

---

# **FORMULARIO SPRING PASO DE PARAMETROS ENTRE CONTROLADOR Y VISTA**

**EDUARD LARA**

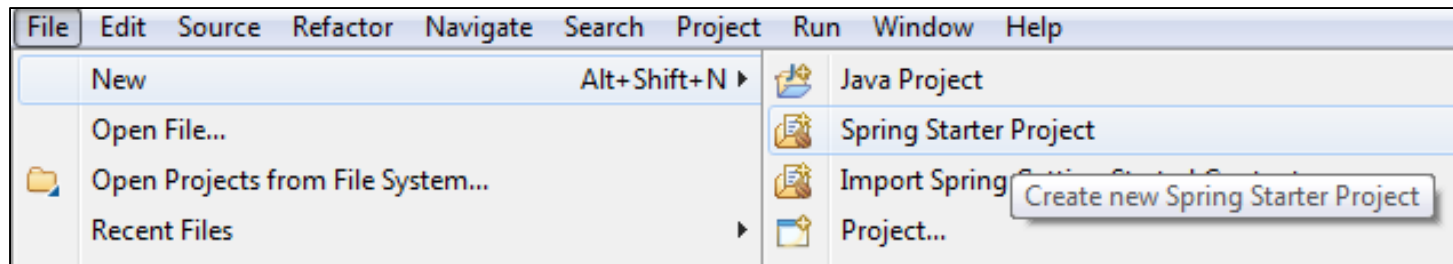
# INDICE

---

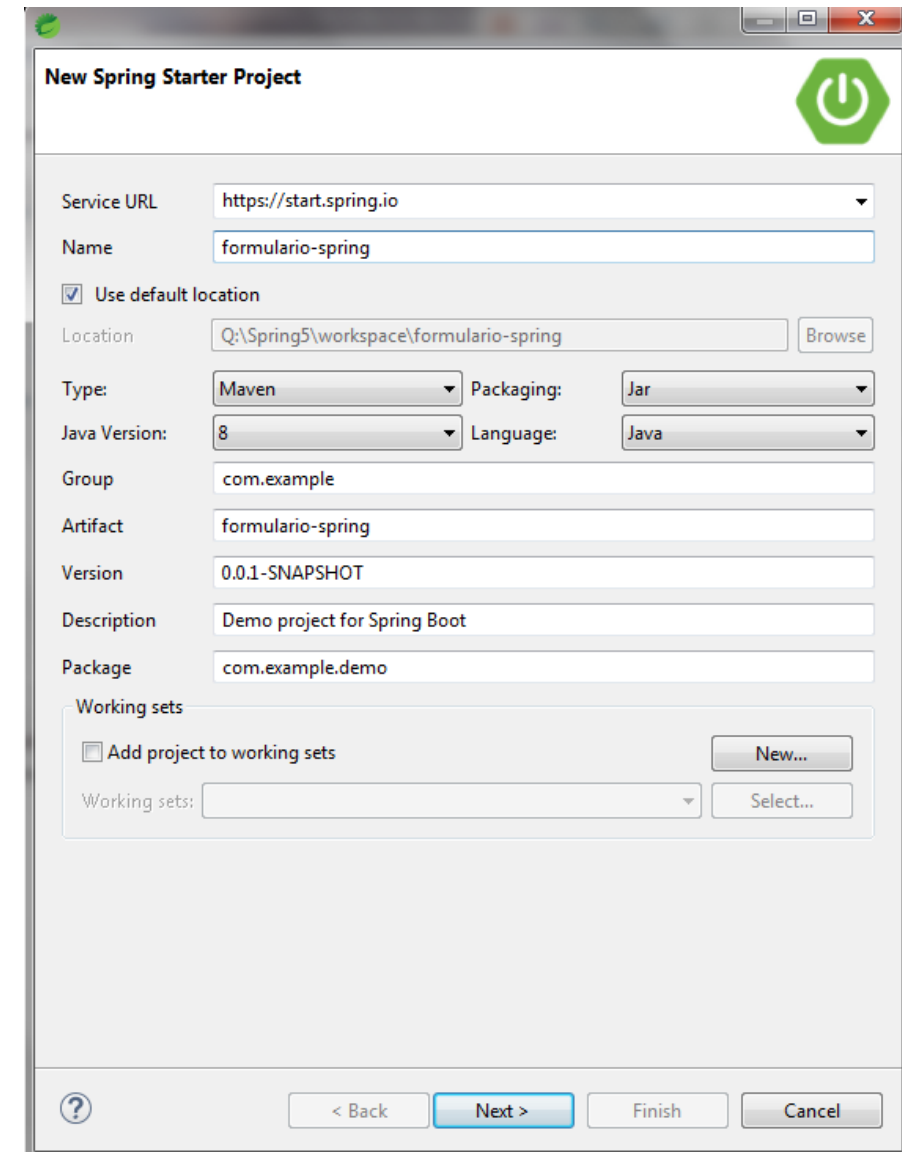
1. Creación proyecto
2. Controlador
3. Creación de vistas
4. Paso parámetros controlador-vista

