>
          </plugin>
          <plugin>
            <artifactId>maven-install-plugin</artifactId>
            <version>2.5.2</version>
          </plugin>
          <plugin>
            <artifactId>maven-deploy-plugin</artifactId>
            <version>2.8.2</version>
          </plugin>
        </plugins>
      </pluginManagement>
    </build>
</project>
```

The **web.xml** defines mapping with different URLs and servlets to handle requests for those URLs.

```
<web-app
xmlns="http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/index.html"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/index.html

http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">

    <servlet>
        <servlet-name>gfg</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/gfg-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>gfg</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

The **gfg-servlet.xml** file handles all HTTP requests for the web applications. The component scan locates and allocated beans according to the defined annotation. The annotation-driven enable the spring

annotation classes. The bean configuration helps in identifying and scanning the JSP located in the views folder.

```
<beans xmlns="http://www.springframework.org/schema/beans/"
    xmlns:context="http://www.springframework.org/schema/context/"
    xmlns:mvc="http://www.springframework.org/schema/mvc/"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans/
                        http://www.springframework.org/schema/beans//spring-
beans.xsd
                        http://www.springframework.org/schema/mvc/
                        http://www.springframework.org/schema/mvc//spring-
mvc.xsd
                        http://www.springframework.org/schema/context/ \

http://www.springframework.org/schema/context//spring-context.xsd">

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

class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix">
            <value>/WEB-INF/views/</value>
        </property>
        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>
    <mvc:annotation-driven />

</beans>
```

The **Student** class in the **com.gfg.model** defines the student object with three objects **firstName**, **lastName**, and **rollNo**. Notice that we have kept the roll number as a string instead of an integer, this will help us to check for possible NumberFormat exceptions.

```
package com.gfg.model;

public class Student {

    private String firstName;
    private String lastName;
    private String rollNo;

    public Student(String firstName, String lastName,
                   String rollNo)
    {
        super();
        this.firstName = firstName;
        this.lastName = lastName;
```

```
        this.rollNo = rollNo;
    }

    public Student() {}

    public String getFirstName() { return firstName; }

    public void setFirstName(String firstName)
    {
        this.firstName = firstName;
    }

    public String getLastName() { return lastName; }

    public void setLastName(String lastName)
    {
        this.lastName = lastName;
    }

    public String getRollNo() { return rollNo; }

    public void setRollNo(String rollNo)
    {
        this.rollNo = rollNo;
    }
}
```

- The LoginController has two methods: showForm (GET) to display the login form, and processForm to handle form data using @ModelAttribute and Model.
- The rollNo field in the Student class is a String but parsed to int, which may throw a NumberFormatException if it's empty or contains letters.
- To handle this, a method numberformatHandler is defined with @ExceptionHandler(NumberFormatException.class) to catch the error and improve user experience.

```
package com.gfg.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;

import com.gfg.model.Student;

@Controller
public class LoginController {
```

```
    @RequestMapping("/login")
    public String showForm(Model theModel) {
        theModel.addAttribute("student", new Student());
        return "portal";
    }
    @RequestMapping("/welcome")
    public String processForm(@ModelAttribute("welcome") Student student,
Model mod) {
        mod.addAttribute("FirstName", student.getFirstName());
        mod.addAttribute("LastName", student.getLastName());
        int n = Integer.parseInt(student.getRollNo());
        mod.addAttribute("RollNo", n);
        return "welcome";
    }
      @ExceptionHandler(value = NumberFormatException.class)
    public String numberformatHandler(Model theModel) {
        theModel.addAttribute("err", "NumberFormatException");
        return "error";
    }
}
```

- The MyExceptionHandler class handles all exceptions in the app and shows user-friendly error pages.
- By adding @ControllerAdvice, it applies to all controllers, allowing Spring MVC to use custom error methods instead of server-generated pages. This is an example of class-level exception handling.

```
package com.gfg.errorhandler;

import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
public class MyExceptionHandler {

    @ExceptionHandler(value = NullPointerException.class)
    public String nullPointerHandler(Model theModel) {
        theModel.addAttribute("err", "NullPointerException");
        return "error";
    }

    @ExceptionHandler(value = Exception.class)
    public String AnyOtherHandler() {
        return "error";
    }
}
```

The **portal.jsp** file in the views folder defines the Student login portal.

```
<%@ taglib prefix "form" uri "http://www.springframework.org/tags/form" %>
```

```
<%@ taglib prefix="form" url="http://www.springframework.org/tags/form" %>

 <html>
<head>
</head>
<body>
    <h1>Student Portal</h1>
    <form:form action="welcome" modelAttribute="student">
       <label>First name:</label>
         <form:input path="firstName" />
        <br><br>

         <label>Last name:</label>
         <form:input path="lastName" />
        <br><br>

         <label>Roll No:</label>
          <form:input path="rollNo" />
         <br><br>

        <input type="submit" value="Submit" />

    </form:form>
</body>
</html>
```

