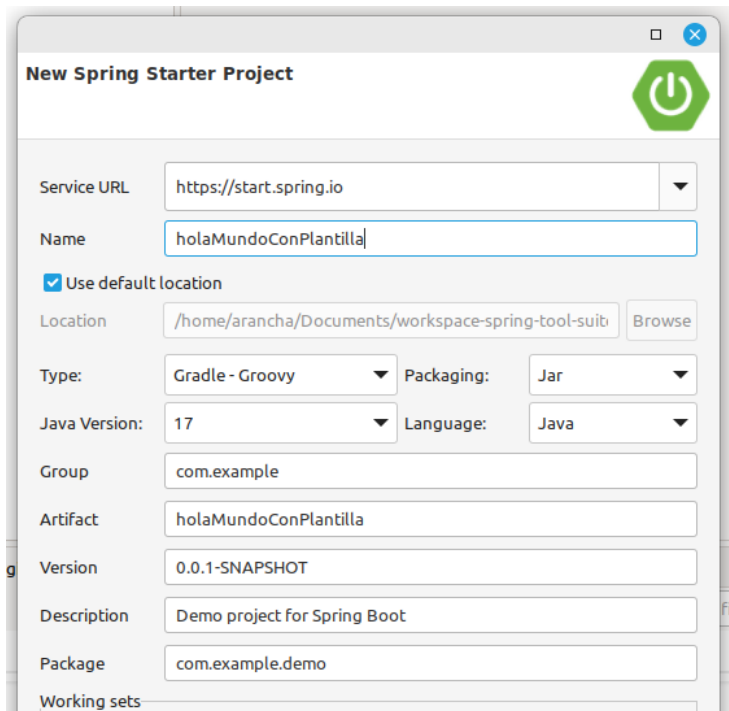


Tarea 3 – UD4

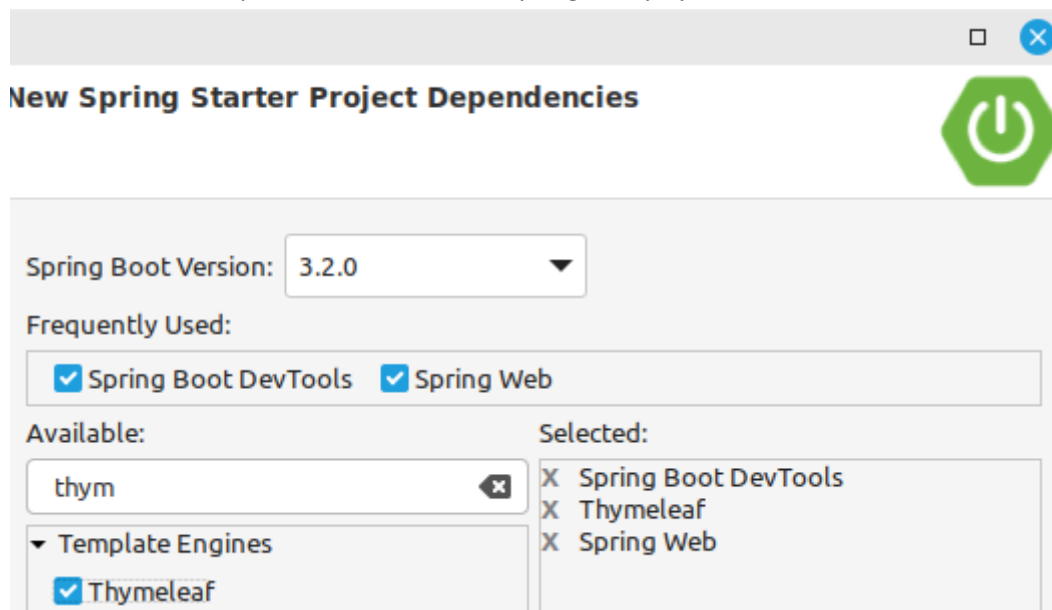
FORMULARIOS

APARTADO 1:

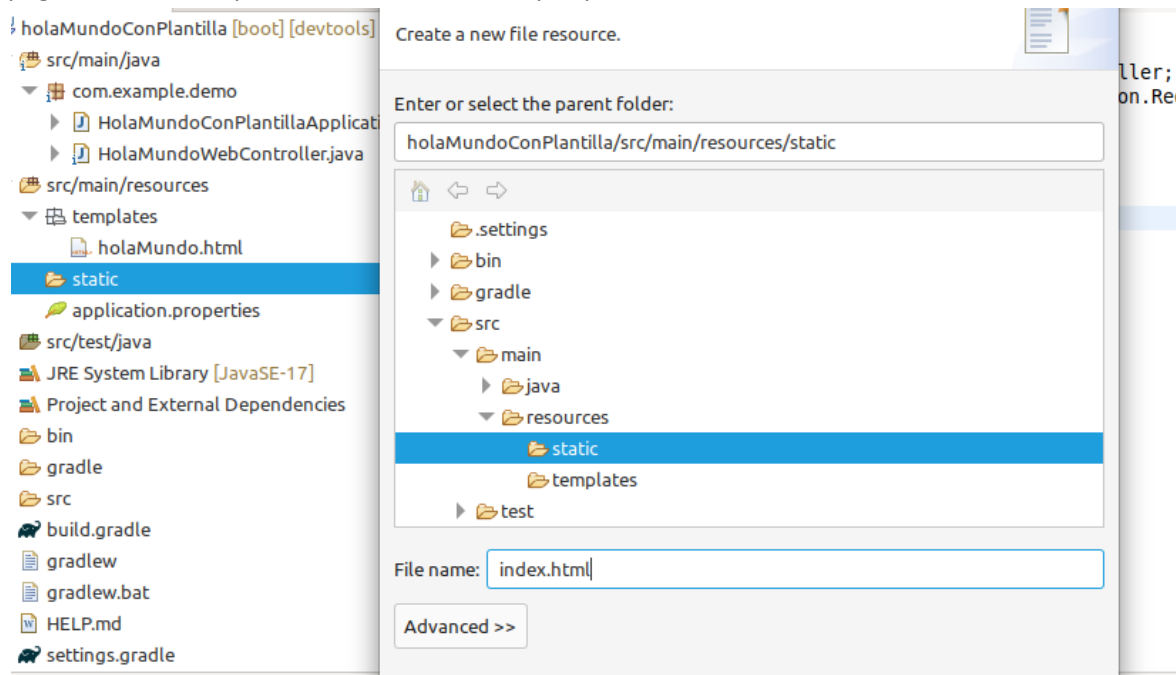
- **Tutorial 1: aplicación de Saludo con nombre por parámetro.**
 - Creo proyecto spring starter: holaMundoConPlantilla (tipo gradle-groovy, packaging en Jar y language en Java:



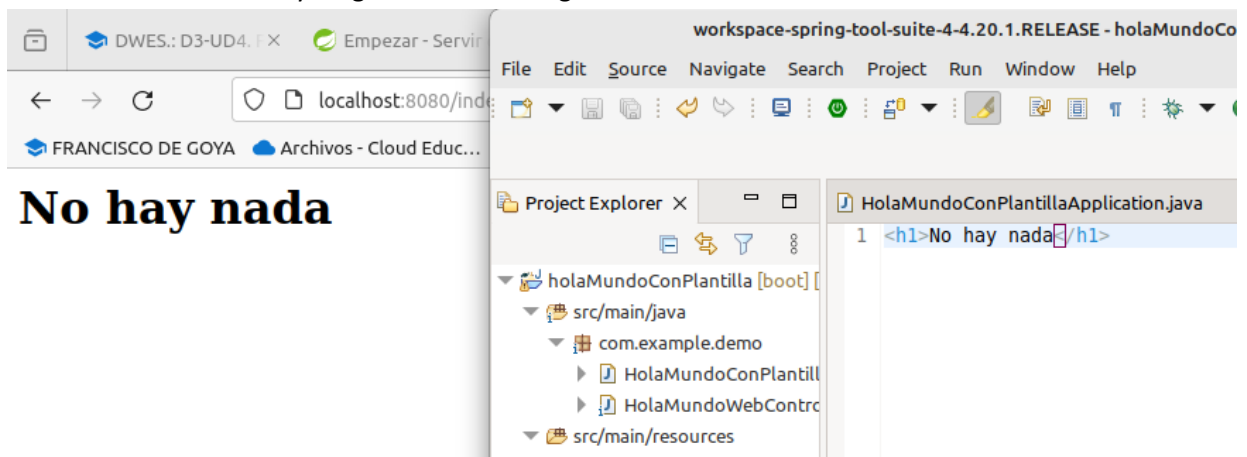
- Selecciono las dependencias DevTools, SpringWeb y Tymeleaf.



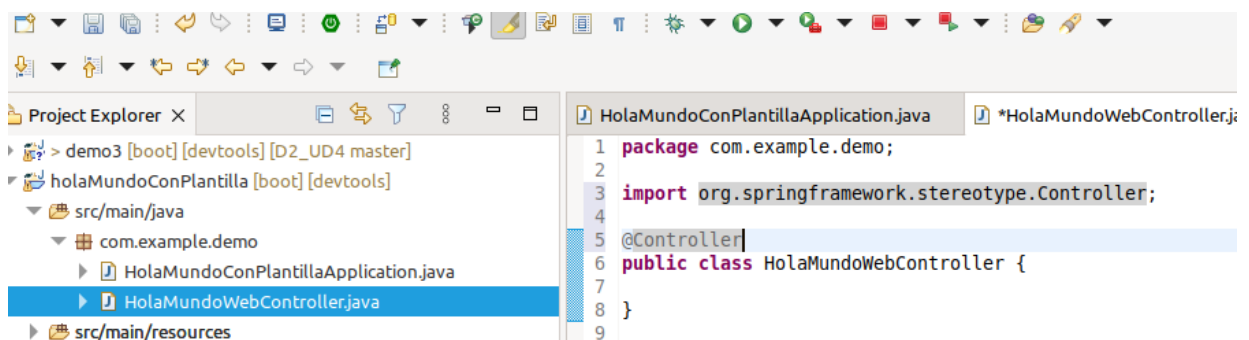
- Creo en carpeta static (dentro de resources) un index.html, la cual nos va a servir para asignar a nuestra aplicación una página predeterminada cuando acceda a la raíz, que servirá como una página estática sin procesamiento dinámico por parte del servidor.



Ponemos línea de texto y cargamos en el navegador:



- Creo clase **Controlador** HolaMundoController:



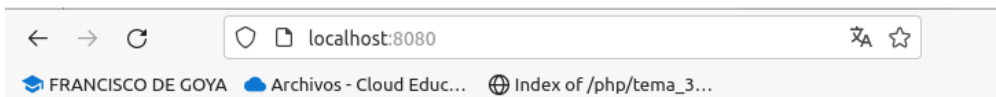
Y añado método procesaHolaMundo con anotación @RequestMapping a la raíz, que devuelve una plantilla holaMundo (la cuál correspondería a la vista):

```

1 package com.example.demo;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5
6 @Controller
7 public class HolaMundoWebController {
8
9     @RequestMapping("/")
10    public String procesaHolaMundo() {
11        return "holaMundo";
12    }

```

Ejecutamos y probamos en el navegador, da error porque no encuentra holaMundo (no lo hemos creado):



Whitelabel Error Page

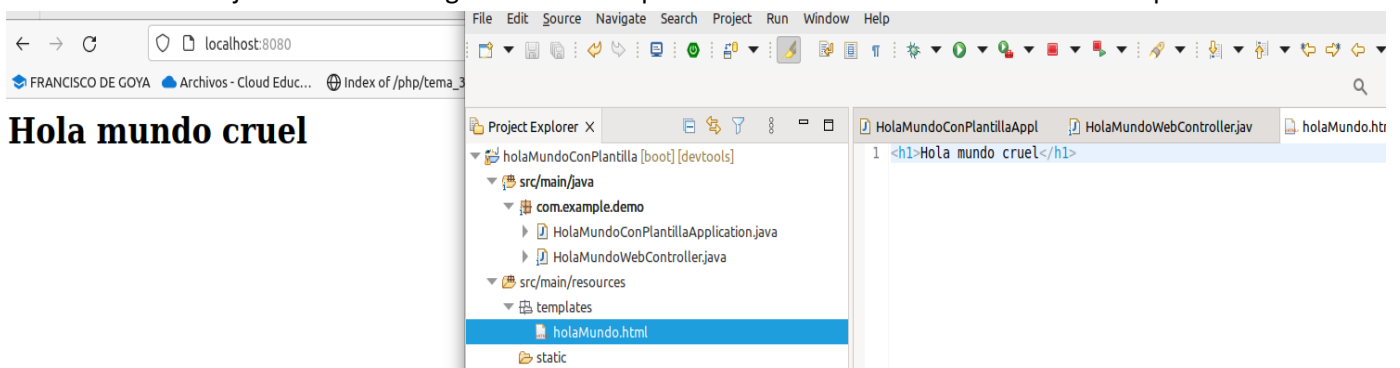
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Dec 19 12:11:10 CET 2023

There was an unexpected error (type=Internal Server Error, status=500).

Error resolving template [holaMundo], template might not exist or might not be accessible: configured Template Resolvers

- Creamos plantilla (VISTA) html en carpeta templates, con una línea de prueba en etiqueta h1, y volvemos a ejecutar en el navegador. Ahora sí que sale correctamente el texto de nuestra plantilla:



- Modificamos controlador para que reciba un parámetro nombre y usarlo en la plantilla para volcarlo en el navegador. Para que esto pase, se añade a un Model para hacerlo accesible a la plantilla mediante model.addAttribute. Y si no recibe un parámetro, coge valor por defecto "Mundo".

