Contenido

[**Sección 81: Spring Framework 5: Formularios Thymeleaf y Data Binding** 3](#_Toc147425947)

[631. Creando nuevo proyecto form 3](#_Toc147425948)

[632. Actualización: para Spring Boot 2.3.0 o superior agregar dependencia validation 3 min 3](#_Toc147425949)

[633. Añadiendo la vista del formulario 3](#_Toc147425950)

[634. Método handler para recibir y procesar los datos enviados 3](#_Toc147425951)

[635. Creando la clase model del formulario 6](#_Toc147425952)

[636. Mapeando los campos del formulario al objeto model 8](#_Toc147425953)

[637. Validación del formulario usando anotaciones 8](#_Toc147425954)

[**REVISAR BIEN LUEGO EL PORQUE NO FUNCIONA COMO ESPERO** 9](#_Toc147425955)

[638. Añadiendo los mensajes de errores en la vista del formulario 9](#_Toc147425956)

[PROYECTO FORM 2 11](#_Toc147425957)

[639. Mostrando los mensajes de errores en la vista del formulario 12](#_Toc147425958)

[640. Probando el formulario 12](#_Toc147425959)

[641. La anotación @SessionAttributes para manter los datos durante el ciclo del form 12](#_Toc147425960)

[642. La anotación @Size para rangos y @Email para validar correo electrónico 12](#_Toc147425961)

[643. Mensajes de errores personalizados usando properties 12](#_Toc147425962)

[644. Archivos properties y acentos con UTF-8 12](#_Toc147425963)

[645. Validación personalizada usando anotación @Pattern para expresiones regulares 12](#_Toc147425964)

[646. Validación personalizada usando una clase Validation 12](#_Toc147425965)

[647. Registrando una clase validador con la anotación @InitBinder 12](#_Toc147425966)

[648. Validación personalizada usando anotaciones 12](#_Toc147425967)

[649. Validación personalizada usando 12](#_Toc147425968)

[anotaciones ejemplo requerido 12](#_Toc147425969)

[650. Validación de números enteros con @Max @Min y NotNull 12](#_Toc147425970)

[651. Validación de fechas con @NotNull 12](#_Toc147425971)

[dando formato con @DateTimeFormat 12](#_Toc147425972)

[652. Validación de fechas con input type date de HTML5 y @Past y @Future 12](#_Toc147425973)

[653. Formateando fechas con @lnitBinder y registrando un CustomDateEditor 12](#_Toc147425974)

[654. Formateando fechas con @lnitBinder y registrando un CustomDateEditor Parte 2 13](#_Toc147425975)

[655. Implementando propio filtro custom property editor para convertir a mayúscula 13](#_Toc147425976)

[656. Lista select desplegable 13](#_Toc147425977)

[657. Formateando fechas en vistas thymeleaf 13](#_Toc147425978)

[658. Llenando lista select con Map 13](#_Toc147425979)

[659. Llenando lista select con objetos del la clase Pais 13](#_Toc147425980)

[660. Añadiendo componente service de Pais 13](#_Toc147425981)

[661. Añadiendo componente pais property editor 13](#_Toc147425982)

[662. Implementando checkboxes 13](#_Toc147425983)

[663. Llenando checkboxes con Map 13](#_Toc147425984)

[664. Llenando checkboxes con objetos del tipo Role 13](#_Toc147425985)

[665. Añadiendo componente role property editor 13](#_Toc147425986)

[666. Checkbox booleano true o false 13](#_Toc147425987)

[667. Radio button 13](#_Toc147425988)

[668. Input type hidden 13](#_Toc147425989)

[669. Poblando pais y roles 13](#_Toc147425990)

[670. Redirect después del POST procesar 13](#_Toc147425991)

[671. Añadiendo estilos CSS al formulario con Bootstrap 13](#_Toc147425992)

[672. Añadiendo estilos CSS Bootstrap a la plantilla resultado 13](#_Toc147425993)

[673. Cargando archivo css Bootstrap localmente 13](#_Toc147425994)

