D1-UD4 Sergio Berrendero Toledano

https://github.com/SergioBerren/DWES-Spring/tree/master/D1UD4

La forma de crear los proyectos es la siguiente, pulsamos click derecho y New > Spring Starter Project, cambiamos el nombre del proyecto al que queramos y también cambiamos el nombre del package de ahí elegimos las dependencias que vamos a necesitar (Spring Boot DevTools, Spring Web y Thymeleaf), una vez elegidas pulsamos Next y al ver que todo está correcto pulsamos Finish. Esperamos a que termine de crearlo y una vez que ha acabado manipularemos src/main/java y src/main/resources. En src/main/java/[nombre del paquete] crearemos las clases java (hay una que se llama [nombre del proyecto]Application que la crea automáticamente y es necesaria para que funcione correctamente), y en src/main/resources crearemos los html (normalmente los crearemos en src/main/resources/templates, pero también se pueden crear archivos html en src/main/resources/static para que en vez del WhiteLabel Error Page nos salga dicho html).

Apartado 1

En este apartado se nos pide realizar tres tutoriales diferentes y pasarlos a castellano, de esta forma podremos ver distintos errores que nos puede dar el código y su solución. Los tutoriales son los siguientes:

a) https://spring.io/guides/gs/serving-web-content/
b) https://spring.io/guides/gs/handling-form-submission/
c) https://spring.io/guides/gs/validating-form-input/

a) En este tutorial se nos pedía crear una página que nos mostrase un “Hola Mundo”, para ello hemos creado las siguientes clases:

SaludoController.java:

Esta clase es un controlador de Spring que maneja una solicitud HTTP GET en la URL /saludo. El método saludo() acepta un parámetro nombre (que se pasa como parámetro de consulta) y lo muestra en una vista. Si no se proporciona un nombre, el valor predeterminado es "Mundo". Los valores se pasan al modelo para que sean accesibles en la vista, y se devuelve el nombre de la vista saludo.

```java
package com.example.saludo;

import org.springframework.stereotype.Controller;

@Controller
public class SaludoController {
    @GetMapping("/saludo")
    public String saludo(@RequestParam(name="nombre", required=false, defaultValue = "Mundo") String nombre, Model modelo) {
    // nombre = "Sergio";
    modelo.addAttribute("name", nombre);
    return "saludo";
    }
}
```

saludo.html:

Este archivo HTML es una plantilla Thymeleaf que muestra el saludo personalizado usando el valor de la variable *name* que se pasó desde el controlador. El atributo *th:text* muestra el saludo. Al no recibir ningún parámetro *name* da Mundo como valor por defecto

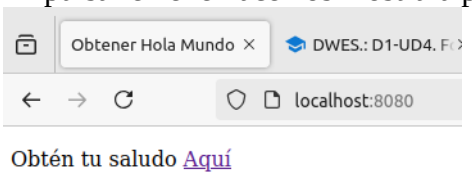
```html
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <title>Página Saludo</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>
    <p th:text="|¡¡¡Hola, ${name}!!!|" />
    <!--name viene de un atributo name que ponemos en modelo.addAttribute("name", nombre);-->
</body>

</html>
```

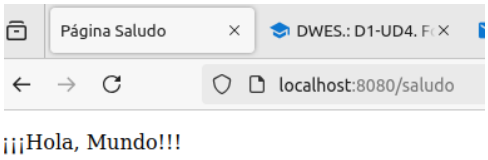
index.html:
Esta clase es esa excepción que mencionamos anteriormente, debido a que esta se crea en el static y no en templates.

Este código HTML muestra un enlace el cual te redirige a localhost:8080/saludo el cual muestra "¡¡¡Hola Mundo!!!"

```html
<!DOCTYPE HTML>
<html>
<head>
    <title>Obtener Hola Mundo</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <p>Obtén tu saludo <a href="/saludo">Aquí</a></p>
</body>
</html>
```

Funcionamiento:
Al pulsar en el enlace nos mostrará por pantalla el siguiente mensaje.





b) En este tutorial se nos pide crear un formulario en el que pongas un id y un mensaje y te muestra los datos por pantalla. Para ello hemos creado dos clases (sin contar la clase por defecto) y dos archivos html.