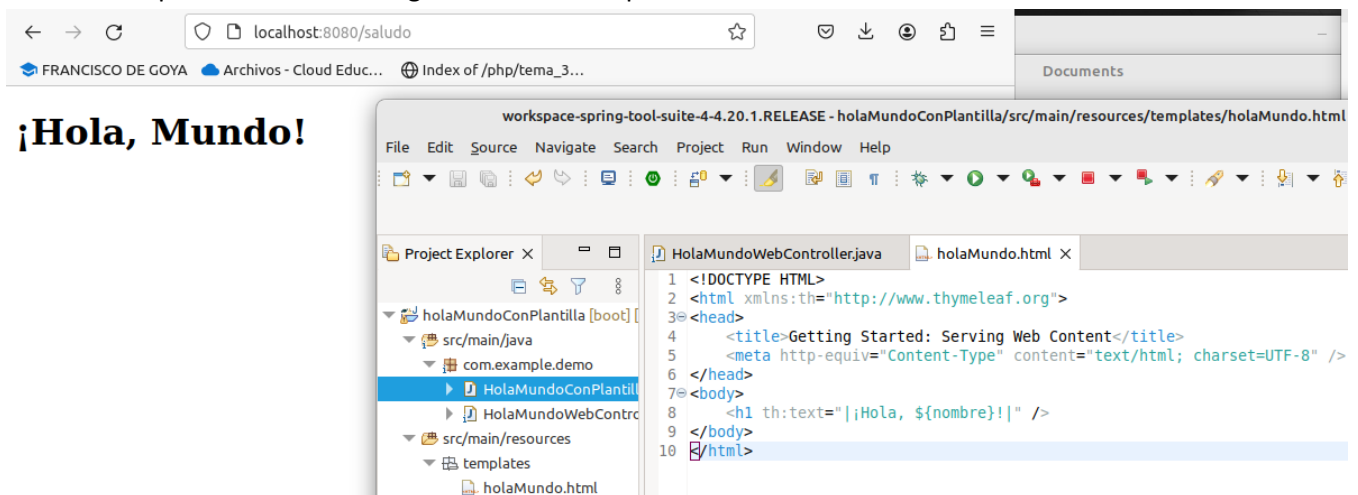
```

1 package com.example.demo;
2
3 import org.springframework.stereotype.Controller;
4
5 @Controller
6 public class HolaMundoWebController {
7
8     @RequestMapping("/saludo")
9     public String procesaHolaMundo(@RequestParam(name="nombre", required=false,
10        defaultValue="Mundo") String nombre, Model model) {
11
12        model.addAttribute("nombre", nombre);
13        return "holaMundo";
14    }
15 }

```

Modificamos la VISTA, plantilla holaMundo para que reciba el valor de nombre.

- Primero probamos en el navegador sin enviarle parámetro:



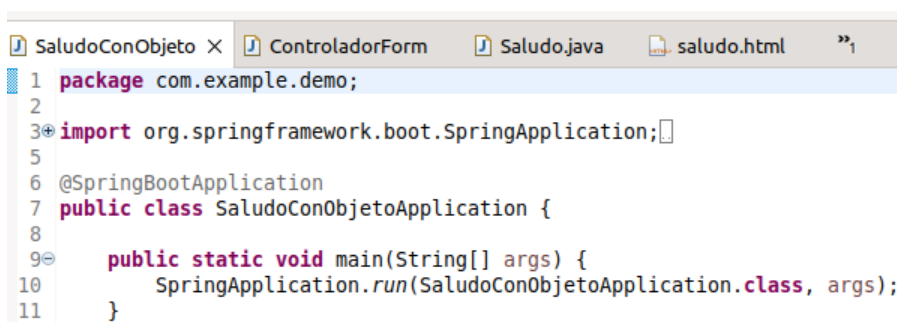
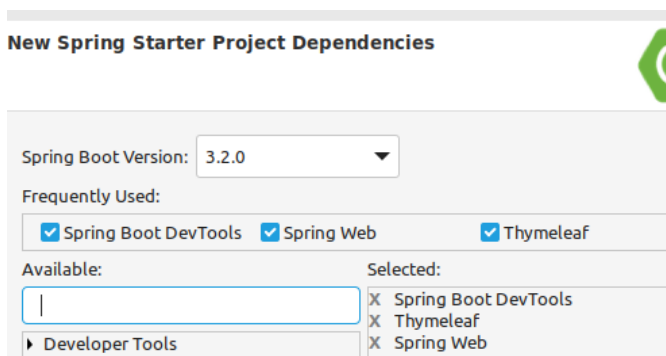
- Y ahora enviamos un parámetro nombre con valor Arancha:



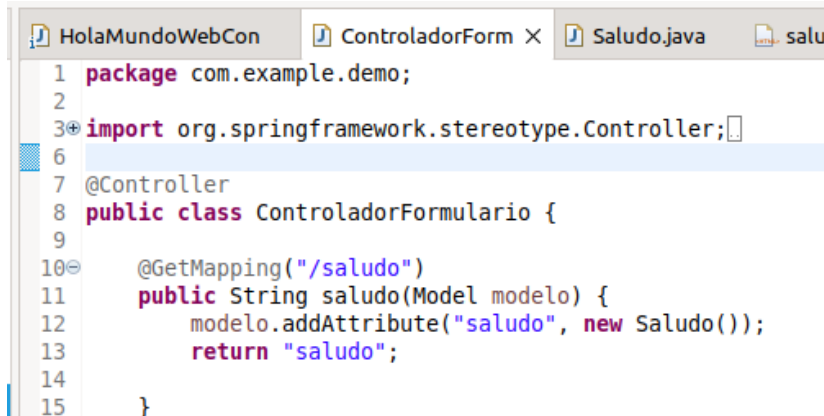
¡Hola, Arancha!

• Tutorial 2: aplicación formulario de datos ID y mensaje sin parámetros.

- Creo proyecto spring starter llamado SaludoConObjeto con las 3 dependencias vistas en el ej anterior:



- **Creo Controlador** llamado ControladorFormulario. Añado anotación GetMapping /Saludo y método que recibe un Model y le añadimos el atributo saludo de tipo Objeto Saludo, y devuelve plantilla VISTA saludo:

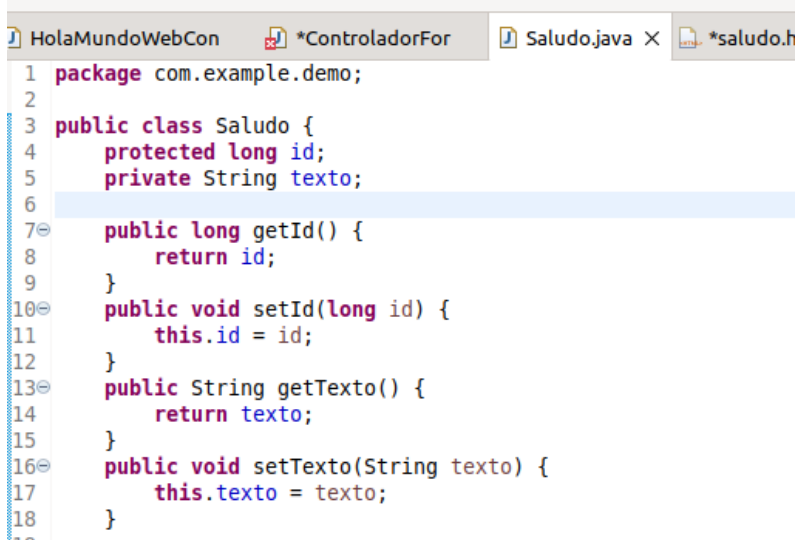


```

1 package com.example.demo;
2
3 import org.springframework.stereotype.Controller;
4
5
6
7 @Controller
8 public class ControladorFormulario {
9
10     @GetMapping("/saludo")
11     public String saludo(Model modelo) {
12         modelo.addAttribute("saludo", new Saludo());
13         return "saludo";
14     }
15 }

```

- **Creo MODELO**, objeto, llamado Saludo con los atributos id y texto, y añado sus getters y setters:

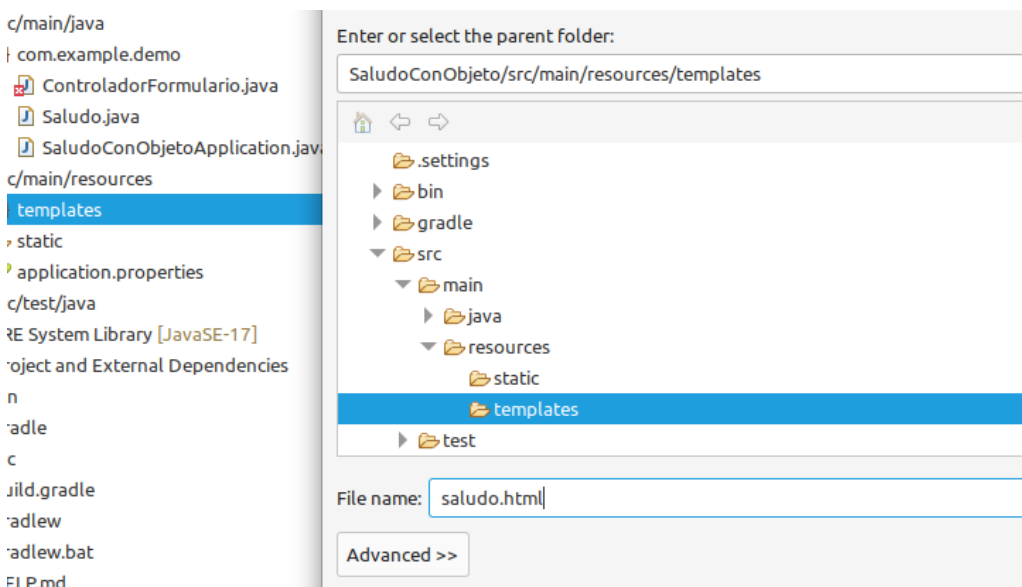


```

1 package com.example.demo;
2
3 public class Saludo {
4     protected long id;
5     private String texto;
6
7     public long getId() {
8         return id;
9     }
10    public void setId(long id) {
11        this.id = id;
12    }
13    public String getTexto() {
14        return texto;
15    }
16    public void setTexto(String texto) {
17        this.texto = texto;
18    }
19 }

```

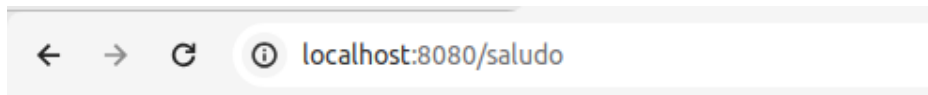
- **Creo VISTA** plantilla html llamada saludo.html (la que tiene que devolver según el getMapping del controlador:



Copio plantilla del tutorial, adaptándola a mis datos

```
ControladorFormulario.java Saludo.java SaludoConObjetoApplication.java saludo.html x
1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4   <title>Getting Started: Handling Form Submission</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8   <h1>Form</h1>
9   <form action="#" th:action="@{/saludo}" th:object="${saludo}" method="post">
10    <p>Id: <input type="text" th:field="*{id}" /></p>
11    <p>Mensaje: <input type="text" th:field="*{texto}" /></p>
12    <p><input type="submit" value="Submit" /> <input type="reset" value="Reset" /></p>
13  </form>
14 </body>
15 </html>
```

Y ejecutamos en el navegador poniendo saludo (el getMapping):

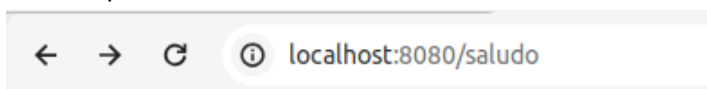


Form

Id:

Mensaje:

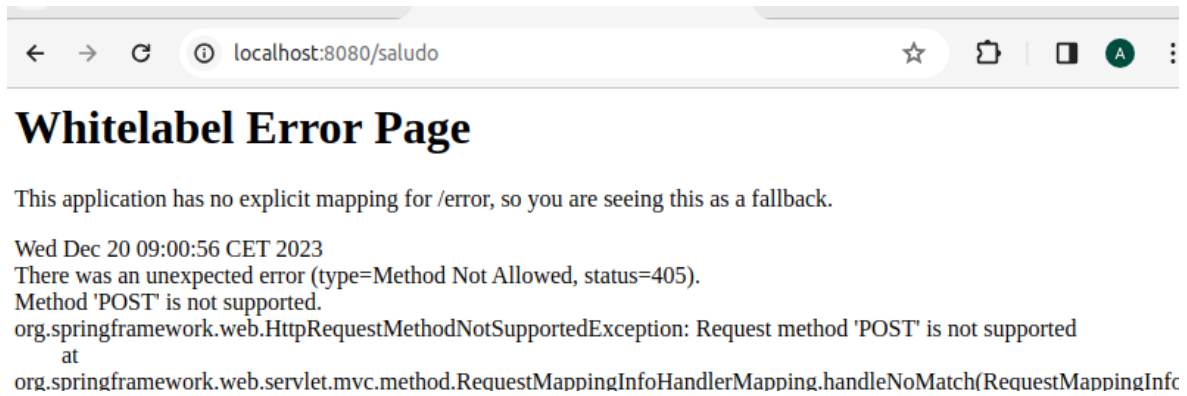
Ponemos datos y al enviar da error porque no hay método POST y según la plantilla, este formulario se envía por POST:



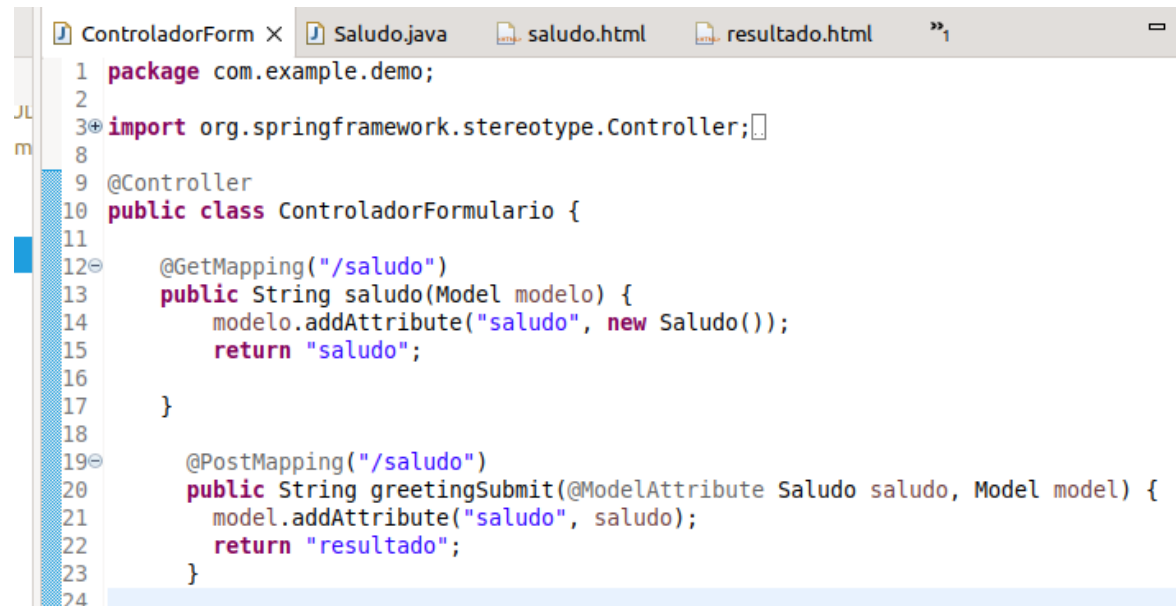
Form

Id:

Mensaje:

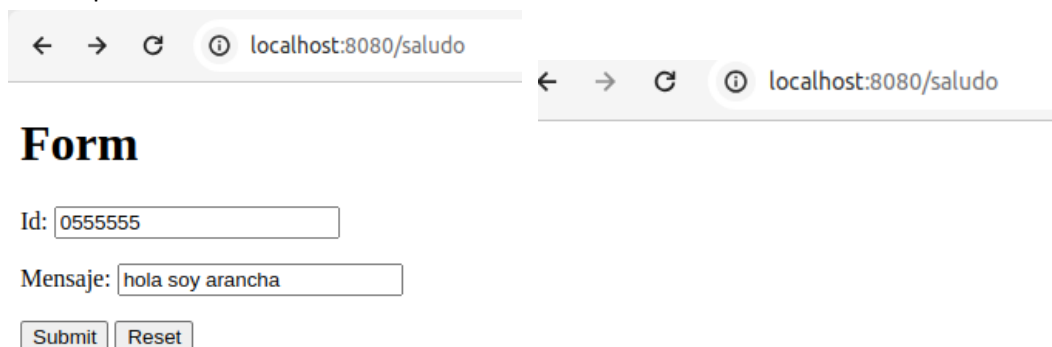


Modifico Controlador: pongo anotación con Post, que recibe objeto de tipo saludo y de tipo Model, que en model.addAttribute, añadimos el valor de saludo al model para enviarlo a la vista/plantilla. Y devuelve plantilla/vista llamada resultado:



Ahora cargo página de nuevo en el navegador y envío datos, ya no da error, el formulario se envía a sí mismo y aparece nuevamente. Ahora funciona porque en la plantilla, el th:action="@{/saludo}" dirige el formulario a /saludo por POST.

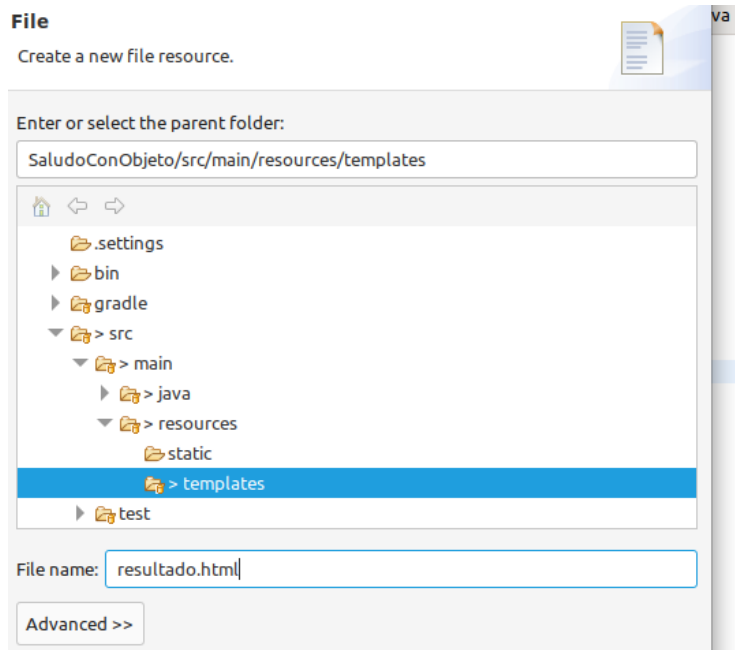
Envío los datos y no hay error, se envía el formulario aunque no aparece nada porque aún no hemos creado plantilla resultado:



Esto cubre el modelo, vista y controlador para presentar el formulario. Ahora podemos revisar el proceso de presentación del formulario. El formulario se ejecuta en /saludo mediante el uso de una llamada POST.

El método del PostMapping recibe el objeto Saludo que es a @ModelAttribute, por lo tanto está unido al contenido de la plantilla, vista.

Ahora creo plantilla de resultado, adaptándola a mis datos. Pongo saludo.id y saludo.texto para captar los valores de mis atributos.



```

1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4   <title>Getting Started: Handling Form Submission</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8   <h1>Result</h1>
9   <p th:text="'id: ' + ${saludo.id}" />
10  <p th:text="'texto: ' + ${saludo.texto}" />
11  <a href="/saludo">Enviar otro mensaje.</a>
12 </body>
13 </html>

```

Podemos usar una sola plantilla para los dos propósitos, pero es más claro usar dos vistas separadas para representar el formulario y mostrar los datos.

Ejecutamos nuevamente en navegador:



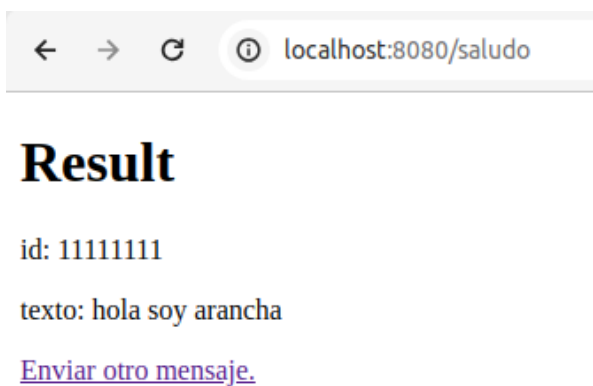
← → ↻ ⓘ localhost:8080/saludo

Form

Id:

Mensaje:

Envío datos y ahora aparece la página con los datos enviados, correspondiente a la plantilla html resultado:



← → ↻ ⓘ localhost:8080/saludo

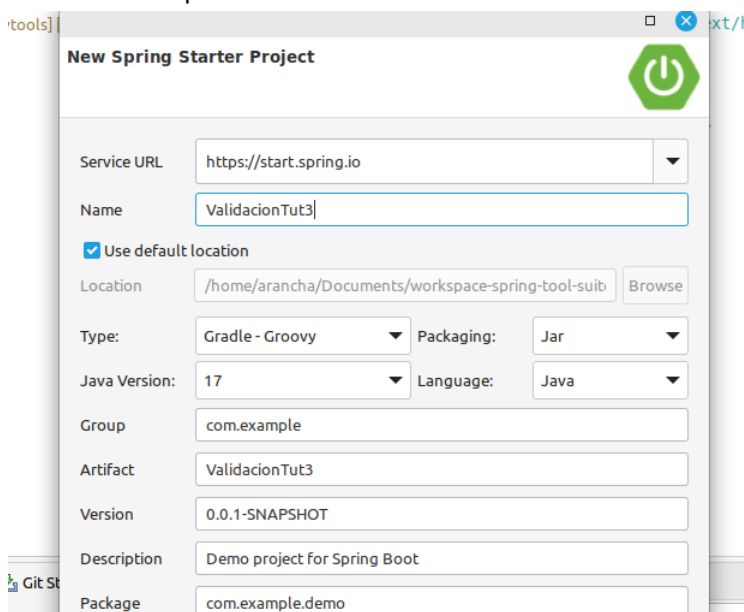
Result

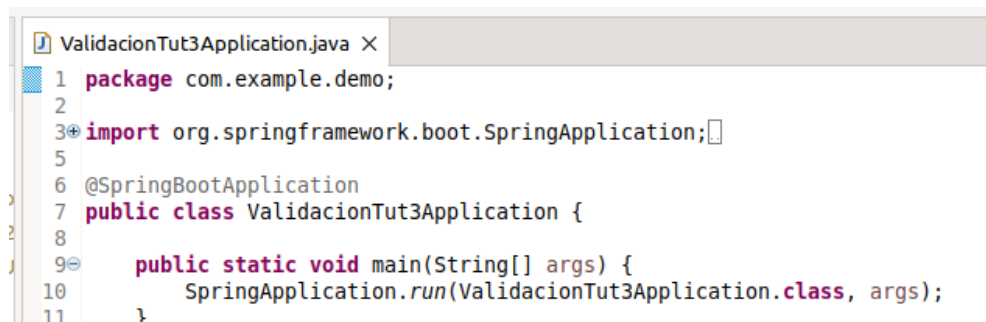
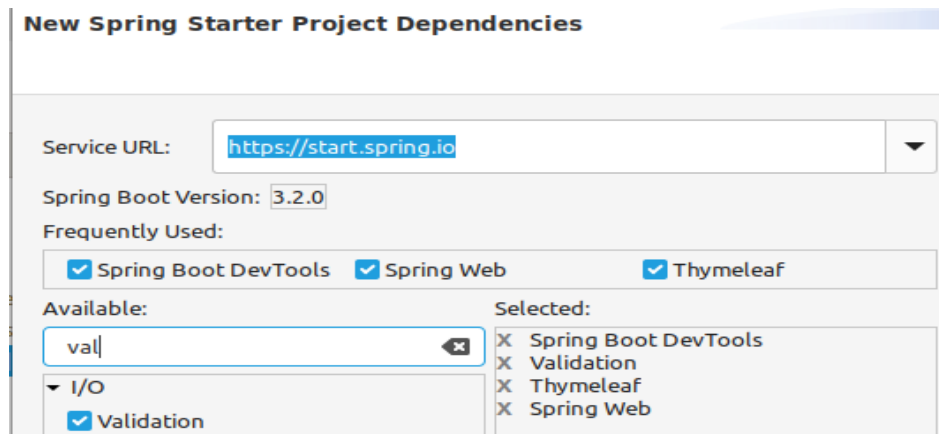
id: 11111111

texto: hola soy arancha

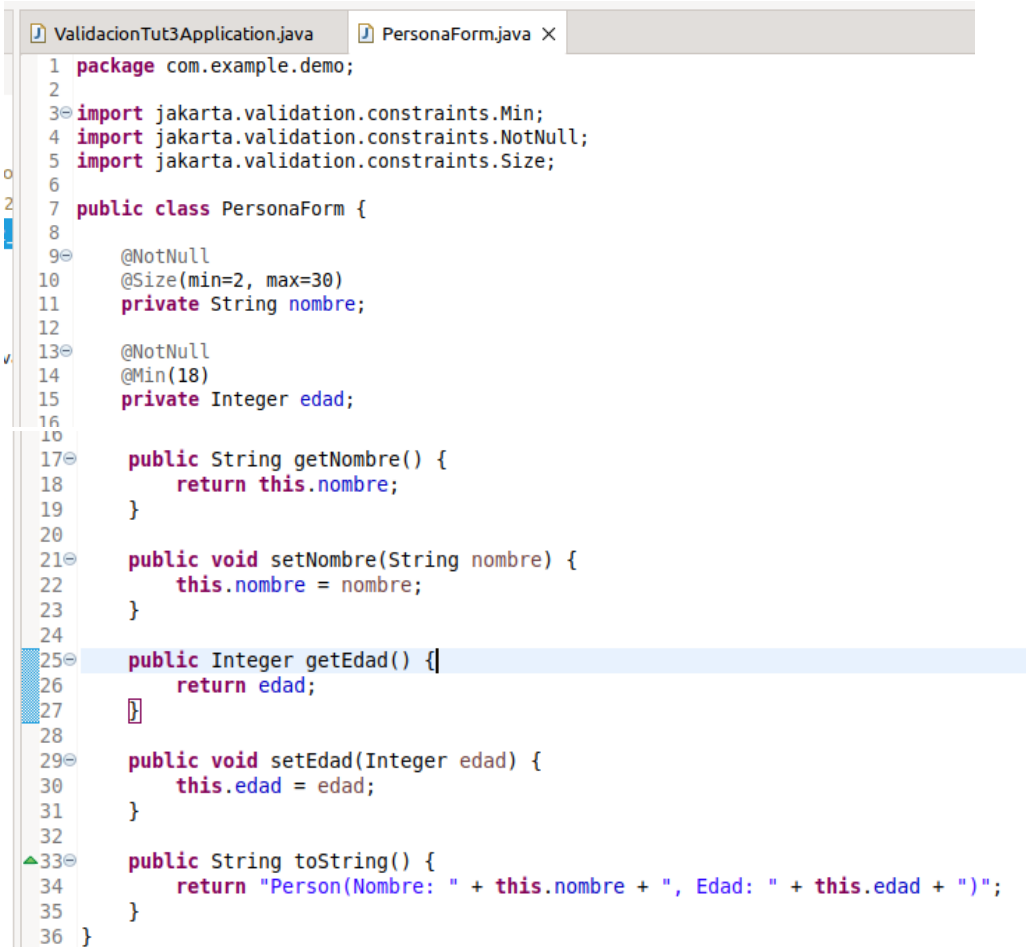
[Enviar otro mensaje.](#)

-
- **Tutorial 3: aplicación formulario de datos nombre y edad con validación.**
 - Creo proyecto spring starter llamado ValidacionTut3 con las 3 dependencias vistas y además añadimos la dependencia validation:



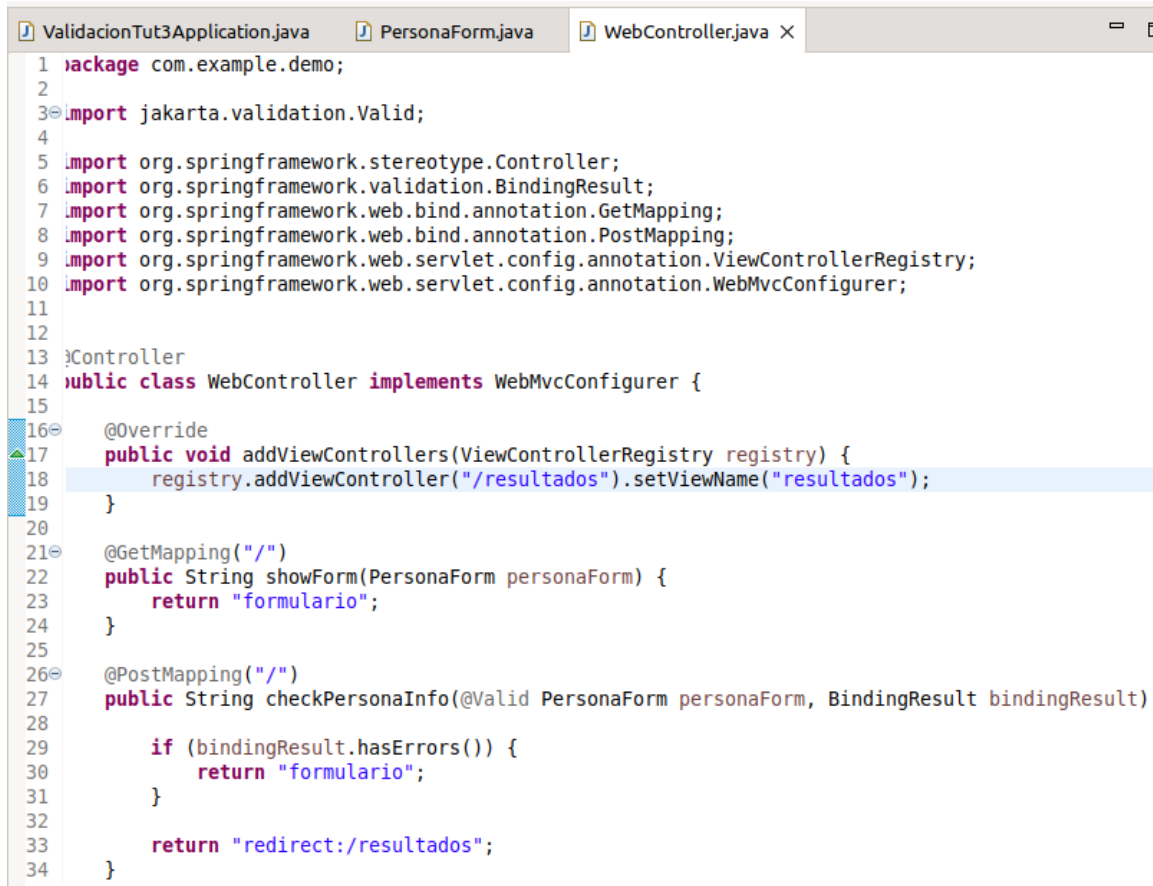


- Creamos **MODELO** PersonaForm con atributos nombre y edad y los getters y setters:



Describimos las siguientes anotaciones:

- **@NotNull**: hace que no permita valor nulo, que es lo que Spring genera si el campo está vacío.
- **@Size**: permite string entre el min y el max de caracteres de longitud.
- **@Min**: no permite valores inferiores a ese número.
- Creo **CONTROLADOR** llamado WebController:



```

1 package com.example.demo;
2
3 import jakarta.validation.Valid;
4
5 import org.springframework.stereotype.Controller;
6 import org.springframework.validation.BindingResult;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
10 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
11
12
13 @Controller
14 public class WebController implements WebMvcConfigurer {
15
16     @Override
17     public void addViewControllers(ViewControllerRegistry registry) {
18         registry.addViewController("/resultados").setViewName("resultados");
19     }
20
21     @GetMapping("/")
22     public String showForm(PersonaForm personaForm) {
23         return "formulario";
24     }
25
26     @PostMapping("/")
27     public String checkPersonaInfo(@Valid PersonaForm personaForm, BindingResult bindingResult)
28     {
29         if (bindingResult.hasErrors()) {
30             return "formulario";
31         }
32
33         return "redirect:/resultados";
34     }
35 }

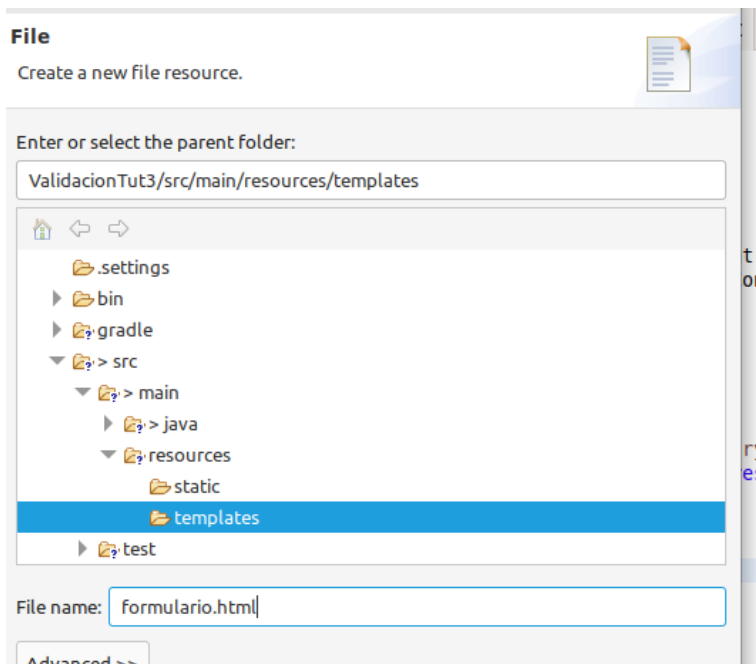
```

El método showForm devuelva la plantilla formulario, que recibe un objeto PersonaForm para que el html pueda asociar los atributos, y en el método checkPersonaInfo vemos cosas nuevas, acepta estos dos argumentos:

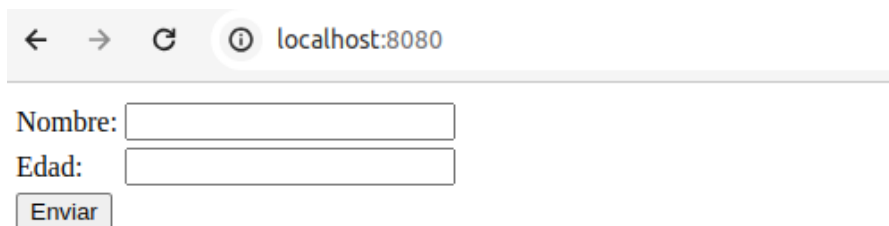
- **personForm**: objeto marcado con **@Valid** para recoger los atributos rellenados en el formulario.
- **bindingResult**: objeto para que pueda probar y recuperar errores de validación.

Si hay errores, devuelve nuevamente la plantilla del formulario, mostrando los atributos de error, pero si los atributos son válidos, nos redirige a la plantilla resultados con redirect:/.

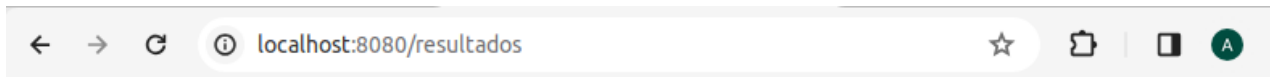
- Creo **VISTA**, plantilla hml formulario.html, copio del tutorial adaptando los nombres a mis atributos:
Observamos que los mensajes de error, estarán en `filesd.hasErrors(---)`.



Aunque nos falte hacer la plantilla resultados, vamos a ejecutarlo en el navegador para ir viendo que funciona:



Y al enviar los datos, obviamente tenemos error al no existir resultados:



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Dec 20 10:21:55 CET 2023

There was an unexpected error (type=Internal Server Error, status=500).

Error resolving template [resultados], template might not exist or might not be accessible by any of the configured Template Resolvers

org.thymeleaf.exceptions.TemplateInputException: Error resolving template [resultados], template might not exist or might not be accessible by any of the configured Template Resolvers

at org.thymeleaf.engine.TemplateManager.resolveTemplate(TemplateManager.java:869)

at org.thymeleaf.engine.TemplateManager.parseAndProcess(TemplateManager.java:607)

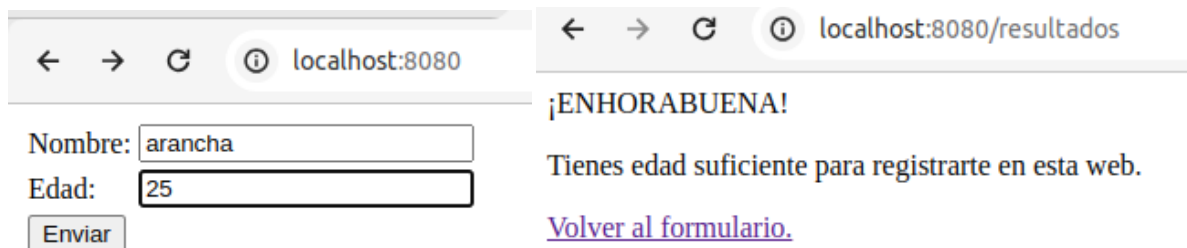
- Creamos **VISTA** html resultados (también en la carpeta templates). Creo también un enlace para volver al formulario, volver a atrás:

```

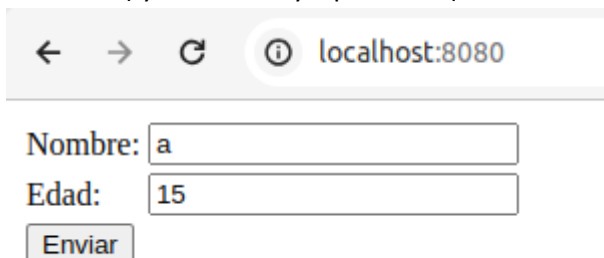
1 <html>
2   <body>
3       <p>¡ENHORABUENA!</p>
4       <p>Tienes edad suficiente para registrarte en esta web.</p>
5       <a href="/">Volver al formulario.</a>
6   </body>
7 </html>

```

Y ejecutamos en el navegador, vemos que se redirige correctamente a la página resultados, con el contenido de esa plantilla:



Ahora voy a provocar errores. En el campo nombre voy a poner una sola letra (el mínimo son 2 caracteres) y en edad voy a poner 15 (el mínimo es 18):

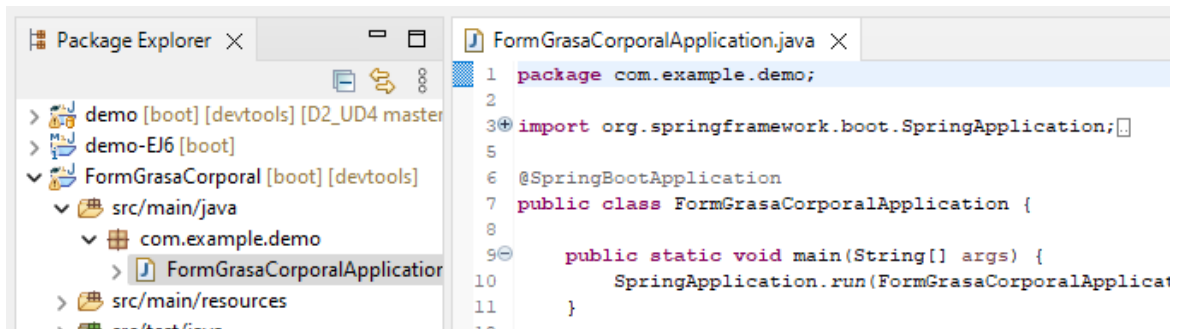


Podemos ver los mensajes de error. Estos mensajes son estándar y están ya creados, predefinidos, pues estos mensajes no los hemos creado en ninguna parte de nuestro código ni en las plantillas html. Muy curioso que además salgan en español:

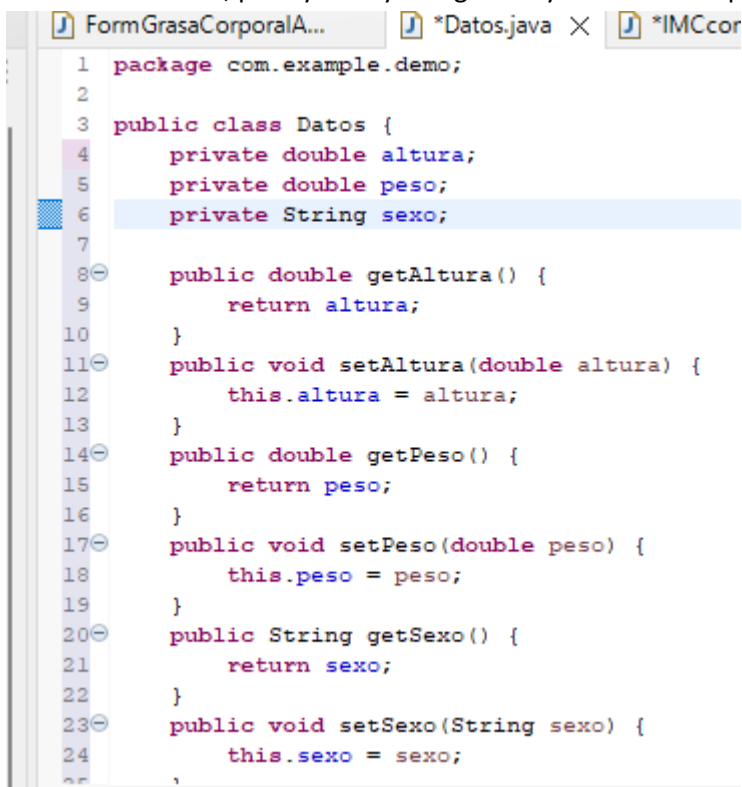
Probamos poniendo solo la edad errónea:

APARTADO 2: aplicación con formulario de entrada de datos que permita calcular nuestro índice de grasa corporal, a partir del valor proporcionado nos tendrá que indicar que clasificación nos encontramos.

- Creo proyecto spring con las dependencias vistas anteriormente:



- **Ahora creo clase objeto/modelo** llamada datos que represente los datos del formulario, con los atributos de altura, peso y sexo y sus getters y setters correspondientes:



- **Creo clase controlador** para que maneje las solicitudes del formulario y los cálculos, llamada IMCcontroller, y ahí creo las anotaciones correspondientes para hacer los cálculos.

Fijándome en los tutoriales anteriores, voy a crear una anotación GetMapping que devuelva el formulario y una anotación PostMapping que devuelva el resultado, ambas siendo plantillas html.

- ❖ @GetMapping(/calcularIMC): recibe un objeto tipo Model, en este caso será datos. Y con model.addAttribute, agrega el valor del atributo datos de tipo objeto Datos al modelo.
- ❖ @PostMapping(/calcularIMC): recibe dos parámetros:
 - @ModelAttribute Datos datos: indica que se debe asociar los datos del formulario a un objeto de tipo Datos.
 - Model model: objeto Model para pasar datos entre controlador y vista.
 - Double imc: creo elemento donde hago el cálculo, obteniendo la altura y peso con su método get.

- String clasificación: creo elemento donde llamo a la función obtenerClasificación, que recibe los parámetros imc y sexo, necesarios en dicha función y se obtendrá el valor return correspondiente.
- Model.addAttribute: esto sirve para agregar el dato imc o clasificación al modelo con el mismo nombre, para que la vista (formulario) acceda a este valor.



```

1 package com.example.demo;
2
3 import org.springframework.ui.Model;
4
5
6
7
8 public class IMCcontroller {
9
10     @GetMapping("/calcularIMC")
11     public String mostrarFormulario(Model model) {
12         //agrego atributo/valor de datos al modelo:
13         model.addAttribute("datos", new Datos());
14         return "formulario";
15     }
16
17     @PostMapping("/calcularIMC")
18     public String calcularBMI(@ModelAttribute Datos datos, Model model) {
19         // Realizar el cálculo del IMC:
20         double imc = datos.getPeso() / (datos.getAltura() * datos.getAltura());
21         //Determino clasificación llamando a la función:
22         String clasificacion = obtenerClasificacion(imc, datos.getSexo());
23
24         // Agregar resultados al modelo
25         model.addAttribute("imc", imc);
26         model.addAttribute("clasificacion", clasificacion);
27
28         return "resultado";
29     }
30

```

- Función obtenerClasificacion: recibe parámetros imc y sexo, primero compruebo con if si el sexo es mujer u hombre, con equalsIgnoreCase para que no diferencia entre mayúsculas y minúsculas, ya que cada sexo tiene una clasificación diferente, que dentro de su if voy comparando con más ifs, y según los datos devuelve un mensaje/clasificación:


```

private String obtenerClasificacion(double imc, String sexo) {
    if ("mujer".equalsIgnoreCase(sexo)) {
        if (imc > 9 && imc < 14) {
            return "Grasa esencial";
        } else if (imc > 13 && imc < 21) {
            return "Atletas";
        } else if (imc > 20 && imc < 25) {
            return "Fitness";
        } else if (imc > 24 && imc < 32) {
            return "Aceptable";
        } else if (imc > 32) {
            return "Sobrepeso";
        }
    }
}

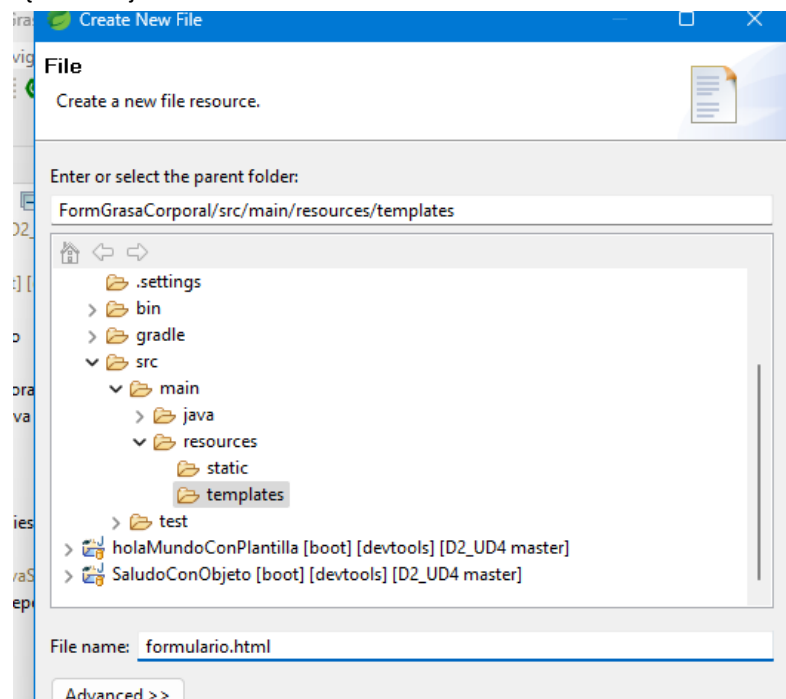
if ("hombre".equalsIgnoreCase(sexo)) {
    if (imc > 1 && imc < 6) {
        return "Grasa esencial";
    } else if (imc > 5 && imc < 14) {
        return "Atletas";
    } else if (imc > 13 && imc < 18) {
        return "Fitness";
    } else if (imc > 17 && imc < 25) {
        return "Aceptable";
    } else if (imc > 25) {
        return "Sobrepeso";
    }
}
return "No disponible";
}
}

```

- **Ahora creo las plantillas HTML**, correspondiente a **VISTA**, en carpeta templates.

Primero formulario que devuelve el GetMapping de la clase controlado. Para que luego en controlador lea los datos, en los campos correspondientes, pongo el nombre del atributo de esta manera

*{nombre}:



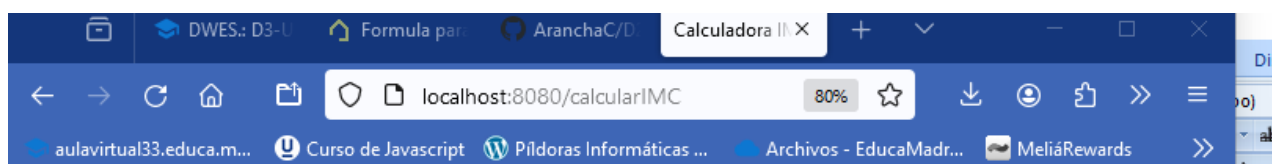


```

1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4   <title>Calculadora IMC</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8   <h1>Formulario cálculo Indice Masa Corporal (IMC)</h1>
9   <h3>Introduce tus datos:</h3>
10  <form action="#" th:action="@{/calcularIMC}" th:object="${datos}" method="post">
11    <p>Sexo (mujer / hombre): <input type="text" th:field="*{sexo}" /></p>
12    <p>Altura: <input type="text" th:field="*{altura}" /></p>
13    <p>Peso: <input type="text" th:field="*{peso}" /></p>
14    <p><input type="submit" value="Calcular IMC" /> <input type="reset" value="Reset" /></p>
15  </form>
16 </body>
17 </html>

```

Pruebo a ejecutar la aplicación, no hay errores, y hago la prueba en el navegador y vemos que aparece el formulario que acabo de crear:



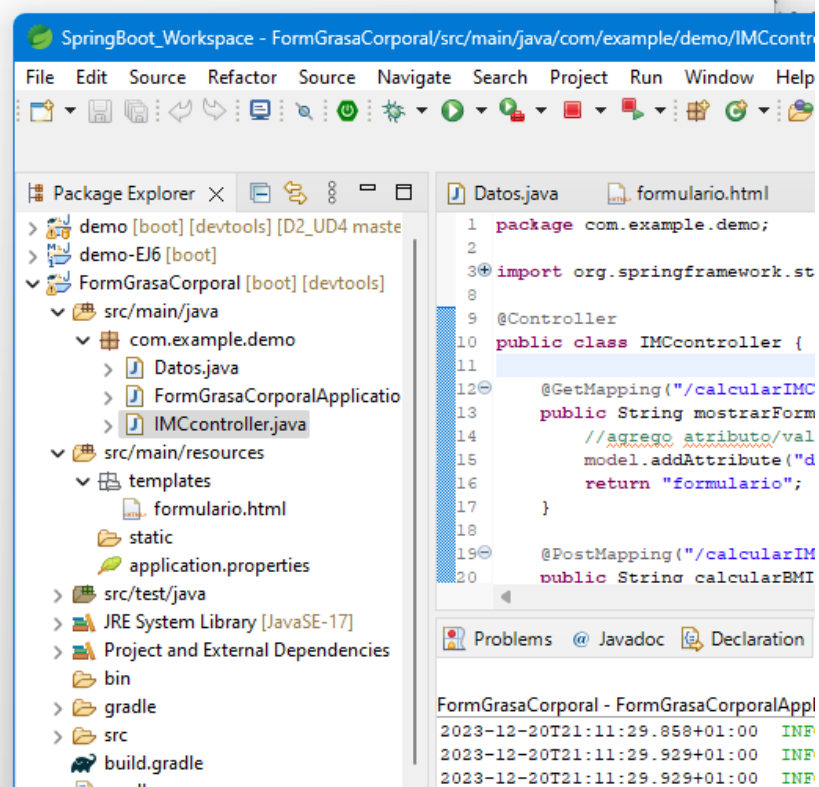
Formulario cálculo Indice Masa Corporal (IMC)

Introduce tus datos:

Sexo (mujer / hombre):

Altura:

Peso:



- Ahora **creo la plantilla (VISTA)** resultado, que devuelve el PostMapping, recibiendo todos los datos introducidos en formulario:

```

1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4     <title>Calculadora IMC</title>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8     <h1>Formulario cálculo Indice Masa Corporal (IMC)</h1>
9     <h3>Resultado</h3>
10    <p th:text="'IMC: ' + ${imc}" />
11    <p th:text="'Clasificación: ' + ${clasificacion}" />
12    <a href="/calcularIMC">Enviar nuevos datos.</a>
13 </body>
14 </html>
15

```

Ejecuto y pruebo en el navegador introduciendo datos:



Formulario cálculo Indice Masa Corporal (IMC)

Introduce tus datos:

Sexo (mujer / hombre):

Altura:

Peso:

Y damos a calcular IMC, botón de enviar y todo funciona correctamente:



Formulario cálculo Indice Masa Corporal (IMC)

Resultado

IMC: 23.11111111111111

Clasificación: Fitness

[Enviar nuevos datos.](#)

Damos al enlace para volver a enviar nuevos datos y probamos con otros:

Formulario cálculo Índice Masa Corporal (IMC)

Introduce tus datos:

Sexo (mujer / hombre):

Altura:

Peso:



Formulario cálculo Índice Masa Corporal (IMC)

Resultado

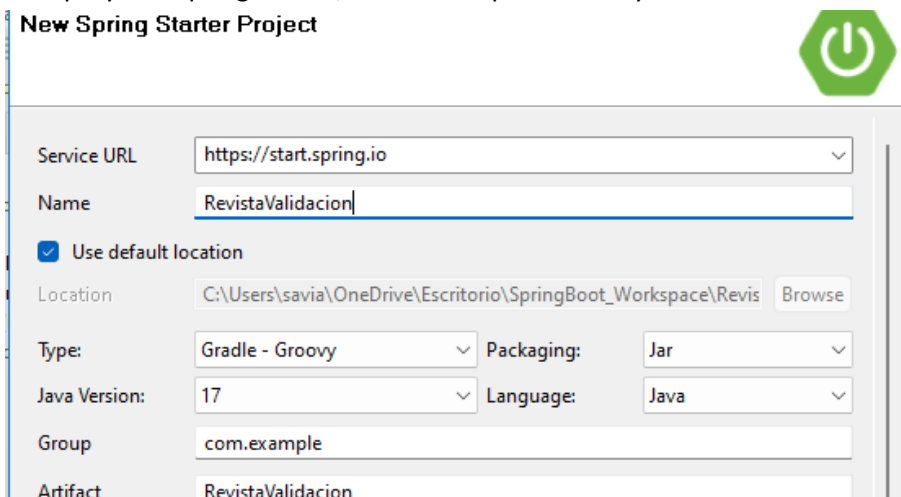
IMC: 28.946124763705104

Clasificación: Sobrepeso

[Enviar nuevos datos.](#)

APARTADO 3: formulario de registro de esta revista online, en él se recogerá la información personal de usuario. Cuando el formulario sea enviado se deberán mostrar los datos que el usuario haya introducido.

- Creo proyecto spring starter, con las 4 dependencias ya vistas:



New Spring Starter Project Dependencies



Spring Boot Version: 3.2.1

Frequently Used:

- ☒ Spring Boot DevTools
 ☒ Spring Web
 ☒ Thymeleaf
 ☒ Validation

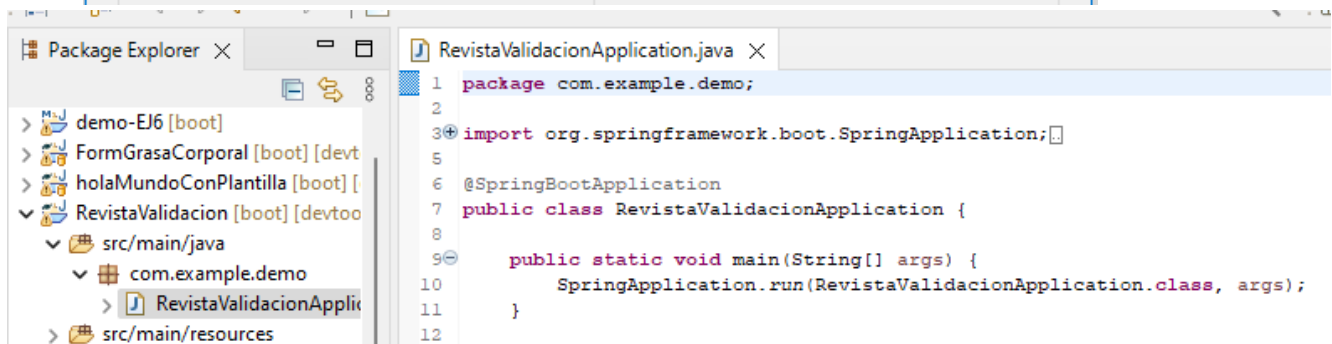
Available:

Type to search dependencies

- ▶ Developer Tools
- ▶ Google Cloud

Selected:

- X Spring Boot DevTools
- X Validation
- X Thymeleaf
- X Spring Web



- Creo **MODELO** objeto Persona con los atributos: nombre, apellido1, apellido2, anyoNac, email, contraseña, sexo, dirección, estudios, array temas.

Creo los getters y setters.

Añado también las siguientes anotaciones de validación:

- @NotEmpty: campo obligatorio para nombre, apellido1 y temas.
- @NotNull: campo obligatorio (no nulo) para anyoNac, contraseña y dirección.
- @Max: valor máximo para anyoNac.
- @Pattern: valor debe cumplir con el patrón para el email.
- @Size: cadena con mínimo y máximo de caracteres indicados para contraseña, dirección y temas. En el caso de temas, sería el número de elementos del array, en este caso tiene que seleccionar entre 2 y 4 opciones.

```

11
12 public class Persona {
13
14     @NotEmpty
15     private String nombre;
16
17     @NotEmpty
18     private String apellido1;
19     private String apellido2;
20
21     @NotNull
22     @Max(2005)
23     private int anyoNac;
24
25     @Pattern(regexp = ".*@.*\\.\\.(com|es)$")
26     private String email;
27
28     @NotNull
29     @Size(min=5, max=10)
30     private String contrasena;
31     private String sexo;
32
33     @NotNull
34     @Size(min=8)
35     private String direccion;
36     private String estudios;
37
38     @NotEmpty
39     @Size(min=2, max=4)
40     private String[] temas;
41

```

- Creo **CONTROLADOR** y método `mostrarFormRegistro` con `GetMapping /registroRevista` que devuelve vista/plantilla `formRegistro`. También `addViewControllers`, fijándome en el ejemplo del tutorial 3, poniendo en `vistaControlador` plantilla `registroOK`:

```

1 package com.example.demo;
2
3 import org.springframework.stereotype.Controller;
4
5
6
7
8 @Controller
9 public class RevistaController implements WebMvcConfigurer {
10
11     @Override
12     public void addViewControllers(ViewControllerRegistry registry) {
13         registry.addViewController("/registroOK").setViewName("registroOK");
14     }
15
16     @GetMapping("/registroRevista")
17     public String mostrarFormRegistro(Persona persona) {
18         return "formRegistro";
19     }
20
21 }
22

```

Añado método registroCorrecto con PostMapping /registroRevista, con parámetro @Valid Persona persona, para recoger los atributos del formulario y bindingResult para los errores de validación.

```
@PostMapping("/registroRevista")
public String registroCorrecto(@Valid Persona persona,
                               BindingResult bindingResult) {

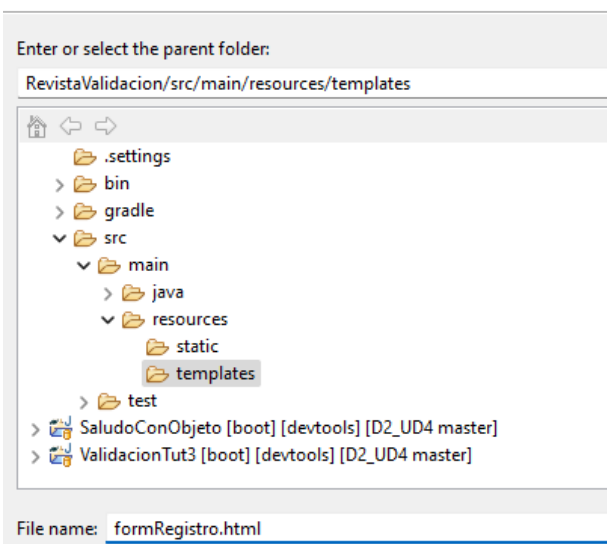
    if (bindingResult.hasErrors()) {
        return "formRegistro";
    }

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

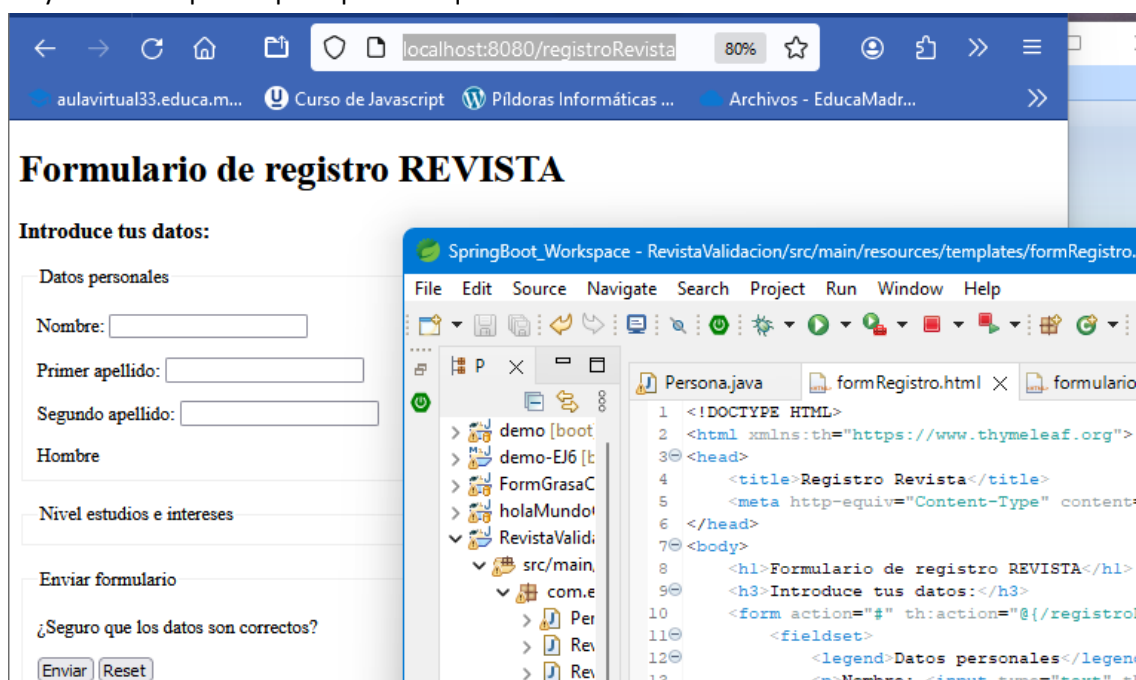
- Creo **VISTA** plantilla formulario formRegistro.html para el GetMapping:

File

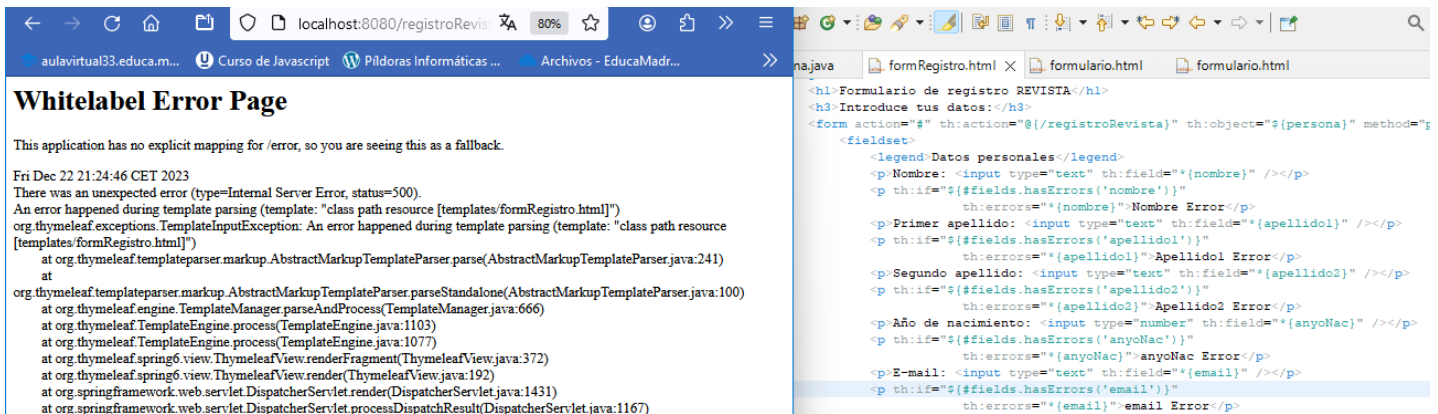
Create a new file resource.



Voy creándolo poco a poco para ver que funciona.

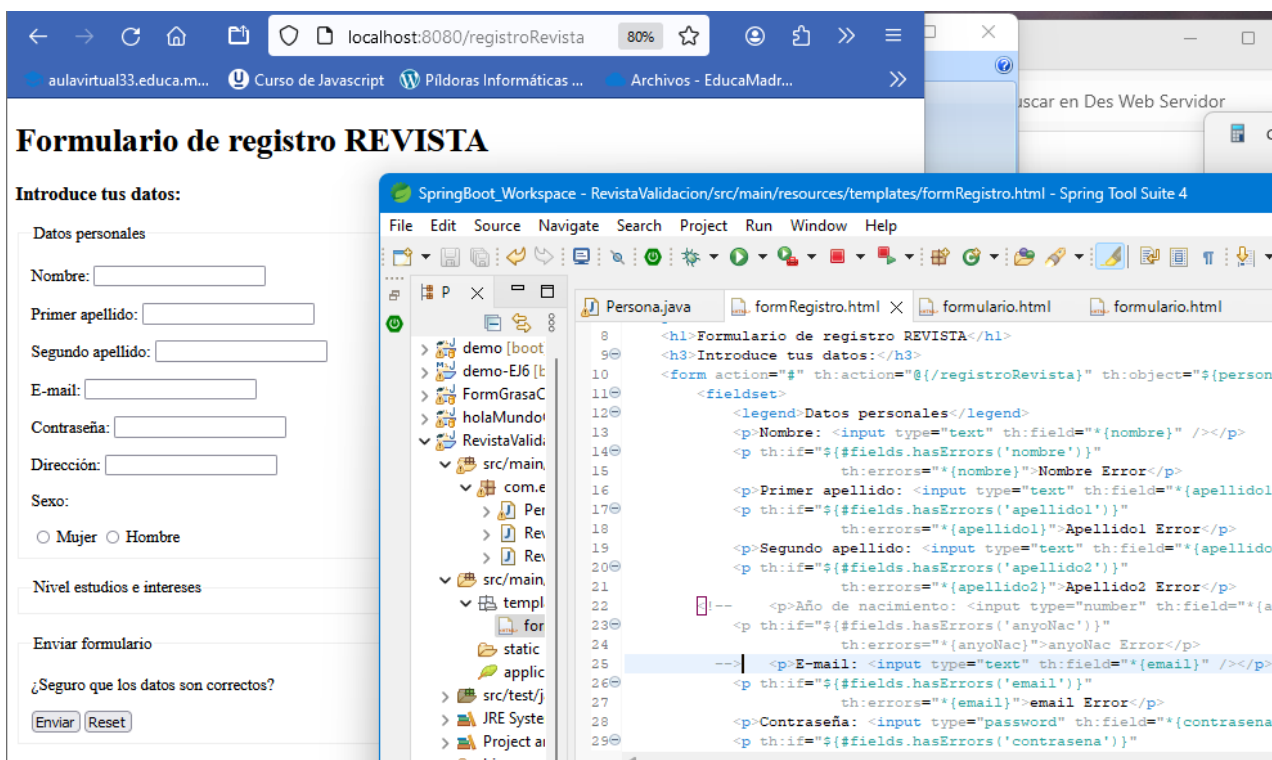


En una de las pruebas veo que falla, voy comentando líneas para ver dónde está el fallo, y veo que el fallo está en la parte año de nacimiento:



The screenshot shows a web browser at localhost:8080/registroRevista displaying a 'Whitelabel Error Page'. The error message states: 'This application has no explicit mapping for /error, so you are seeing this as a fallback.' Below this, a stack trace is visible, starting with 'Fri Dec 22 21:24:46 CET 2023' and 'There was an unexpected error (type=Internal Server Error, status=500). An error happened during template parsing (template: "class path resource [templates/formRegistro.html]")'. The stack trace points to 'org.thymeleaf.exceptions.TemplateInputException: An error happened during template parsing (template: "class path resource [templates/formRegistro.html]")' at 'org.thymeleaf.templateparser.markup.AbstractMarkupTemplateParser.parse'.

Overlaid on the right is a code editor showing the HTML template 'formRegistro.html'. The form includes fields for 'Nombre', 'Primer apellido', 'Segundo apellido', 'Año de nacimiento', 'E-mail', and 'Contraseña'. The 'Año de nacimiento' field is highlighted in blue in the code editor.



The screenshot shows a web browser at localhost:8080/registroRevista displaying the 'Formulario de registro REVISTA'. The form is titled 'Introduce tus datos:' and includes sections for 'Datos personales' (Nombre, Primer apellido, Segundo apellido, E-mail, Contraseña, Dirección, Sexo) and 'Nivel estudios e intereses'. There are 'Enviar' and 'Reset' buttons at the bottom.

Overlaid on the right is a code editor showing the HTML template 'formRegistro.html'. The form includes fields for 'Nombre', 'Primer apellido', 'Segundo apellido', 'Año de nacimiento', 'E-mail', and 'Contraseña'. The 'E-mail' field is highlighted in blue in the code editor.

Revisando todo el código y la clase Controller me di cuenta de que no había añadido el get y set de este atributo, por lo que los añado y vuelvo al navegador, por fin funciona:

← → ↻ 🏠 📁 localhost:8080/registroRevista 80%

aulavirtual33.educa.m... Curso de Javascript Píldoras Informáticas ... Arc

Formulario de registro REVISTA

Introduce tus datos:

Datos personales

Nombre:

Primer apellido:

Segundo apellido:

Año de nacimiento:

E-mail:

Contraseña:

Dirección:

Sexo:

☐ Mujer ☐ Hombre

Nivel estudios e intereses

Enviar formulario

¿Seguro que los datos son correctos?

Sigo completando el formulario con el resto de datos:

```

1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4   <title>Registro Revista</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8   <h1>Formulario de registro REVISTA</h1>
9   <h3>Introduce tus datos:</h3>
10  <form action="#" th:action="@{/registroRevista}" th:object="${persona}" method="post">
11    <fieldset>
12      <legend><strong> Datos personales</strong></legend>
13      <p>Nombre: <input type="text" th:field="*{nombre}" /></p>
14      <p th:if="${#fields.hasErrors('nombre')}">
15        th:errors="*{nombre}">Nombre Error</p>
16      <p>Primer apellido: <input type="text" th:field="*{apellido1}" /></p>
17      <p th:if="${#fields.hasErrors('apellido1')}">
18        th:errors="*{apellido1}">Apellido1 Error</p>
19      <p>Segundo apellido: <input type="text" th:field="*{apellido2}" /></p>
20      <p th:if="${#fields.hasErrors('apellido2')}">
21        th:errors="*{apellido2}">Apellido2 Error</p>
22      <p>Año de nacimiento: <input type="number" th:field="*{anyoNac}" /></p>
23      <p th:if="${#fields.hasErrors('anyoNac')}">
24        th:errors="*{anyoNac}">anyoNac Error</p>
25      <p>E-mail: <input type="text" th:field="*{email}" /></p>
26      <p th:if="${#fields.hasErrors('email')}">
27        th:errors="*{email}">email Error</p>
28      <p>Contraseña: <input type="password" th:field="*{contrasena}" /></p>
29      <p th:if="${#fields.hasErrors('contrasena')}">
30        th:errors="*{contrasena}">contrasena Error</p>

```

```

31      <p>Dirección: <input type="text" th:field="*{direccion}" /></p>
32      <p th:if="${#fields.hasErrors('direccion')}"
33          th:errors="*{direccion}">direccion Error</p>
34      <br>
35      <p>Sexo:</p>
36      <input type="radio" id="mujer" th:field="*{sexo}" value="mujer"/>
37      <label for="mujer">Mujer</label>
38      <input type="radio" id="hombre" th:field="*{sexo}" value="hombre"/>
39      <label for="hombre">Hombre</label>
40
41  </fieldset>
42  <br>
43  <fieldset>
44      <legend><strong>Nivel estudios e intereses</strong></legend>
45      <label>Nivel de estudios:</label><br>
46      <input type="radio" th:field="*{estudios}" value="escolar"/>
47      <label/>Certificado escolar</label><br>
48      <input type="radio" th:field="*{estudios}" value="eso"/>
49      <label/>Graduado en E.S.O.</label><br>
50      <input type="radio" th:field="*{estudios}" value="bach"/>
51      <label/>Bachiller</label><br>
52      <input type="radio" th:field="*{estudios}" value="fpm"/>
53      <label/>Formación Profesion GRADO MEDIO</label><br>
54      <input type="radio" th:field="*{estudios}" value="fpS"/>
55      <label/>Formación Profesion GRADO SUPERIOR</label><br>
56      <input type="radio" th:field="*{estudios}" value="diplo"/>
57      <label/>Diplomado</label><br>
58      <input type="radio" th:field="*{estudios}" value="lic-doc"/>
59      <label/>Licenciado o Doctorado</label>
60      <br><br>
61
62      <input type="checkbox" th:field="*{temas}" value="musica"/>
63      <label>Música</label><br>
64      <input type="checkbox" th:field="*{temas}" value="deportes"/>
65      <label>Deportes</label><br>
66      <input type="checkbox" th:field="*{temas}" value="cine"/>
67      <label>Cine</label><br>
68      <input type="checkbox" th:field="*{temas}" value="libros"/>
69      <label>Libros</label><br>
70      <input type="checkbox" th:field="*{temas}" value="cocina"/>
71      <label>Cocina</label><br>
72      <input type="checkbox" th:field="*{temas}" value="viajes"/>
73      <label>Viajes</label>
74  </fieldset>
75  <br>
76  <fieldset>
77      <legend><strong> Enviar formulario</strong></legend>
78      <p>¿Seguro que los datos son correctos?</p>
79      <input type="submit" value="Enviar" /> <input type="reset" value="Reset" />
80  </fieldset>
81  </form>
82 </body>
83 </html>

```

Hago la prueba en el navegador, se visualiza todo correctamente, doy a enviar sin rellenar datos:

localhost:8080/registro

80%

aulavirtual33.educa.m...

Curso de Javascript

Píldoras Informáticas ...

Formulario de registro REVISTA

Introduce tus datos:

Datos personales

Nombre:

no debe estar vacío

Primer apellido:

no debe estar vacío

Segundo apellido:

Año de nacimiento:

E-mail:

debe coincidir con ".*@.*\.(com|es)\$"

Contraseña:

el tamaño debe estar entre 5 y 10

Dirección:

el tamaño debe estar entre 8 y 2147483647

Nivel estudios e intereses

Nivel de estudios:

☐ Certificado escolar

☐ Graduado en E.S.O.

☐ Bachiller

☐ Formación Profesional GRADO MEDIO

☐ Formación Profesional GRADO SUPERIOR

☐ Diplomado

☐ Licenciado o Doctorado

Recibir artículos sobre los siguientes temas:

☐ Música

☐ Deportes

☐ Cine

☐ Libros

☐ Cocina

☐ Viajes

Enviar formulario

¿Seguro que los datos son correctos?

Enviar

Reset

Salen mensajes de error correctamente, menos en el campo temas, que también es obligatorio. Y tras mucho investigar sin éxito, vuelvo a revisar el código y me doy cuenta de que no había puesto en la parte de los checkbox, el campo de errores por lo que obviamos sin esa línea, no iba a salir el mensaje de error....

Lo modifico el html y vuelvo a probar:

```

74         <label>Viajes</label>
75         <p th:if="{#fields.hasErrors('temas')}"
76             th:errors="{#fields.hasErrors('temas')}">temas Error</p>
77     </fieldset>
78     <br>

```

Sigue sin salir el error, ya recorro a ChatGpt y tras varios prompts, me sugiere que el array temas lo debo de inicializar. Por lo que en la clase Modelo Persona, creo constructor y ahí inicializo el array:

```

101     }
102     // Constructor que inicializa el array temas
103     public Persona() {
104         this.temas = new String[]{};
105     }

```

Vuelvo a probar en el navegador y ahora sí que sale el mensaje:

el tamaño debe estar entre 8 y 2147483647

Sexo:

☐ Mujer ☐ Hombre

Nivel estudios e intereses

Nivel de estudios:

☐ Certificado escolar

☐ Graduado en E.S.O.

☐ Bachiller

☐ Formación Profesional GRADO MEDIO

☐ Formación Profesional GRADO SUPERIOR

☐ Diplomado

☐ Licenciado o Doctorado

Recibir artículos sobre los siguientes temas:

☐ Música

☐ Deportes

☐ Cine

☐ Libros

☐ Cocina

☐ Viajes

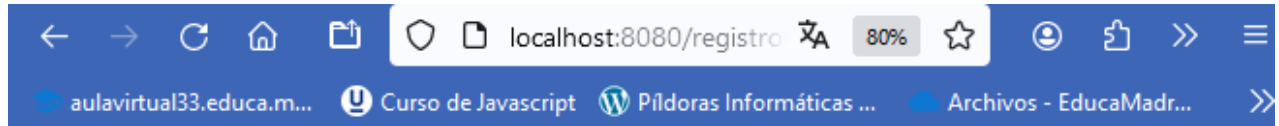
no debe estar vacío

el tamaño debe estar entre 2 y 4

Enviar formulario

¿Seguro que los datos son correctos?

Pongo datos correctos y doy a enviar. Sale error porque no hemos creado plantilla registroOK:



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Dec 22 23:11:39 CET 2023

There was an unexpected error (type=Internal Server Error, status=500).

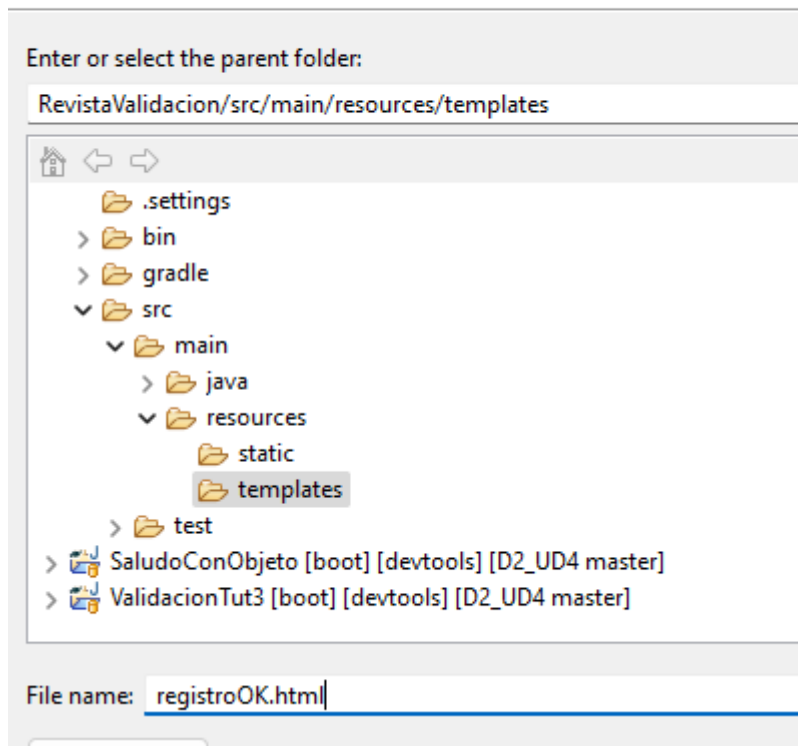
Error resolving template [registroOK], template might not exist or might not be accessible by any of the configured Template Resolvers

org.thymeleaf.exceptions.TemplateInputException: Error resolving template [registroOK], template might not exist or might not be accessible by any of the configured Template Resolvers

- Creo **VISTA** registroOK que devuelve el PostMapping:

File

Create a new file resource.

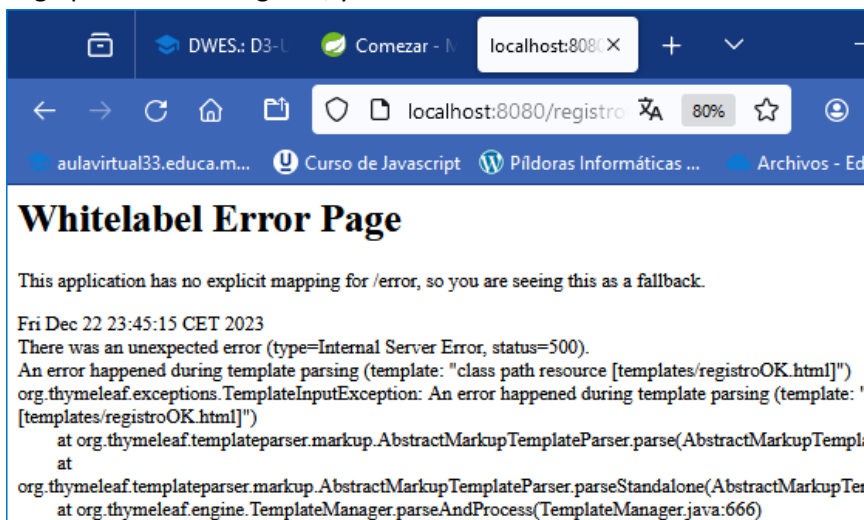


```

Persona.java registroOK.html resultados.html *RevistaController.java
1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4     <title>Registro Revista</title>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8     <h1>Formulario de registro REVISTA</h1>
9     <h3>Introduce tus datos:</h3>
10    <p th:text="'Nombre: ' + ${persona.nombre}" />
11    <p th:text="'Primer Apellido: ' + ${persona.apellido1}" />
12    <p th:text="'Segundo Apellido: ' + ${persona.apellido2}" />
13    <p th:text="'Años de Nacimiento: ' + ${persona.anyoNac}" />
14    <p th:text="'E-mail: ' + ${persona.email}" />
15    <p th:text="'Contraseña: ' + ${persona.contrasena}" />
16    <p th:text="'Dirección: ' + ${persona.direccion}" />
17 </body>
18 </html>
19

```

Hago prueba en navegador, y sale error:



Me doy cuenta de que no había añadido el Model y model.addAttribute en el controlador, eso es lo que hace pasar los datos entre modelo y vista. Lo añado, tanto en el GetMapping como en el PostMapping y también hago un cambio en el método adViewControllers:

```

@Override
public void addViewControllers(ViewControllerRegistry registry) {
    registry.addViewController("/registroRevista").setViewName("registroOK");
}

@GetMapping("/registroRevista")
public String mostrarFormRegistro(Model model) {
    model.addAttribute("persona", new Persona());
    return "formRegistro";
}

@PostMapping("/registroRevista")
public String registroCorrecto(@Valid @ModelAttribute("persona")
    Persona persona, BindingResult bindingResult, Model model) {

    if (bindingResult.hasErrors()) {
        return "formRegistro";
    }
    return "registroOK";
}

```

Y ahora en el navegador salen los datos:



← → ↺ 🏠 📁 localhost:8080/registroRevis... 80%

aulavirtual33.educa.m... Curso de Javascript Píldoras Informáticas ...

Formulario de registro REVISTA

Introduce tus datos:

Nombre: arancha

Primer Apellido: chicharro

Segundo Apellido:

Años de Nacimiento: 0

E-mail: arancha_20_91@hotmail.com

Contraseña: sssssssss

Dirección: 555555555555555555555555

Sigo modificando la plantilla de registroOK para que aparezcan los demás datos, y el campo de contraseña se podría capar, pero como es nuestra página de datos de registro, está bien que sepamos la contraseña elegida.

Para mostrar los temas, como es un array, tenemos que usar `th:each` para recorrerlo. Y lo muestro en una lista para que salgan todos los temas seleccionados listados y no todo junto:

```
Persona.java  RevistaController.java  registroOK.html X
1  <!DOCTYPE HTML>
2  <html xmlns:th="https://www.thymeleaf.org">
3  <head>
4      <title>Registro Revista</title>
5      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6  </head>
7  <body>
8      <h1>Formulario de registro REVISTA completado</h1>
9      <h3>RESUMEN DE LOS DATOS:</h3>
10     <fieldset>
11         <legend><strong> Datos personales</strong></legend>
12         <p th:text="'Nombre: ' + ${persona.nombre}" />
13         <p th:text="'Primer Apellido: ' + ${persona.apellido1}" />
14         <p th:text="'Segundo Apellido: ' + ${persona.apellido2}" />
15         <p th:text="'Años de Nacimiento: ' + ${persona.anyoNac}" />
16         <p th:text="'E-mail: ' + ${persona.email}" />
17         <p th:text="'Contraseña: ' + ${persona.contrasena}" />
18         <p th:text="'Dirección: ' + ${persona.direccion}" />
19         <p th:text="'Sexo: ' + ${persona.sexo}" />
20     </fieldset>
21     <fieldset>
22         <legend><strong> Nivel de estudios e intereses</strong></legend>
23         <p th:text="'Estudios: ' + ${persona.estudios}" />
24         <p>Temas de Interés:</p>
25         <ul>
26             <li th:each="tema : ${persona.temas}" th:text="${tema}"></li>
27         </ul>
28     </fieldset>
29 </body>
30 </html>
```

Pruebo en el navegador introduciendo datos:

Formulario de registro REVISTA

Introduce tus datos:

Datos personales

Nombre:

Primer apellido:

Segundo apellido:

Año de nacimiento:

E-mail:

Contraseña:

Dirección:

Sexo:

☒ Mujer ☐ Hombre

Nivel estudios e intereses

Nivel de estudios:

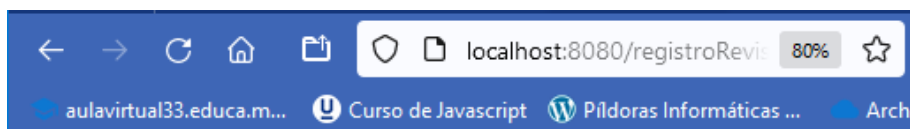
- ☐ Certificado escolar
- ☐ Graduado en E.S.O.
- ☐ Bachiller
- ☐ Formación Profesional GRADO MEDIO
- ☐ Formación Profesional GRADO SUPERIOR
- ☒ Diplomado
- ☐ Licenciado o Doctorado

Recibir artículos sobre los siguientes temas:

- ☐ Música
- ☐ Deportes
- ☒ Cine
- ☒ Libros
- ☒ Cocina
- ☒ Viajes

Enviar formulario

¿Seguro que los datos son correctos?



Formulario de registro REVISTA completado

RESUMEN DE LOS DATOS:

Datos personales

Nombre: Pepita

Primer Apellido: García

Segundo Apellido: López

Años de Nacimiento: 1989

E-mail: pepita@gmail.com

Contraseña: 123456

Dirección: calle mayor 35, madrid

Sexo: mujer

Nivel de estudios e intereses

Estudios: diplo

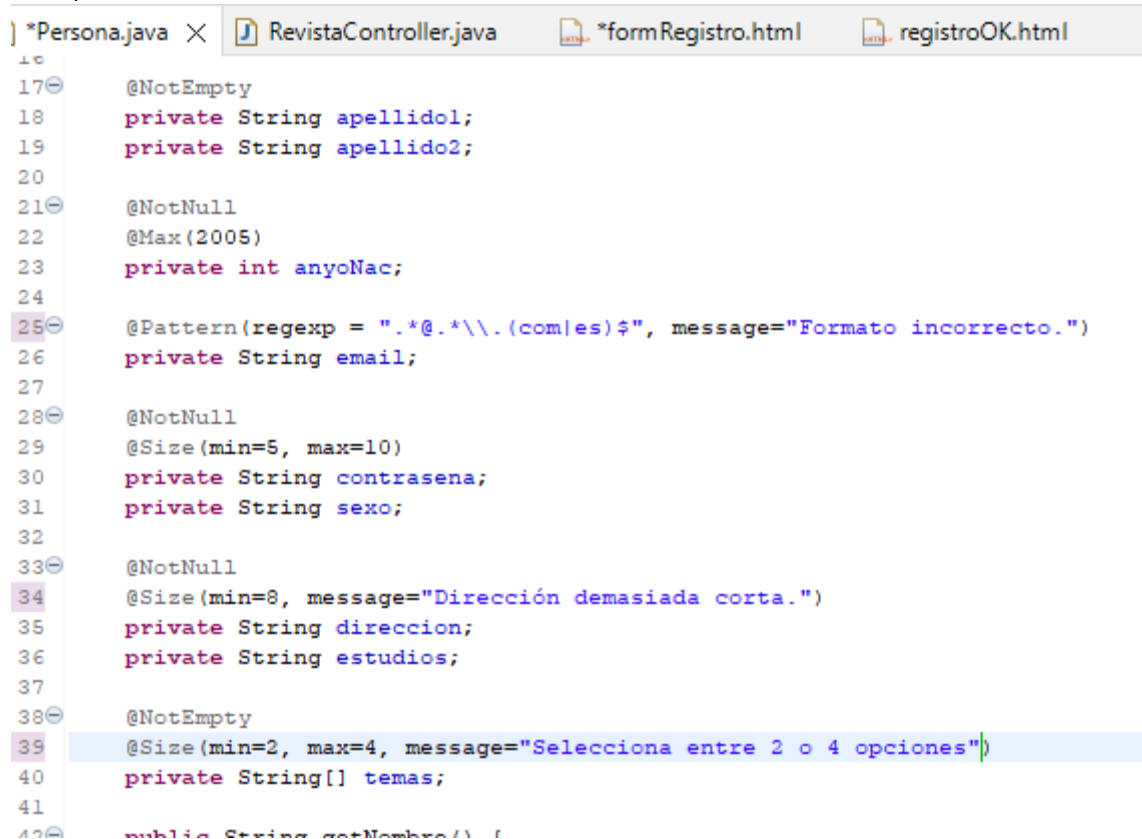
Temas de Interés:

- cine
- libros
- cocina
- viajes

Me doy cuenta que en estudios, el valor que aparece es el value del input que está en modo clave, lo modifico en el html para que aparezca toda la palabra

```
<legend><strong>Nivel estudios e intereses</strong></legend>
<label>Nivel de estudios:</label><br>
<input type="radio" th:field="**{estudios}" value="Certificado escolar"/>
<label/>Certificado escolar</label><br>
<input type="radio" th:field="**{estudios}" value="E.S.O."/>
<label/>Graduado en E.S.O.</label><br>
<input type="radio" th:field="**{estudios}" value="Bachiller"/>
<label/>Bachiller</label><br>
<input type="radio" th:field="**{estudios}" value="FP G.M."/>
<label/>Formación Profesional GRADO MEDIO</label><br>
<input type="radio" th:field="**{estudios}" value="FP G.S."/>
<label/>Formación Profesional GRADO SUPERIOR</label><br>
<input type="radio" th:field="**{estudios}" value="Diplomado"/>
<label/>Diplomado</label><br>
<input type="radio" th:field="**{estudios}" value="Licenciado o Doctorado"/>
<label/>Licenciado o Doctorado</label>
```

También modifico la clase Persona para personalizar algunos mensajes de error (el de la dirección, temas, email).



```
17 @NotEmpty
18 private String apellidos;
19 private String apellido2;
20
21 @NotNull
22 @Max(2005)
23 private int anyoNac;
24
25 @Pattern(regexp = ".*@.*\\.\\.(com|es)$", message="Formato incorrecto.")
26 private String email;
27
28 @NotNull
29 @Size(min=5, max=10)
30 private String contrasena;
31 private String sexo;
32
33 @NotNull
34 @Size(min=8, message="Dirección demasiado corta.")
35 private String direccion;
36 private String estudios;
37
38 @NotEmpty
39 @Size(min=2, max=4, message="Selecciona entre 2 o 4 opciones")
40 private String[] temas;
41
42 public String getNombre() {
```

Ahora pruebo otra vez en el navegador para ver estos nuevos mensajes de error, también introduzco una fecha mayor de 2005 para ver este error no visto en las pruebas anteriores porque no es campo obligatorio:

Nombre: <input type="text"/>	Año de nacimiento: <input type="text" value="2010"/>
Primer apellido: <input type="text"/>	debe ser menor que o igual a 2005
Segundo apellido: <input type="text"/>	E-mail: <input type="text" value="aaaa.com"/>
Año de nacimiento: <input type="text" value="2010"/>	Formato incorrecto.
E-mail: <input type="text" value="aaaa.com"/>	Contraseña: <input type="text"/>
Contraseña: <input type="text"/>	el tamaño debe estar entre 5 y 10
Dirección: <input type="text"/>	Dirección: <input type="text"/>
	Dirección demasiado corta
Sexo:	Sexo:
<input type="radio"/> Mujer <input type="radio"/> Hombre	<input type="radio"/> Mujer <input type="radio"/> Hombre
<hr/>	
Nivel estudios e intereses	Nivel estudios e intereses
Nivel de estudios:	Nivel de estudios:
<input type="radio"/> Certificado escolar	<input type="radio"/> Certificado escolar
<input type="radio"/> Graduado en E.S.O.	<input type="radio"/> Graduado en E.S.O.
<input type="radio"/> Bachiller	<input type="radio"/> Bachiller
<input type="radio"/> Formación Profesiona GRADO MEDIO	<input type="radio"/> Formación Profesiona GRADO MEDIO
<input type="radio"/> Formación Profesiona GRADO SUPERIOR	<input type="radio"/> Formación Profesiona GRADO SUPERIOR
<input type="radio"/> Diplomado	<input type="radio"/> Diplomado
<input type="radio"/> Licenciado o Doctorado	<input type="radio"/> Licenciado o Doctorado
Recibir artículos sobre los siguientes temas:	Recibir artículos sobre los siguientes temas:
<input checked="" type="checkbox"/> Música	<input checked="" type="checkbox"/> Música
<input type="checkbox"/> Deportes	<input type="checkbox"/> Deportes
<input type="checkbox"/> Cine	<input type="checkbox"/> Cine
<input type="checkbox"/> Libros	<input type="checkbox"/> Libros
<input type="checkbox"/> Cocina	<input type="checkbox"/> Cocina
<input type="checkbox"/> Viajes	<input type="checkbox"/> Viajes
	Selecciona entre 2 o 4 opciones

Por último, añado datos correctos y vuelvo a enviar:

Formulario de registro REVISTA completado

RESUMEN DE LOS DATOS:

Datos personales

Nombre: Arancha

Primer Apellido: Chicharro

Segundo Apellido: Montejano

Años de Nacimiento: 1991

E-mail: aa@aa.com

Contraseña: 123456

Dirección: calle mayor 35, madrid

Sexo: mujer

Nivel de estudios e intereses

Estudios: FP G.S.

Temas de Interés:

- musica
- cocina
- viajes

CONCLUSIONES:

Esta práctica ha sido muy formativa, desde el primer tutorial hasta el último ejercicio, he ido aprendiendo poco a poco, y avanzando en código y conocimientos. Ha sido muy interesante ir viendo esta mejoría y sobre todo el resultado en el navegador.

El resultado de todo el código junto, la unión de html/java, comprender el **modelo/vista/controlador**. Al hacer todas las partes de este reto, he avanzado mucho y aprendido sobre el código, volver al código Java y mejorar sobre él.

También la parte de validación a través de todas las anotaciones en la clase Modelo, y como relacionar los atributos del MODELO(clase objeto) con la VISTA (html) .

Después de hacer todas las pruebas y su correcto resultado, aún con problemas pero que he ido solucionando, me puedo sentir muy satisfecha y contenta con el resultado y con todo lo aprendido.