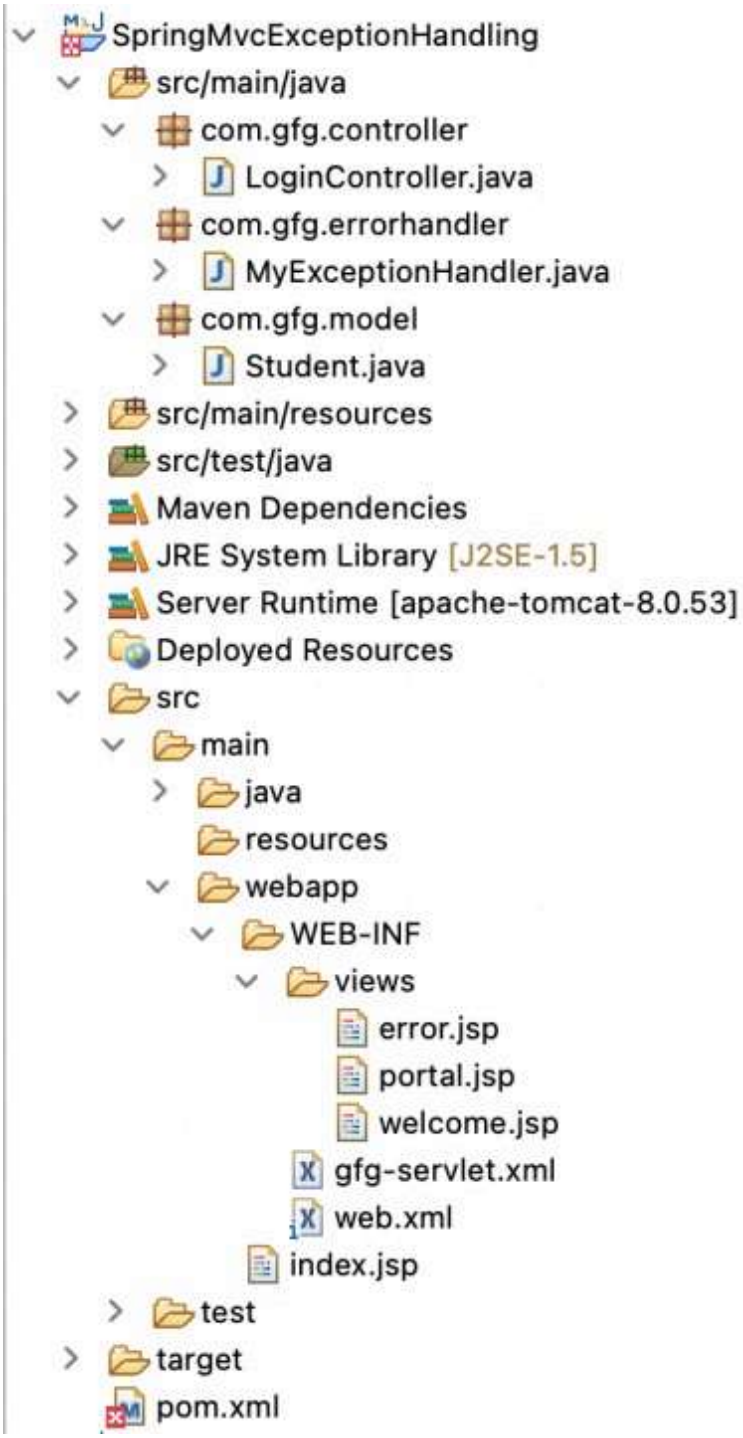
The **welcome.jsp** page in the views folder defines the welcome page for our application.

```
<%@ taglib prefix="form" url="http://www.springframework.org/tags/form" %>

 <html>
<head>
</head>
<body>
    <h1>Student Portal</h1>
    <form:form action="welcome" modelAttribute="student">
       <label>First name:</label>
         <form:input path="firstName" />
        <br><br>

         <label>Last name:</label>
         <form:input path="lastName" />
        <br><br>

         <label>Roll No:</label>
          <form:input path="rollNo" />
         <br><br>

        <input type="submit" value="Submit" />

    </form:form>
</body>
</html>
```