Datos.java:
Esta clase es la encargada de capturar la información del formulario (id y contenido), en ella se declaran las variables que necesitaremos y los getters y setters. Se utiliza como una estructura para almacenar los datos de un formulario enviado.

```java
package com.example.manejoSolicitudEnvioFormularioUsuarios;

public class Datos {

  private long id;
  private String contenido;

  public long getId() {
    return id;
  }

  public void setId(long id) {
    this.id = id;
  }

  public String getContenido() {
    return contenido;
  }

  public void setContenido(String contenido) {
    this.contenido = contenido;
  }

}
```

DatosController.java

Este es otro controlador de Spring, que gestiona la lógica para mostrar y procesar un formulario. Usa *@ModelAttribute* para enlazar un objeto Datos con el formulario.

```java
package com.example.manejoSolicitudEnvioFormularioUsuarios;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class DatosController {

  @GetMapping("/formulario")
  public String datosFormulario(Model modelo) {
    modelo.addAttribute("formulario", new Datos());
    return "formulario";
  }

  @PostMapping("/formulario")
  public String datosEnviados(@ModelAttribute Datos datos, Model modelo) {
    modelo.addAttribute("formulario", datos);
    return "resultado";
  }
}
```

formulario.html

Este formulario permite a los usuarios ingresar un id y un contenido, que se envían al controlador para su procesamiento.

```html
<!DOCTYPE HTML>
<html xmlns:th="https://www.thymeleaf.org">
<head>
    <title>Formulario crear usuario</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <h1>Formulario</h1>
    <form action="#" th:action="@{/formulario}" th:object="${formulario}" method="post">
        <p>Id: <input type="text" th:field="*{id}" /></p>
        <p>Mensaje: <input type="text" th:field="*{contenido}" /></p>
        <p><input type="submit" value="Enviar" /> <input type="reset" value="Reiniciar" /></p>
    </form>
</body>
</html>
```

resultado.html
Este archivo muestra los resultados del formulario, como el id y el contenido ingresado, utilizando Thymeleaf para acceder a los valores del modelo.

```
<!DOCTYPE HTML>
<html xmlns:th="https://www.thymeleaf.org">
<head>
    <title>Resultado</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <h1>Resultado</h1>
    <p th:text="'id: ' + ${datos.id}" />
    <p th:text="'contenido: ' + ${datos.contenido}" />
    <a href="/formulario">Envía otro mensaje</a>
</body>
</html>
```

Funcionamiento:
Esta será la página de inicio, en caso de que se pulse el botón reiniciar o al enlace que veremos más adelante nos mostrará esto.

Escribimos el id que deseamos y el mensaje y damos a enviar.

Una vez enviado nos muestra los datos por pantalla y un enlace para volver a empezar (nombrado anteriormente).

Resultado

id: 100

contenido: Hola

Envía otro mensaje

c) En este apartado se nos pedía hacer un formulario en el que hay que rellenar nombre y edad y hacer una validación de la información pasada por el usuario. En caso de que el nombre sea menor a dos caracteres y/o la edad sea menor a 18 dará un error y pedirá al usuario que ponga datos que estén permitidos. Para ello hemos creado las siguientes clases:

PersonaFormulario.java:
Esta clase representa los datos de un formulario de usuario. Incluye validaciones de *JSR-303* con anotaciones como *@NotNull*, *@Min*, y *@Size* para asegurar que los valores sean válidos antes de enviarlos.

```java
package com.example.miTercerFormulario;

import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;

public class PersonaFormulario {

    @NotNull
    @Size(min=2, max=30)
    private String nombre;

    @NotNull
    @Min(18)
    private Integer edad;

    public String getNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Integer getEdad() {
        return edad;
    }

    public void setEdad(Integer edad) {
        this.edad = edad;
    }

    public String toString() {
        return "Person(Nombre: " + this.nombre + ", Edad: " + this.edad + ")";
    }
}
```

WebController.java:
Este controlador maneja el formulario PersonaFormulario. Utiliza validación con *@Valid* y BindingResult para asegurarse de que los datos son correctos antes de enviarlos.

@Valid: Realiza la validación del objeto PersonaFormulario.
BindingResult: Permite manejar los errores de validación.

```java
package com.example.miTercerFormulario;

import jakarta.validation.Valid;

import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;


@Controller
public class WebController implements WebMvcConfigurer {

    @Override
    public void addViewControllers(ViewControllerRegistry registro) {
        registro.addViewController("/resultado").setViewName("resultado");
    }

    @GetMapping("/")
    public String mostrarFormulario(PersonaFormulario personaFormulario) {
        return "formulario";
    }

    @PostMapping("/")
    public String checkInformacionPersona(@Valid PersonaFormulario personaFormulario, BindingResult bindingResult) {

        if (bindingResult.hasErrors()) {
            return "formulario";
        }

        return "redirect:/resultado";
    }
}
```

formulario.html:

Este es el formulario que captura los datos de la persona, como nombre, edad y otras propiedades. Usa Thymeleaf para enlazar los campos con el objeto PersonaFormulario y mostrar errores de validación.

```html
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
    <body>
        <form action="#" th:action="@{/}" th:object="${personaFormulario}" method="post">
            <table>
                <tr>
                    <td>Nombre:</td>
                    <td><input type="text" th:field="*{nombre}" /></td>
                    <td th:if="${#fields.hasErrors('nombre')}" th:errors="*{nombre}">Nombre Error</td>
                </tr>
                <tr>
                    <td>Edad:</td>
                    <td><input type="text" th:field="*{edad}" /></td>
                    <td th:if="${#fields.hasErrors('edad')}" th:errors="*{edad}">Edad Error</td>
                </tr>
                <tr>
                    <td><button type="submit">Enviar</button></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

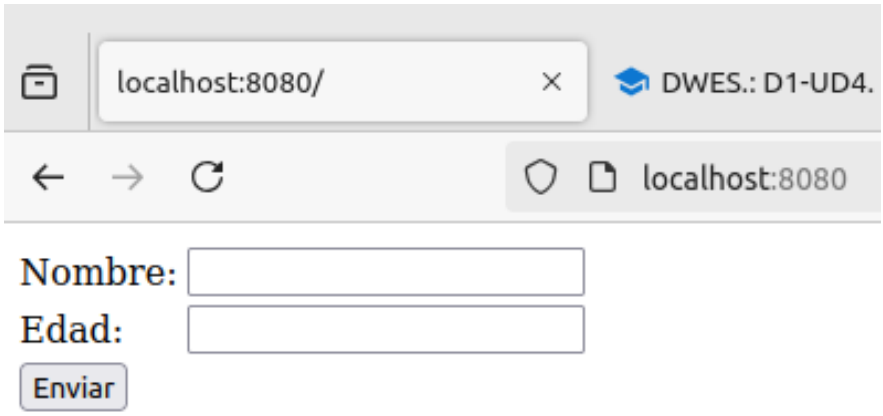
resultado.html:

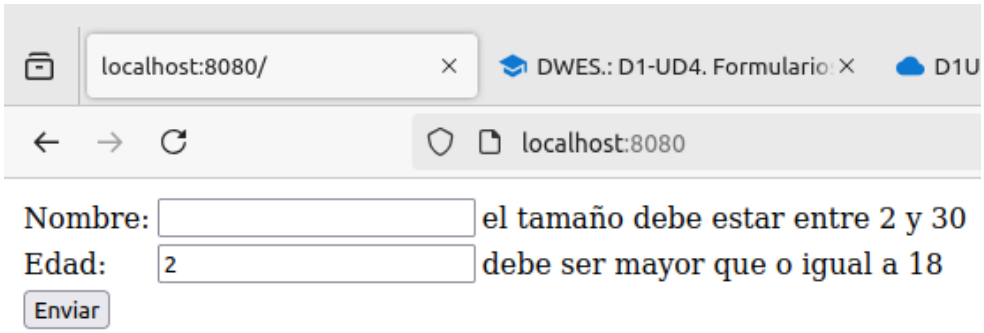
Este HTML muestra un mensaje indicando que la persona es lo suficientemente mayor para registrarse. Es la vista de éxito tras enviar un formulario válido.

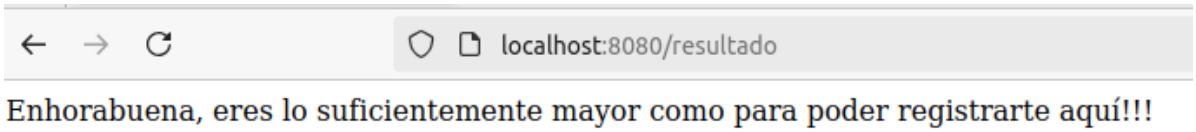
```html
<html>
    <body>
        Enhorabuena, eres lo suficientemente mayor como para poder registrarte aquí!!!
    </body>
</html>
```

Funcionamiento:

Relleno los campos y doy a enviar, si hay algún error nos mostrará el error y si no lo hay nos mostrará el siguiente mensaje







Apartado 2
En este apartado se nos pide desarrollar un formulario el cual sea capaz de calcular el índice de grasa corporal. Para ello creamos las siguientes clases:

DatosGrasaCorporal.java:
Esta clase contiene la lógica para calcular el porcentaje de grasa corporal, utilizando el Índice de Masa Corporal (IMC) y otros parámetros como el perímetro abdominal (PA) y la masa corporal (MC). También incluye métodos para clasificar el resultado de la grasa corporal.