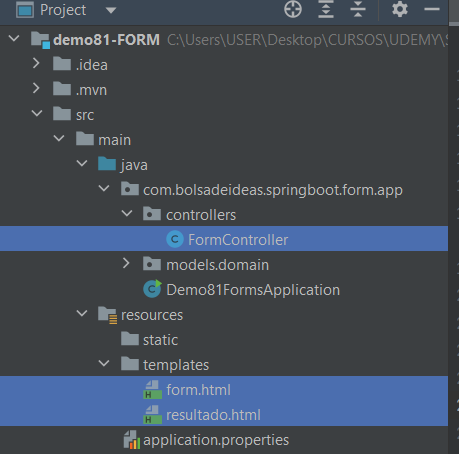
# **Sección 81: Spring Framework 5: Formularios Thymeleaf y Data Binding**

# 631. Creando nuevo proyecto form

# 632. Actualización: para Spring Boot 2.3.0 o superior agregar dependencia validation 3 min

# 633. Añadiendo la vista del formulario

# 634. Método handler para recibir y procesar los datos enviados



package com.bolsadeideas.springboot.form.app.controllers;  
  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
  
@Controller  
public class FormController {  
  
 @GetMapping("/form")  
 public String form(Model model){  
 model.addAttribute("titulo","Creando usuario");  
 return "form";  
 }  
  
 @PostMapping("/form")  
 public String procesar(Model model,  
 @RequestParam String username,  
 @RequestParam String password,  
 @RequestParam String email){  
  
  
 model.addAttribute("titulo","Procesando usuario");  
  
 model.addAttribute("username",username);  
 model.addAttribute("password",password);  
 model.addAttribute("email",email);  
  
 return "resultado";  
 }  
  
}

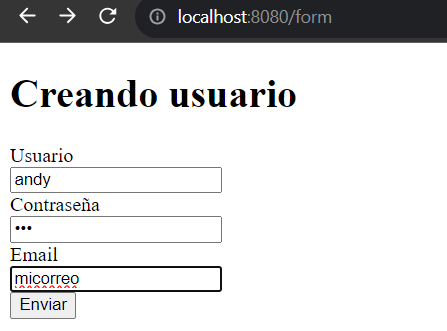
form.html

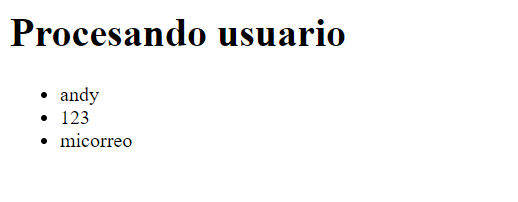
<!doctype html>  
<html lang="en" xmlns:th="http://thymeleaf.org">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Form</title>  
</head>  
  
<body>  
  
<h1 th:text="${titulo}"></h1>  
  
 <form th:action="@{/form}">  
  
 <div>  
 <label for="username">Usuario </label>  
 <div>  
 <input type="text" name="username" id="username">  
 </div>  
 </div>  
  
 <div>  
 <label for="password">Contraseña </label>  
 <div>  
 <input type="password" name="password" id="password">  
 </div>  
 </div>  
  
 <div>  
 <label for="email">Email </label>  
 <div>  
 <input type="text" name="email" id="email">  
 </div>  
 </div>  
  
 <div>  
 <div>  
 <input type="submit" value="Enviar">  
 </div>  
 </div>  
  
 </form>  
  
</body>  
  
</html>

Resulado.html

<!doctype html>  
<html lang="en" xmlns:th="http://thymeleaf.org">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Form</title>  
</head>  
  
<body>  
  
<h1 th:text="${titulo}"></h1>  
  
<ul>  
  
 <li th:text="${username}"></li>  
 <li th:text="${password}"></li>  
 <li th:text="${email}"></li>  
</ul>

</body>  
  
</html>





# 635. Creando la clase model del formulario

A diferencia del anterior creamos la clase models/domain/Usuario

package com.bolsadeideas.springboot.form.app.models.domain;  
  
import lombok.Data;  
  
@Data  
public class Usuario {  
  
 //deben ser igual al name del form  
  
 private String username;  
 private String password;  
 private String email;  
  
}

Modificamos el metodo con el Post del controller (PARA ALMACENAR EL ANTERIOR COMENTAMOS Y LO VOLVIMOS A CREAR)

package com.bolsadeideas.springboot.form.app.models.domain;  
  
import lombok.Data;  
  
@Data  
public class Usuario {  
  
 //deben ser igual al name del form  
  
 private String username;  
 private String password;  
 private String email;  
  
}

FormController

instanciamos la clase Usuario y la add al model

@PostMapping("/form")  
public String procesar(Model model,  
 @RequestParam String username,  
 @RequestParam String password,  
 @RequestParam String email){  
  
 model.addAttribute("titulo","Procesando usuario");  
  
 Usuario usuario = new Usuario();  
 usuario.setUsername(username);  
 usuario.setPassword(password);  
 usuario.setEmail(email);  
  
 model.addAttribute("usuario",usuario);  
  
 return "resultado";  
}

form.html

no se modifico nada

resultado.html

solo agregamso el usuario.

<ul>  
 <!-- usuario es del cotroller -->  
 <li th:text="${usuario.username}"></li>  
 <li th:text="${usuario.password}"></li>  
 <li th:text="${usuario.email}"></li>  
</ul>

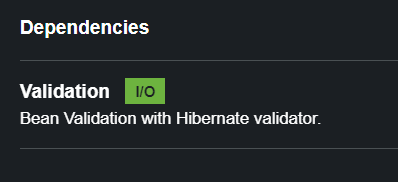
# 636. Mapeando los campos del formulario al objeto model

N el controller movemos la instancia a los parámetros del metodo

//MAPEAMOS LOS CAMPOS DEL FORMULARIO AL OBJETO MODEL  
@PostMapping("/form")  
public String procesar( Usuario usuario , Model model){  
  
 model.addAttribute("titulo","Procesando usuario");  
 model.addAttribute("usuario",usuario);  
  
 return "resultado";  
}

El resto esta igual

# 637. Validación del formulario usando anotaciones



<dependency>

<groupId>javax.validation</groupId>

<artifactId>validation-api</artifactId>

<version>2.0.1.Final</version>

</dependency>

NO NECESITO ESA DEPENDENCIA, necesito usar jakarta desde el model primero exportato del

En el controller necesito

import jakarta.validation.Valid;

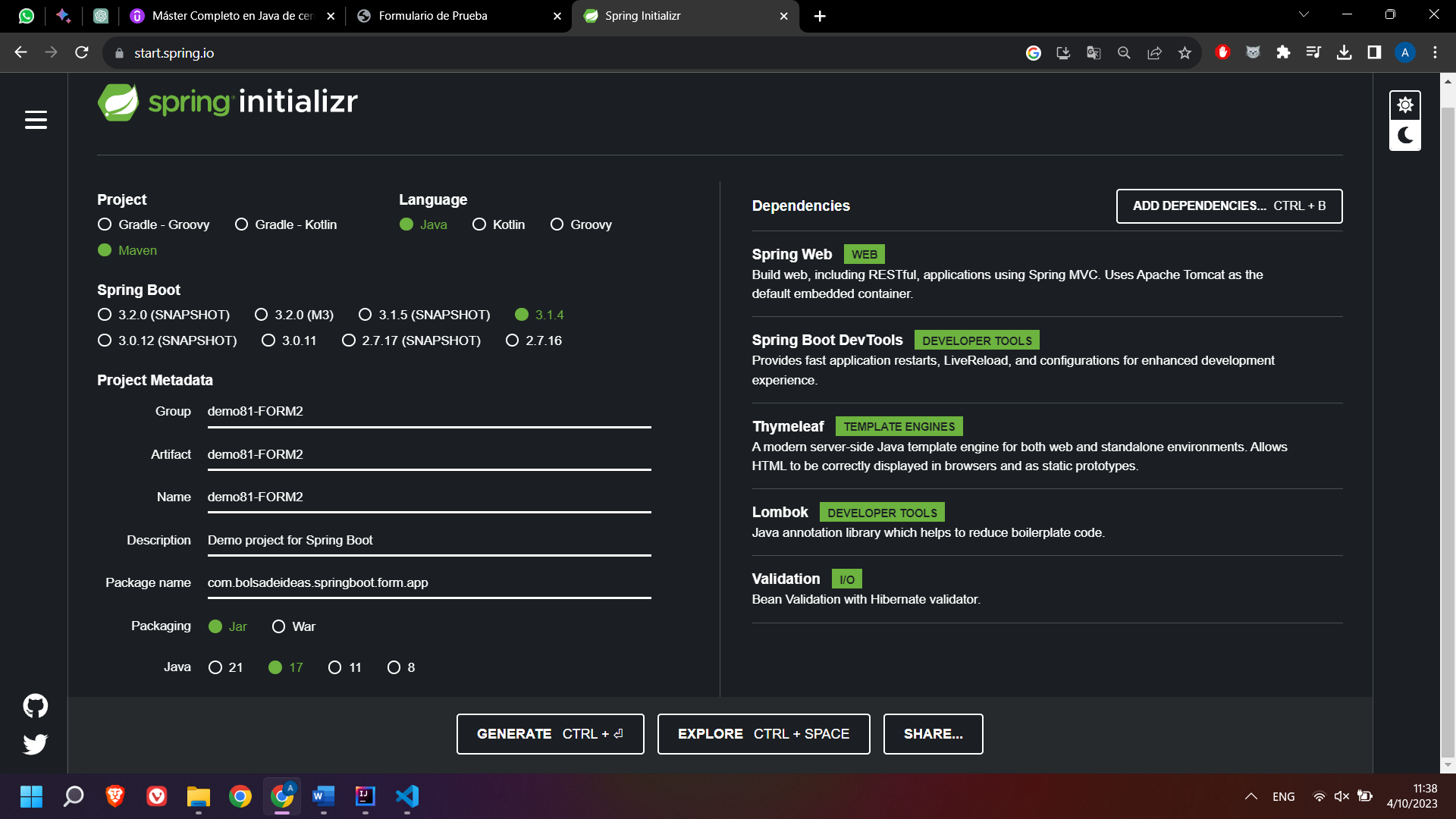
y en el Model

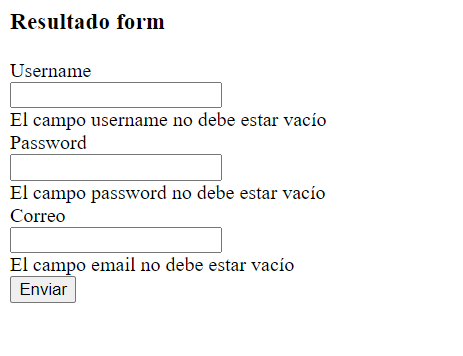
import jakarta.validation.constraints.NotEmpty;

# **REVISAR BIEN LUEGO EL PORQUE NO FUNCIONA COMO ESPERO**

**ESRAMOS TRABAJANDO CON EL FORM2 PERO QUITE EL POMP EL VALIDATE**

# PROYECTO FORM 2



****

package com.bolsadeideas.springboot.form.app.controllers;  
  
import com.bolsadeideas.springboot.form.app.models.domain.Usuario;  
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;//@Valid  
  
import java.util.HashMap;  
import java.util.Map;  
  
@Controller  
public class FormController {  
  
 @GetMapping("/form")  
 public String form(Model model) {  
 Usuario usuario = new Usuario();  
 model.addAttribute("titulo", "Formulario usuarios");  
 model.addAttribute("usuario", usuario);  
 return "form";  
 }  
  
 //Validación del formulario usando anotaciones  
 @PostMapping("/form")  
 public String procesar(@Valid Usuario usuario, BindingResult result, Model model) {  
  
 model.addAttribute("titulo", "Resultado form");  
  
 if(result.hasErrors()) {  
 Map<String, String> errores = new HashMap<>();  
 result.getFieldErrors().forEach(err ->{  
 errores.put(err.getField(), "El campo ".concat(err.getField()).concat(" ").concat(err.getDefaultMessage()));  
 });  
 model.addAttribute("error", errores);  
 return "form";  
 }  
  
 model.addAttribute("usuario", usuario);  
  
 return "resultado";  
 }  
  
}

package com.bolsadeideas.springboot.form.app.models.domain;  
  
import jakarta.validation.constraints.NotEmpty;  
import lombok.Data;  
  
@Data  
public class Usuario {  
  
 @NotEmpty  
 private String username;  
  
 @NotEmpty  
 private String password;  
  
 @NotEmpty  
 private String email;  
  
}

form.html

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
<meta charset="UTF-8">  
<title th:text="${titulo}"></title>  
</head>  
<body>  
 <h3 th:text="${titulo}"></h3>  
  
 <form th:action="@{/form}" method="post">  
  
 <div>  
 <label for="username">Username</label>  
 <div>  
 <input type="text" name="username" id="username" th:value="${usuario.username}">  
 </div>  
 <div th:if="${error != null && error.containsKey('username')}" th:text="${error.username}"></div>  
 </div>  
 <div>  
 <label for="password">Password</label>  
 <div>  
 <input type="password" name="password" id="password">  
 </div>  
 <div th:if="${error != null && error.containsKey('password')}" th:text="${error.password}"></div>  
 </div>  
 <div>  
 <label for="email">Correo</label>  
 <div>  
 <input type="text" name="email" id="email" th:value="${usuario.email}">  
 </div>  
 <div th:if="${error != null && error.containsKey('email')}" th:text="${error.email}"></div>  
 </div>  
   
 <div>  
 <div>  
 <input type="submit" value="Enviar">  
 </div>  
 </div>   
 </form>  
</body>  
</html>

Resultado.html

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
<meta charset="UTF-8">  
<title th:text="${titulo}"></title>  
</head>  
<body>  
 <h3 th:text="${titulo}"></h3>  
  
 <ul>  
 <li th:text="${usuario.username}"></li>  
 <li th:text="${usuario.password}"></li>  
 <li th:text="${usuario.email}"></li>  
 </ul>  
</body>  
</html>

# 

# 638. Añadiendo los mensajes de errores en la vista del formulario

Es el mismo 637 pero la parte del front

# 639. El atributo object de thymeleaf en el elements form

# 640. Mostrar valores de atributos del objeto model en el formulario

# 641. La anotación @SessionAttributes para manter los datos durante el ciclo del form

# 642. La anotación @Size para rangos y @Email para validar correo electrónico

# 643. Mensajes de errores personalizados usando properties

# 644. Archivos properties y acentos con UTF-8

# 645. Validación personalizada usando anotación @Pattern para expresiones regulares

# 646. Validación personalizada usando una clase Validation

# 647. Registrando una clase validador con la anotación @InitBinder

# 648. Validación personalizada usando anotaciones

# 649. Validación personalizada usando

# anotaciones ejemplo requerido

# 650. Validación de números enteros con @Max @Min y NotNull

# 651. Validación de fechas con @NotNull

# dando formato con @DateTimeFormat

# 652. Validación de fechas con input type date de HTML5 y @Past y @Future

# 653. Formateando fechas con @lnitBinder y registrando un CustomDateEditor

# 654. Formateando fechas con @lnitBinder y registrando un CustomDateEditor Parte 2

# 655. Implementando propio filtro custom property editor para convertir a mayúscula

# 656. Lista select desplegable

# 657. Formateando fechas en vistas thymeleaf

# 658. Llenando lista select con Map

# 659. Llenando lista select con objetos del la clase Pais

# 660. Añadiendo componente service de Pais

# 661. Añadiendo componente pais property editor

# 662. Implementando checkboxes

# 663. Llenando checkboxes con Map

# 664. Llenando checkboxes con objetos del tipo Role

# 665. Añadiendo componente role property editor

# 666. Checkbox booleano true o false

# 667. Radio button

# 668. Input type hidden

# 669. Poblando pais y roles

# 670. Redirect después del POST procesar

# 671. Añadiendo estilos CSS al formulario con Bootstrap

# 672. Añadiendo estilos CSS Bootstrap a la plantilla resultado

# 673. Cargando archivo css Bootstrap localmente