The **error.jsp** page is a simple exception handler page that defines the name of the exception and informs the user about an exception.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Opps....</h1>
    <h1> ${err} Exception caused</h1>
</body>
</html>
```

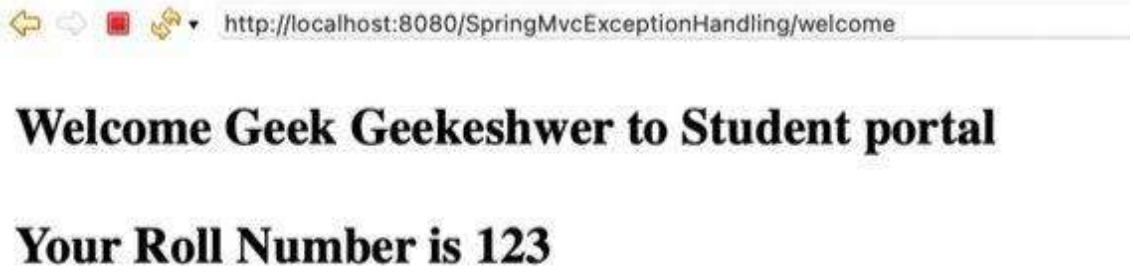
After adding all the classes and configuration files your project would look something like this:

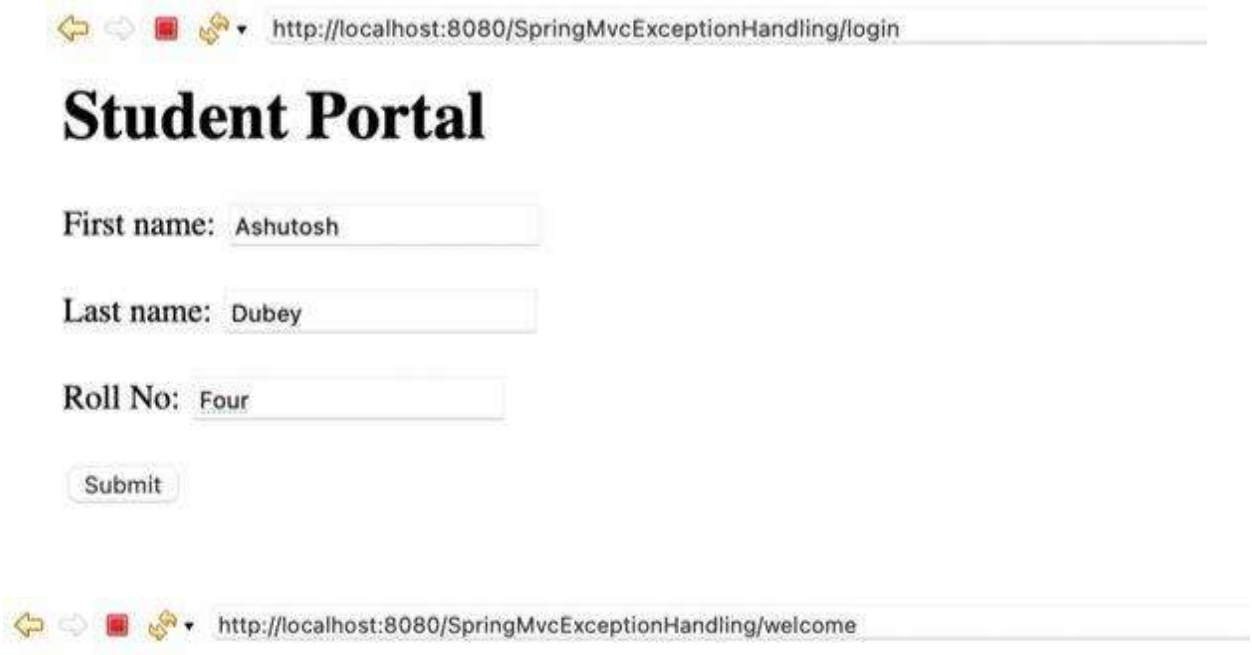


Now that we have completed our project, it's time to run it on a tomcat server, just start the tomcat server and type **http:localhost:8080/SpringMvcExceptionHandling/login**


http://localhost:8080/SpringMvcExceptionHandling/login
Student Portal
First name:
Last name:
Roll No:
Submit


http://localhost:8080/SpringMvcExceptionHandling/login
Student Portal
First name: Geek
Last name: Geekeshwer
Roll No: 123
Submit


http://localhost:8080/SpringMvcExceptionHandling/welcome
Welcome Geek Geekeshwer to Student portal
Your Roll Number is 123



Exception occured

Comment More info

Campus Training Program

GeeksforGeeks
Sanchhaya Education Private Limited
Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam