Develop an Interactive Spring MVC

Estimated time needed: 30 minutes

Overview

In this lab, you will be provided with the project structure for a Spring MVC (Model-View-Controller) application created using Spring web packages for web pages and Thymeleaf for templating. You will be adding interactivity in the application to provide for registration of the user details using a Model with built-in data validation.

Learning objectives

After completing this lab, you will be able to:

- Clone the Spring MVC app, import it in the workspace, and test it.
- Add the dependencies for data validation
- Add API endpoints in the controller
- Create webpages forms with Thymeleaf to bind the form fields to the model attributes
- Create Spring MVC validation on the model object using the @Valid annotation and the validate the data based on constraints defined in the model class
- Run the application

Prerequisites (optional)

You should know basic Java programming before you get started with this lab. Please ensure that you complete the labs sequentially as they are constructed progressively. Some background in HTML and CSS will be useful.

Clone the Spring MVC Project

1. Open a terminal and run the following command to clone the Spring MVC project.

   ```
   git clone https://github.com/ibm-developer-skills-network/utrde-spring_mvc_lab.git
   ```

2. Change to the application directory.

   ```
   cd /home/project/utrde-spring_mvc_lab/webdemo
   ```

3. Use `mvn clean` (maven clean) to clean any existing class files and and `mvn install` (maven install) to compile the files in the project directory and generate the runnable jar file. You can run them together as below to clean and install.

   ```
   mvn clean install
   ```

4. Run the following command to start the application from the jar file.

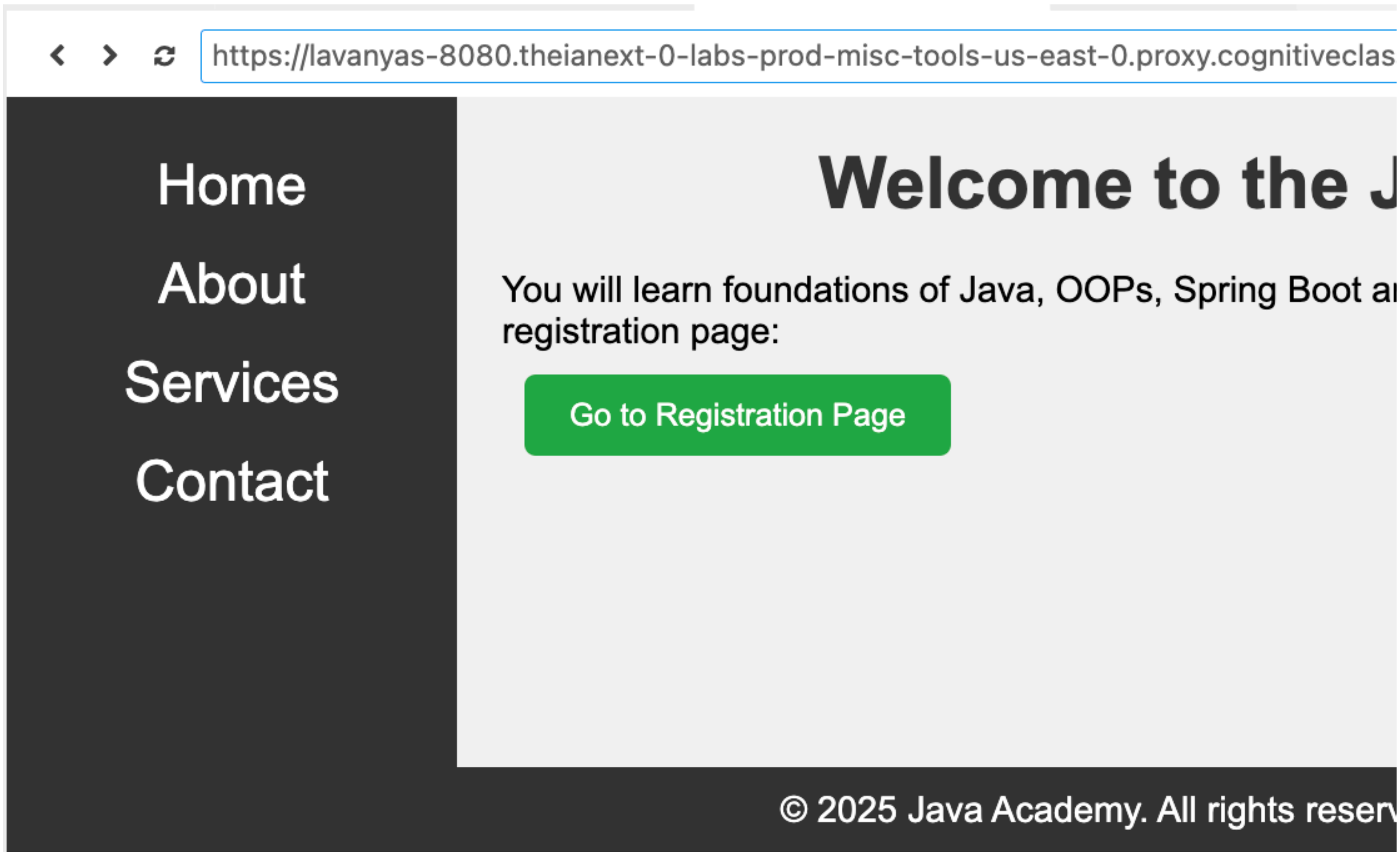
   ```
   java -jar target/webdemo-0.0.1-SNAPSHOT.jar
   ```

The server will start in port 8080.

5. Click on the button below to open the browser page to access the end point.

Website

You should see a page like this.



https://lavanyas-8080.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclas

Home
About
Services
Contact

Welcome to the J

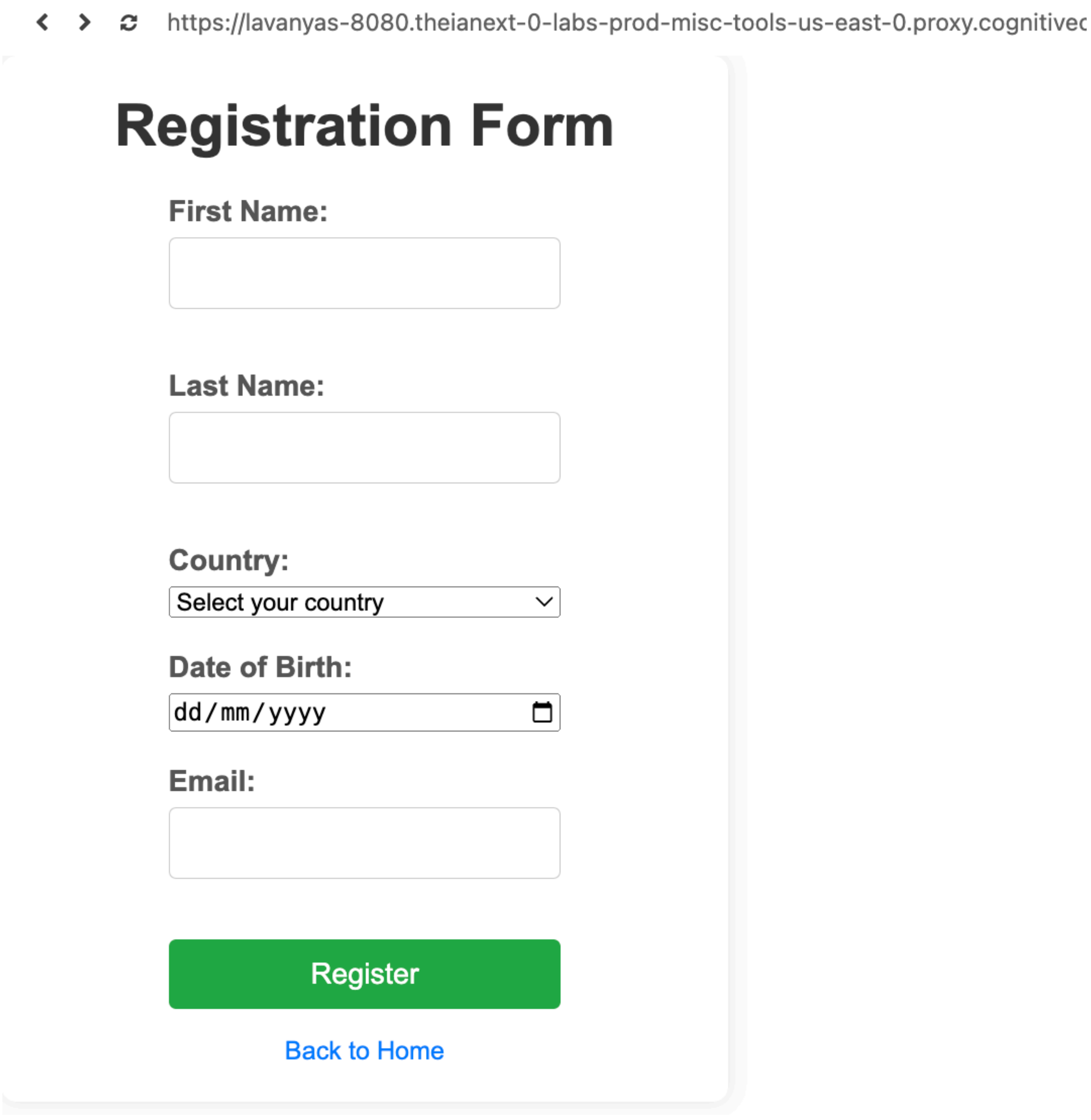
You will learn foundations of Java, OOPs, Spring Boot ar registration page:

Go to Registration Page

© 2025 Java Academy. All rights reser

6. Click `Go to Registration Page`.

7. It takes you to a registration page. But the page is not functional. You will configure this and add interaction and validation to your page in the next step.


https://lavanyas-8080.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitivec

8. Stop the server, by pressing `Ctrl+c` in the terminal.

Add code to handle registration

1. Select the button below to open the `pom.xml` and add the following code in it.

Open **pom.xml** in IDE

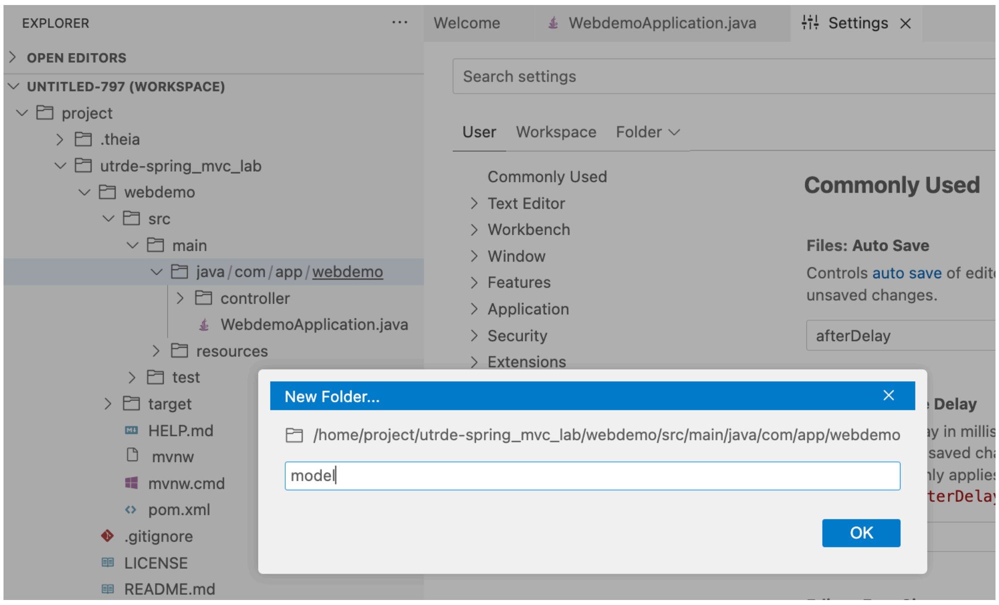
2. Ensure that the packaging is set to `jar`.

```
<groupId>com.app</groupId>
<artifactId>webdemo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>webdemo</name>
```

3. Add the following dependency in `pom.xml` along with the other dependencies to provide the packages to handle the data validation.

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
```

4. Add a new folder called model under `com/app/webdemo`.



New Folder...

/home/project/utrde-spring_mvc_lab/webdemo/src/main/java/com/app/webdemo

model

5. Add a new file under the folder `model` named `RegistrationForm.java`.

6. Copy and paste the following code in the file.

```
package com.app.webdemo.model;
import jakarta.validation.constraints.*;
import java.util.Date;
import org.springframework.format.annotation.DateTimeFormat;
public class RegistrationForm {
    @NotBlank(message = "First Name is required")
    private String firstName;
    @NotBlank(message = "Last Name is required")
    private String lastName;
    @NotBlank(message = "Country is required")
    private String country;
    @Past(message = "Date of Birth must be in the past")
    @DateTimeFormat(pattern = "yyyy-MM-dd") // Specify the date format
    private Date dob;
    @NotBlank(message = "Email is required")
    @Email(message = "Invalid email format")
    private String email;
    // Getters and Setters
    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 getCountry() {
        return country;
    }
    public void setCountry(String country) {
        this.country = country;
    }
    public Date getDob() {
        return dob;
    }
    public void setDob(Date dob) {
        this.dob = dob;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

The RegistrationForm class is a model class used in a Spring MVC application to represent the data entered in a user registration form.
It enforces validation rules on the form data using annotations.
When a user submits the registration form, the form data is bound to an instance of the RegistrationForm class. Spring MVC validates the RegistrationForm object using the annotations (@NotBlank, @Email, @Past, etc.). If any validation rule is violated, an error message is generated and stored in the BindingResult object.

7. Select the button below to open the `registration.html`.

Open **registration.html** in IDE

8. Paste the following code in `registration.html` replacing the existing content.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Page</title>
    <!-- Link to the external CSS file -->
    <link rel="stylesheet" th:href="@{/css/styles.css}">
</head>
<body>
    <div class="form-container">
        <h1>Registration Form</h1>
        <form action="/register" method="post" th:object="${registrationForm}">
            <!-- First Name -->
            <label for="firstName">First Name:</label>
            <input type="text" id="firstName" th:field="*{firstName}" required>
            <span th:if="${#fields.hasErrors('firstName')}" th:errors="*{firstName}" class="error"></span>
            <br>
            <!-- Last Name -->
            <label for="lastName">Last Name:</label>
            <input type="text" id="lastName" th:field="*{lastName}" required>
            <span th:if="${#fields.hasErrors('lastName')}" th:errors="*{lastName}" class="error"></span>
            <br>
            <!-- Country -->
            <label for="country">Country:</label>
            <select id="country" th:field="*{country}" required>
                <option value="">Select your country</option>
                <option value="usa">United States</option>
                <option value="canada">Canada</option>
                <option value="uk">United Kingdom</option>
                <option value="australia">Australia</option>
                <option value="india">India</option>
            </select>
            <span th:if="${#fields.hasErrors('country')}" th:errors="*{country}" class="error"></span>
            <br>
            <!-- Date of Birth -->
            <label for="dob">Date of Birth:</label>
            <input type="date" id="dob" th:field="*{dob}" required>
            <span th:if="${#fields.hasErrors('dob')}" th:errors="*{dob}" class="error"></span>
            <br>
            <!-- Email -->
            <label for="email">Email:</label>
            <input type="email" id="email" th:field="*{email}" required>
            <span th:if="${#fields.hasErrors('email')}" th:errors="*{email}" class="error"></span>
            <br>
            <!-- Submit Button -->
            <button type="submit">Register</button>
        </form>
        <!-- Link back to the home page -->
        <a th:href="@{/}">Back to Home</a>
    </div>
</body>
</html>
```

This HTML code is a Thymeleaf template for a user registration form. It integrates with Spring MVC to bind form fields to a model object (RegistrationForm) and display validation errors.

9. Select the button below to open the `com/app/webdemo/controller/WebdemoController.java`.

Open **registration.html** in IDE

10. Paste the following code in `WebdemoController.java` replacing the existing content.

```
package com.app.webdemo.controller;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import com.app.webdemo.model.RegistrationForm;
import jakarta.validation.Valid;
@Controller
public class WebdemoController {
    // Serve the index.html page
    @GetMapping("/")
    public String home() {
        return "index"; // returns Thymeleaf template index.html
    }
   // Display the registration form
    @GetMapping("/registration")
    public String showRegistrationForm(Model model) {
        model.addAttribute("registrationForm", new RegistrationForm());
        return "registration"; // returns Thymeleaf template registration.html
    }
    // Handle form submission
    @PostMapping("/register")
    public String handleRegistration(
            @Valid RegistrationForm registrationForm,
            BindingResult bindingResult,
            Model model) {
        // Check for validation errors
        if (bindingResult.hasErrors()) {
            return "registration"; // Return to the form with error messages
        }
        // Add form data to the model for the success page
        model.addAttribute("firstName", registrationForm.getFirstName());
        model.addAttribute("lastName", registrationForm.getLastName());
        model.addAttribute("country", registrationForm.getCountry());
        model.addAttribute("dob", registrationForm.getDob());
        model.addAttribute("email", registrationForm.getEmail());
        return "success"; // returns Thymeleafe template success.html
    }
}
```

11. Under templates, create `success.html` and add the following code in it. This is meant for displaying successful registration.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Successful</title>
    <!-- Link to the external CSS file -->
    <link rel="stylesheet" th:href="@{/css/styles.css}">
</head>
<body>
    <div class="form-container">
        <h1>Registration Successful!</h1>
        <p>Thank you for registering. Here are your details:</p>
        <ul>
            <li><strong>First Name:</strong> <span th:text="${firstName}"></span></li>
            <li><strong>Last Name:</strong> <span th:text="${lastName}"></span></li>
            <li><strong>Country:</strong> <span th:text="${country.toUpperCase()}"></span></li>
            <li><strong>Date of Birth:</strong> <span th:text="${dob}"></span></li>
            <li><strong>Email:</strong> <span th:text="${email}"></span></li>
        </ul>
        <a th:href="@{/}">Back to Home</a>
    </div>
</body>
</html>
```

Run the jar and test the code

3. Use `mvn clean install` to clean and install the application and generate jar file.

```
mvn clean install
```

4. Run the following command to start the application from the jar file.

```
java -jar target/webdemo-0.0.1-SNAPSHOT.jar
```

The server will start in port 8080.

4. Click on the button below to open the browser page to access the end point.

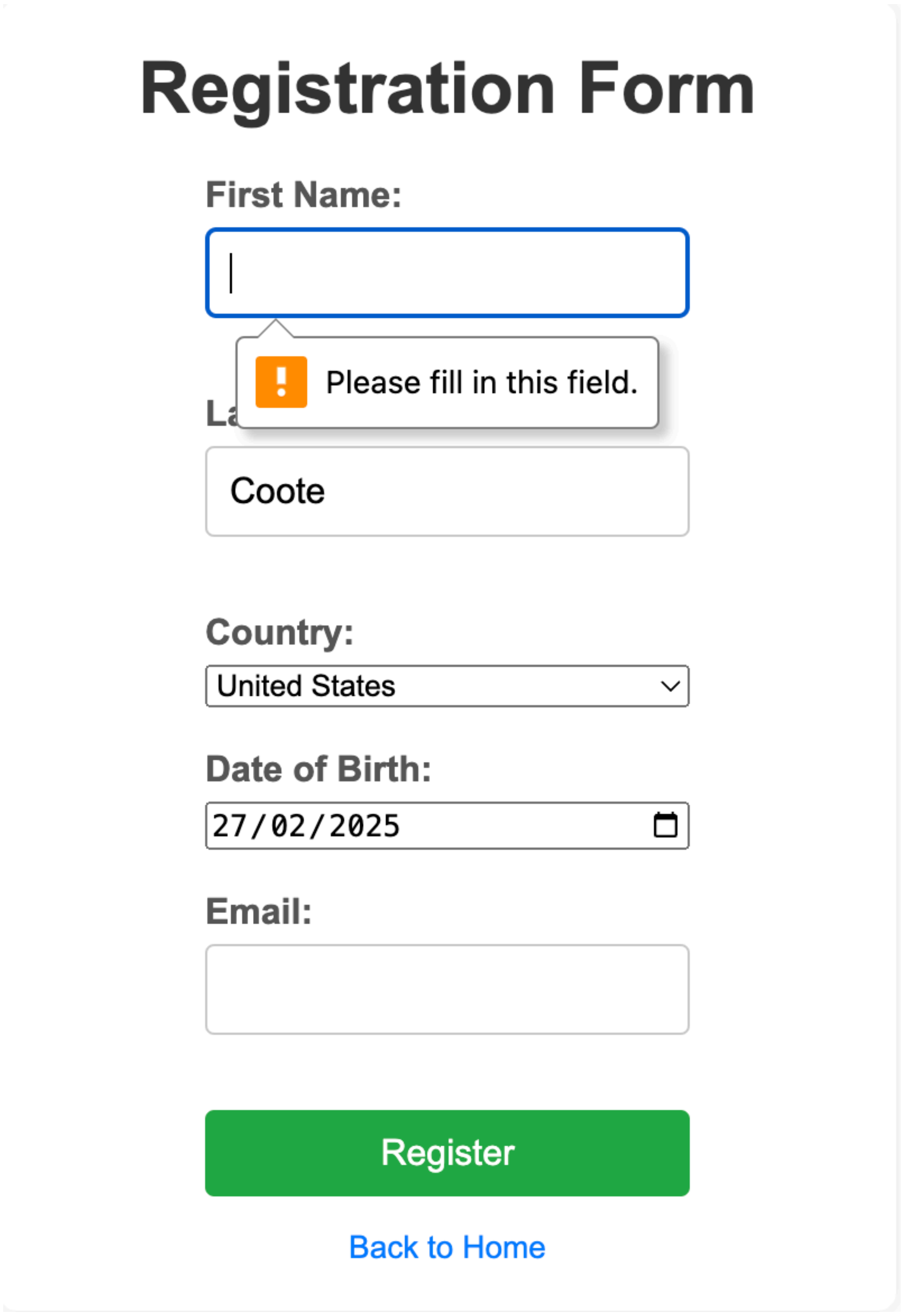
Website

5. The home page opens up. Click `Go to Registration Page`.

6. Enter incorrect data to register to test the validation.

   - Try to leave a few fields blank
   - Try to enter a Date of birth in future date
   - Try to enter an invalid email id

You get appropriate error messages displayed.



Registration Form

First Name:

Please fill in this field.

La

Coote

Country:

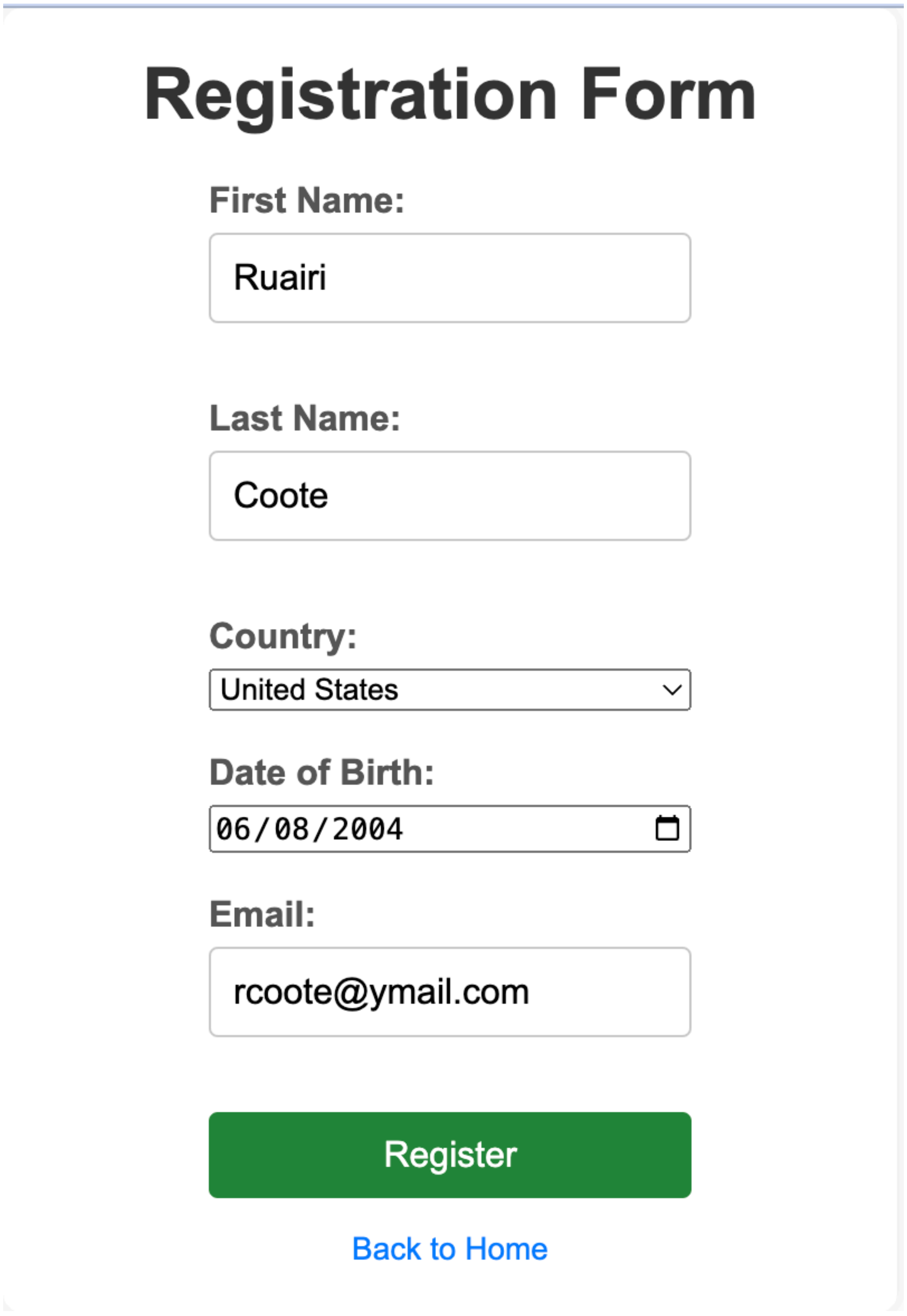
United States

Date of Birth:

27/02/2025

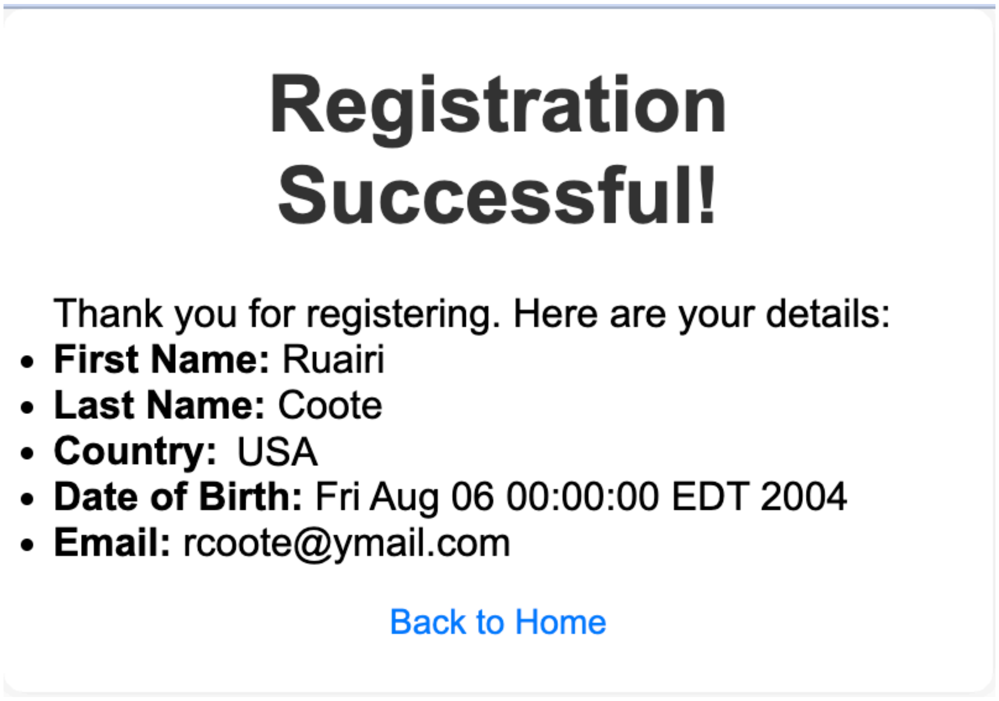
Email:

7. Enter correct data and click `Register`.


Registration Form

First Name:	Ruairi
Last Name:	Coote
Country:	United States
Date of Birth:	06/08/2004
Email:	rcoote@ymail.com

8. When the registration is successful, the page is displayed with success message.


Registration Successful!

Thank you for registering. Here are your details:

- **First Name:** Ruairi
- **Last Name:** Coote
- **Country:** USA
- **Date of Birth:** Fri Aug 06 00:00:00 EDT 2004
- **Email:** rcoote@ymail.com

Practice exercise

1. Add another field in the registration form to enter the annual income.
2. Make changes to the `RegistrationForm.java` to validate the income.
3. Test the form to see whether it is handling incorrect data entry.

Conclusion

In this lab you have:

- Cloned a Spring MVC app, imported it in the workspace, and tested it
- Added the dependencies for data validation
- Added API endpoints in the controller
- Created webpage forms with Thymeleaf
- Created Spring MVC validation on the model object using the @Valid annotation and validated the data based on constraints defined in the model class

Author(s)

Lavanya