```java
package com.example.grasaCorporal;

public class DatosGrasaCorporal {

    public double calcularGrasaMujeres(double imc, double pa) {
        return ((imc / pa) * 10) + imc + 10;
    }

    public double calcularGrasaHombres(double mc, double pa, double imc) {
        return ((mc / pa) * 10) + imc;
    }

    public String clasificarHombres(double porcentajeGrasa) {
        if (porcentajeGrasa < 6) {
            return "Esencial";
        } else if (porcentajeGrasa < 14) {
            return "Atleta";
        } else if (porcentajeGrasa < 18) {
            return "Fitness";
        } else if (porcentajeGrasa < 25) {
            return "Aceptable";
        } else {
            return "Obesidad";
        }
    }

    public String clasificarMujeres(double porcentajeGrasa) {
        if (porcentajeGrasa < 14) {
            return "Esencial";
        } else if (porcentajeGrasa < 21) {
            return "Atleta";
        } else if (porcentajeGrasa < 25) {
            return "Fitness";
        } else if (porcentajeGrasa < 32) {
            return "Aceptable";
        } else {
            return "Obesidad";
        }
    }
}
```

GrasaCorporalController.java:

Este controlador maneja las solicitudes GET y POST para calcular el porcentaje de grasa corporal, dependiendo de los parámetros del formulario. Utiliza la clase DatosGrasaCorporal para realizar los cálculos y luego pasa los resultados a la vista.

```java
package com.example.grasaCorporal;

import org.springframework.stereotype.Controller;

@Controller
public class GrasaCorporalController {

    // Instancia de la lógica de negocio
    private DatosGrasaCorporal datos = new DatosGrasaCorporal();

    @GetMapping("/")
    public String formulario() {
        return "formulario";
    }

    @PostMapping("/calcular")
    public String calcularGrasa(
            @RequestParam("genero") String genero,
            @RequestParam("imc") double imc,
            @RequestParam("pa") double pa,
            @RequestParam(value = "mc", required = false) Double mc,
            Model model) {

        double porcentajeGrasa;
        String clasificacion;

        // Realizar el cálculo según el género
        if ("hombre".equals(genero) && mc != null) {
            porcentajeGrasa = datos.calcularGrasaHombres(mc, pa, imc);
            clasificacion = datos.clasificarHombres(porcentajeGrasa);
        } else {
            porcentajeGrasa = datos.calcularGrasaMujeres(imc, pa);
            clasificacion = datos.clasificarMujeres(porcentajeGrasa);
        }

        // Pasar datos a la vista
        model.addAttribute("porcentajeGrasa", porcentajeGrasa);
        model.addAttribute("clasificacion", clasificacion);

        return "resultado";
    }
}
```

formulario.html:

Esta HTML es un formulario donde los usuarios ingresan su género, IMC, PA y MC (para hombres), y luego pueden calcular su porcentaje de grasa corporal.

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Calcular Grasa Corporal</title>
</head>
<body>
    <h1>Calcular Grasa Corporal</h1>
    <form action="/calcular" method="post">
        <label for="genero">Género:</label>
        <select id="genero" name="genero">
            <option value="hombre">Hombre</option>
            <option value="mujer">Mujer</option>
        </select><br><br>

        <label for="imc">IMC:</label>
        <input type="number" id="imc" name="imc" step="0.1" required><br><br>

        <label for="pa">Perímetro Abdominal (PA):</label>
        <input type="number" id="pa" name="pa" step="0.1" required><br><br>

        <label for="mc">Masa Corporal (MC, solo hombres):</label>
        <input type="number" id="mc" name="mc" step="0.1"><br><br>

        <button type="submit">Calcular</button>
    </form>
</body>
</html>
```

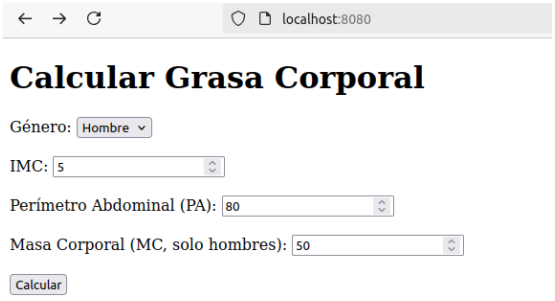
resultado.html:

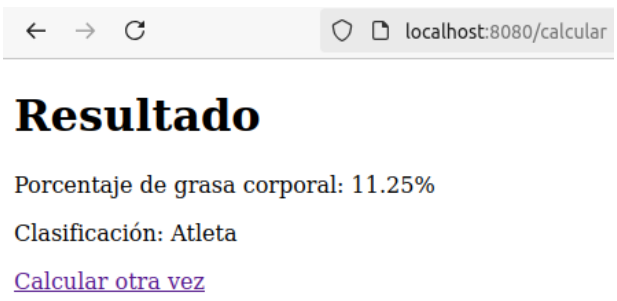
Este archivo HTML muestra los resultados del cálculo del porcentaje de grasa corporal y su clasificación, dependiendo de los valores ingresados en el formulario.

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Resultado</title>
</head>
<body>
    <h1>Resultado</h1>
    <p>Porcentaje de grasa corporal: <span th:text="${porcentajeGrasa}"></span>%</p>
    <p>Clasificación: <span th:text="${clasificacion}"></span></p>
    <a href="/">Calcular otra vez</a>
</body>
</html>
```

Funcionamiento:

El funcionamiento de esta aplicación es muy sencillo, el usuario simplemente tiene que rellenar los datos requeridos y dependiendo de la opción seleccionada hace unos cálculos u otros y dependiendo de el resultado muestra un mensaje u otro.





Apartado 3

En este apartado se nos pide crear un formulario más avanzado y su validación, para ello debemos añadir la dependencia Validation además de las anteriormente nombradas.

DatosFormulario.java

La clase DatosFormulario tiene varios campos con validaciones para nombre, correo electrónico, fecha de nacimiento, etc. Estos campos se utilizan en un formulario para registrar usuarios.

```java
package com.example.formularioAvanzado;

import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Past;
import jakarta.validation.constraints.Size;
import java.time.LocalDate;
import java.util.List;

public class DatosFormulario {

    @NotEmpty(message = "El nombre es obligatorio")
    private String nombre;

    @NotEmpty(message = "El primer apellido es obligatorio")
    private String primerApellido;

    @NotEmpty(message = "El segundo apellido es obligatorio")
    private String segundoApellido;

    @Email(message = "El correo electrónico debe ser válido")
    @NotEmpty(message = "El correo electrónico es obligatorio")
    private String email;

    @NotNull(message = "La fecha de nacimiento es obligatoria")
    @Past(message = "La fecha de nacimiento debe ser en el pasado")
    private LocalDate fechaNacimiento;

    @NotEmpty(message = "La contraseña es obligatoria")
    @Size(min = 6, message = "La contraseña debe tener al menos 6 caracteres")
    private String contrasena;

    @NotEmpty(message = "El sexo es obligatorio")
    private String sexo;

    @NotEmpty(message = "La dirección no puede estar vacía")
    private String direccion;

    @NotEmpty(message = "El nivel de estudios es obligatorio")
    private String nivelEstudios;

    @NotEmpty(message = "Debe seleccionar al menos un tema de interés")
    private List<String> temasInteres;

    // Getters y Setters
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getPrimerApellido() {
        return primerApellido;
    }
```

```java
public void setPrimerApellido(String primerApellido) {
    this.primerApellido = primerApellido;
}

public String getSegundoApellido() {
    return segundoApellido;
}

public void setSegundoApellido(String segundoApellido) {
    this.segundoApellido = segundoApellido;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public LocalDate getFechaNacimiento() {
    return fechaNacimiento;
}

public void setFechaNacimiento(LocalDate fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
}

public String getContrasena() {
    return contrasena;
}

public void setContrasena(String contrasena) {
    this.contrasena = contrasena;
}

public String getSexo() {
    return sexo;
}

public void setSexo(String sexo) {
    this.sexo = sexo;
}

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getNivelEstudios() {
    return nivelEstudios;
}
```

```java
    public void setNivelEstudios(String nivelEstudios) {
        this.nivelEstudios = nivelEstudios;
    }

    public List<String> getTemasInteres() {
        return temasInteres;
    }

    public void setTemasInteres(List<String> temasInteres) {
        this.temasInteres = temasInteres;
    }
}
```

FormularioController.java

Este controlador maneja un formulario avanzado con varios campos y validaciones, mostrando el resultado después de procesar los datos.

```java
package com.example.formularioAvanzado;

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 jakarta.validation.Valid;

@Controller
public class FormularioController {

    @GetMapping("/formulario")
    public String mostrarFormulario(DatosFormulario datosFormulario) {
        return "formulario";
    }

    @PostMapping("/formulario")
    public String procesarFormulario(@Valid DatosFormulario datosFormulario, BindingResult resultado, Model modelo) {
        if (resultado.hasErrors()) {
            return "formulario"; // Si hay errores, se regresa al formulario
        }

        modelo.addAttribute("formulario", datosFormulario); // Pasa el formulario validado al modelo
        return "resultado"; // Redirige a la vista "resultado.html"
    }

}
```

formulario.html

Este archivo HTML muestra un formulario de registro con varios campos como nombre, apellido, correo electrónico, fecha de nacimiento, etc. Cada campo está validado y utiliza Thymeleaf para mostrar mensajes de error si no se cumplen las validaciones.

```html
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Formulario de Registro</title>
    </head>

    <body>
        <div>
            <h2>Formulario de Registro</h2>
            <form action="/formulario" method="post" th:action="@{/formulario}" th:object="${datosFormulario}">
                <fieldset>
                    <legend>Datos Personales</legend>
                    <!-- Nombre -->
                    <div>
                        <label for="nombre">Nombre</label>
                        <input type="text" id="nombre" th:field="*{nombre}" placeholder="Nombre">
                        <div th:if="${#fields.hasErrors('nombre')}" th:errors="*{nombre}"></div>
                    </div>

                    <!-- Primer Apellido -->
                    <div>
                        <label for="primerApellido">Primer Apellido</label>
                        <input type="text" id="primerApellido" th:field="*{primerApellido}" placeholder="Primer Apellido">
                        <div th:if="${#fields.hasErrors('primerApellido')}" th:errors="*{primerApellido}"></div>
                    </div>

                    <!-- Segundo Apellido -->
                    <div>
                        <label for="segundoApellido">Segundo Apellido</label>
                        <input type="text" id="segundoApellido" th:field="*{segundoApellido}"
                            placeholder="Segundo Apellido">
                        <div th:if="${#fields.hasErrors('segundoApellido')}" th:errors="*{segundoApellido}"></div>
                    </div>

                    <!-- Correo electrónico -->
                    <div>
                        <label for="email">Correo Electrónico</label>
                        <input type="email" id="email" th:field="*{email}" placeholder="Correo Electrónico">
                        <div th:if="${#fields.hasErrors('email')}" th:errors="*{email}"></div>
                    </div>

                    <!-- Fecha de nacimiento -->
                    <div>
                        <label for="fechaNacimiento">Fecha de Nacimiento</label>
                        <input type="date" id="fechaNacimiento" th:field="*{fechaNacimiento}">
                        <div th:if="${#fields.hasErrors('fechaNacimiento')}" th:errors="*{fechaNacimiento}"></div>
                    </div>

                    <!-- Contraseña -->
                    <div>
                        <label for="contrasena">Contraseña</label>
                        <input type="password" id="contrasena" th:field="*{contrasena}" placeholder="Contraseña">
                        <div th:if="${#fields.hasErrors('contrasena')}" th:errors="*{contrasena}"></div>
                    </div>

                    <!-- Sexo -->
                    <div>
```

```
                <!-- Sexo -->
                <div>
                    <label>Sexo</label>
                    <div>
                        <input type="radio" id="masculino" th:field="*{sexo}" value="Varón"> Varón<br>
                        <input type="radio" id="femenino" th:field="*{sexo}" value="Mujer"> Mujer<br>
                    </div>
                    <div th:if="${#fields.hasErrors('sexo')}" th:errors="*{sexo}"></div>
                </div>

                <!--Dirección-->
                <div>
                    <label for="direccion">Dirección</label>
                    <input type="text" id="direccion" th:field="*{direccion}"
                        placeholder="Calle o Plaza, número y piso">
                    <div th:if="${#fields.hasErrors('direccion')}" th:errors="*{direccion}"></div>
                </div>
            </fieldset><br>
            <fieldset>
                <legend>Nivel de estudios e intereses</legend>
                <!-- Nivel de estudios -->
                <div>
                    <label>Nivel de Estudios</label>
                    <div>
                        <input type="radio" th:field="*{nivelEstudios}" value="Certificado escolar"> Certificado escolar<br>
                        <input type="radio" th:field="*{nivelEstudios}" value="Graduado de la ESO"> Graduado en ESO<br>
                        <input type="radio" th:field="*{nivelEstudios}"
                            value="Bachiller - Formación Profesional">Bachiller - Formación profesional<br>
                        <input type="radio" th:field="*{nivelEstudios}" value="Diplomado"> Diplomado<br>
                        <input type="radio" th:field="*{nivelEstudios}" value="Licenciado"> Licenciado<br>
                        <input type="radio" th:field="*{nivelEstudios}" value="Doctorado"> Doctorado
                    </div>
                    <div th:if="${#fields.hasErrors('nivelEstudios')}" th:errors="*{nivelEstudios}"></div>
                </div>

                <!-- Temas de Interés -->
                <div>
                    <label>Temas de Interés</label>
                    <div>
                        <input type="checkbox" th:field="*{temasInteres}" value="Música"> Música<br>
                        <input type="checkbox" th:field="*{temasInteres}" value="Deportes"> Deportes<br>
                        <input type="checkbox" th:field="*{temasInteres}" value="Cine"> Cine<br>
                        <input type="checkbox" th:field="*{temasInteres}" value="Libros"> Libros<br>
                        <input type="checkbox" th:field="*{temasInteres}" value="Ciencia"> Ciencia<br>
                        <input type="checkbox" th:field="*{temasInteres}" value="Tecnología"> Tecnología
                    </div>
                    <div th:if="${#fields.hasErrors('temasInteres')}" th:errors="*{temasInteres}"></div>
                </div>
            </fieldset><br>
            <fieldset>
                <legend>Enviar formulario</legend>
                <!-- Botón Enviar -->
                <p>¿Estás seguro de que los datos son correctos?</p>
                <button type="submit">Enviar</button>
            </fieldset>
        </form>
    </div>
</body>
</html>
```

resultado.html

Este archivo HTML muestra los resultados del formulario después de ser enviado. Aquí se muestran los datos ingresados por el usuario.

```html
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Resultados del Formulario</title>
    </head>

    <body>
        <div>
            <h2>Datos del Formulario</h2>
            <table>
                <tr>
                    <td>Nombre</td>
                    <td th:text="${formulario.nombre}"></td>
                </tr>
                <tr>
                    <td>Primer Apellido</td>
                    <td th:text="${formulario.primerApellido}"></td>
                </tr>
                <tr>
                    <td>Segundo Apellido</td>
                    <td th:text="${formulario.segundoApellido}"></td>
                </tr>
                <tr>
                    <td>Correo Electrónico</td>
                    <td th:text="${formulario.email}"></td>
                </tr>
                <tr>
                    <td>Fecha de Nacimiento</td>
                    <td th:text="${formulario.fechaNacimiento}"></td>
                </tr>
                <tr>
                    <td>Sexo</td>
                    <td th:text="${formulario.sexo}"></td>
                </tr>
                <tr>
                    <td>Dirección</td>
                    <td th:text="${formulario.direccion}"></td>
                </tr>
                <tr>
                    <td>Nivel de Estudios</td>
                    <td th:text="${formulario.nivelEstudios}"></td>
                </tr>
                <tr>
                    <td>Temas de Interés</td>
                    <td>
                        <ul>
                            <li th:each="tema : ${formulario.temasInteres}" th:text="${tema}"></li>
                        </ul>
                    </td>
                </tr>
            </table>
        </div>
        <div>
            <a href="/formulario">Volver atrás</a>
        </div>
    </body>
</html>
```

Funcionamiento:

Este es el formulario que aparecerá al inicio

Formulario de Registro

Datos Personales

Nombre [Nombre]
Primer Apellido [Primer Apellido]
Segundo Apellido [Segundo Apellido]
Correo Electrónico [Correo Electrónico]
Fecha de Nacimiento [dd/mm/aaaa]
Contraseña [Contraseña]
Sexo
- ○ Varón
- ○ Mujer

Dirección [Calle o Plaza, número y piso]

Nivel de estudios e intereses

Nivel de Estudios
- ○ Certificado escolar
- ○ Graduado en ESO
- ○ Bachiller - Formación profesional
- ○ Diplomado
- ○ Licenciado
- ○ Doctorado

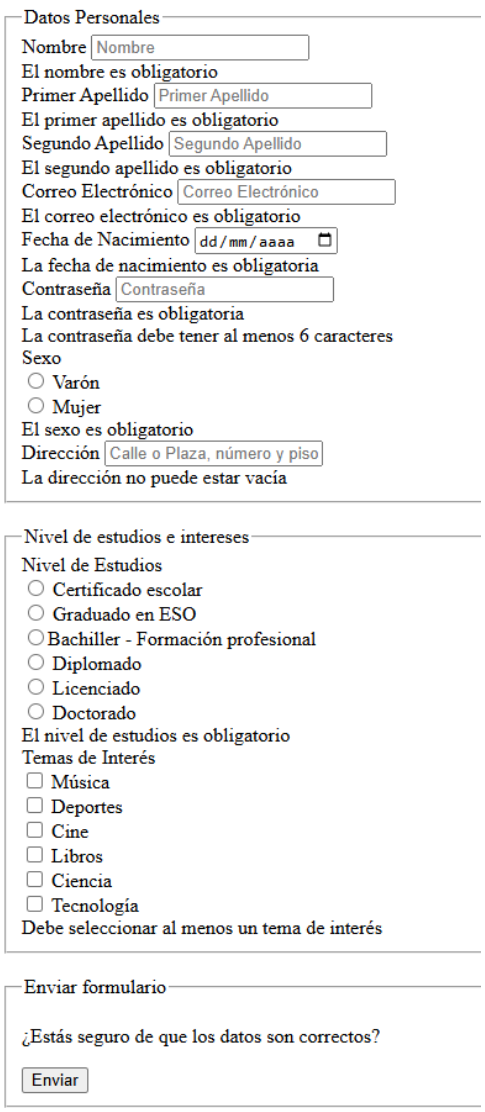
Temas de Interés
- ☐ Música
- ☐ Deportes
- ☐ Cine
- ☐ Libros
- ☐ Ciencia
- ☐ Tecnología

Enviar formulario

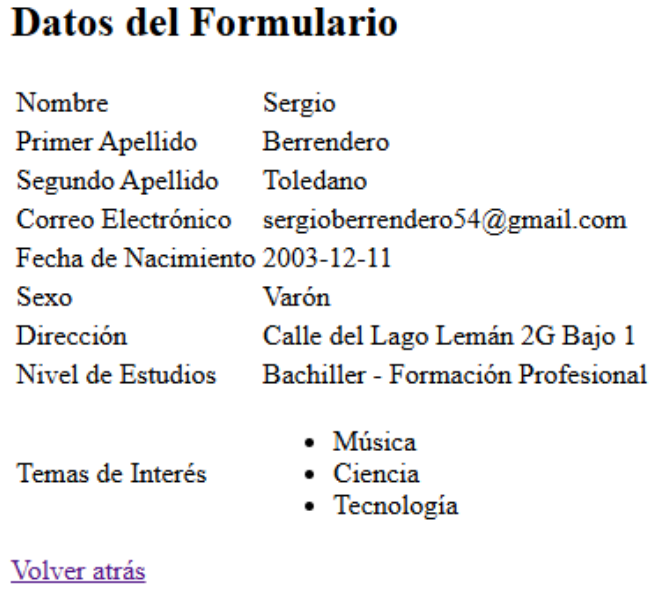
¿Estás seguro de que los datos son correctos?

[Enviar]

En caso de dejar datos sin rellenar mostrará que no se pueden dejar campos en blanco con los mensajes que se ven debajo de cada campo.



En caso de que esté todo correcto mostrará los datos introducidos por pantalla, en caso de dar al enlace nos volverá a mostrar el formulario vacío



Conclusión:

Esta práctica nos ha servido para romper el hielo con Spring y ver algunas de las posibilidades que ofrece este framework.

En esta práctica los ejercicios están relacionados con el manejo de formularios en una aplicación Spring con Thymeleaf. Cada ejercicio presenta en mayor o menor medida una forma de capturar datos de los usuarios, validarlos y luego mostrar los resultados de acuerdo a las reglas definidas en los controladores y clases de datos.