# 1. CREACION PROYECTO

**Paso 1)** Creamos un proyecto Spring Boot, en la opción de menu File/New/Spring Starter Project:



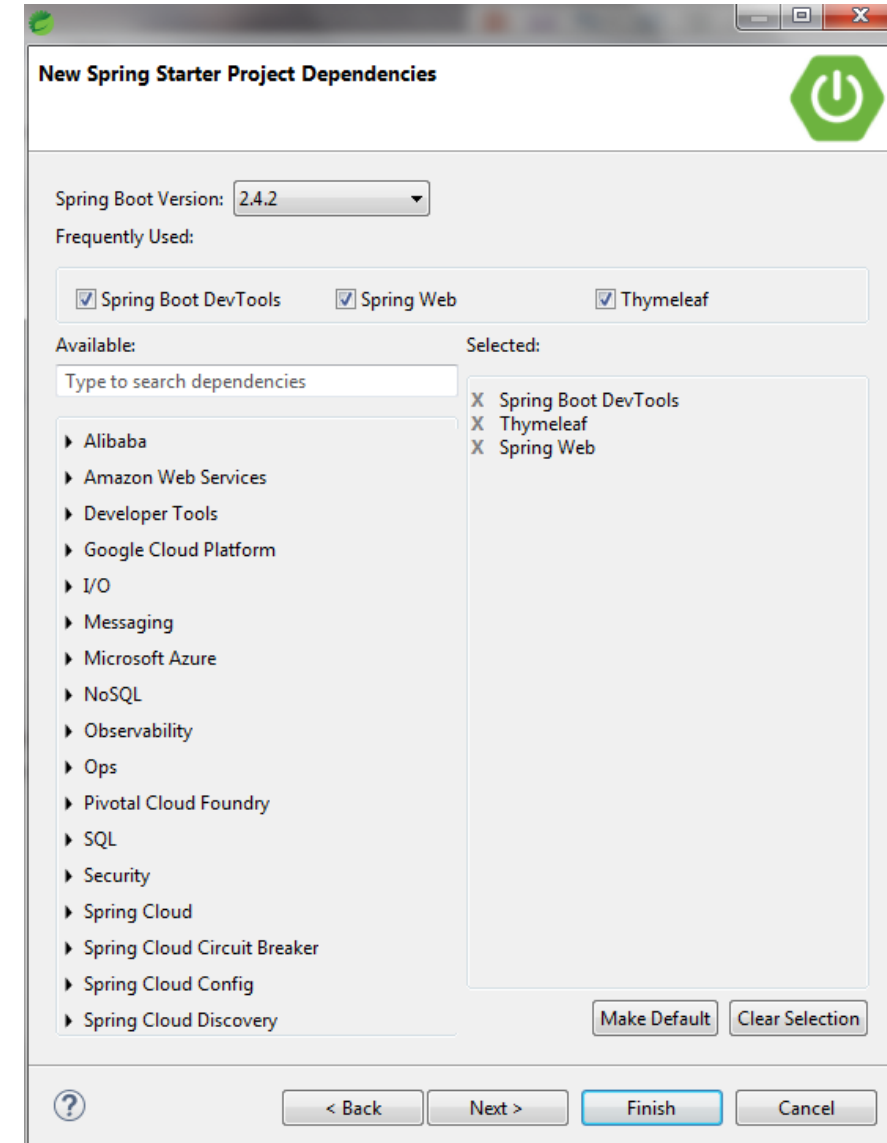
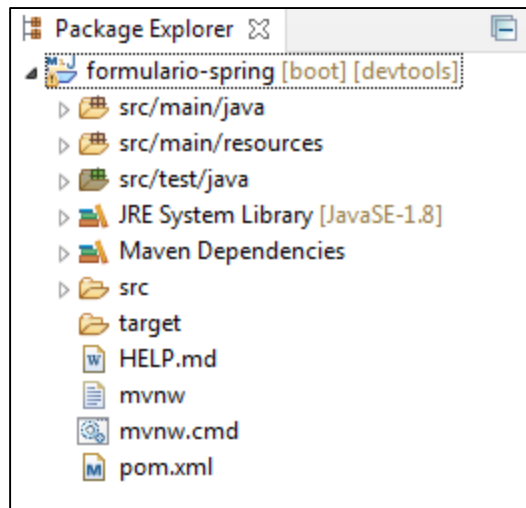
Podemos dejar por defecto los valores que nos presenta el wizard. Si se desea se puede cambiar el nombre de proyecto, el package raíz, el tipo de proyecto (Maven o Gradle) y/o la versión de Java.



# 1. CREACION PROYECTO

**Paso 2)** Agregamos las siguientes dependencias:

- Spring Web (necesaria)
- Spring Boot Dev Tools (muy importante ya que cualquier cambio que hagamos en nuestro código java, de forma automática se va a actualizar en el despliegue sin tener que reiniciar el servidor)
- Thymeleaf, para hacer uso de estas plantillas, que hacen el papel de las típicas jsp



# 1. CREACION PROYECTO

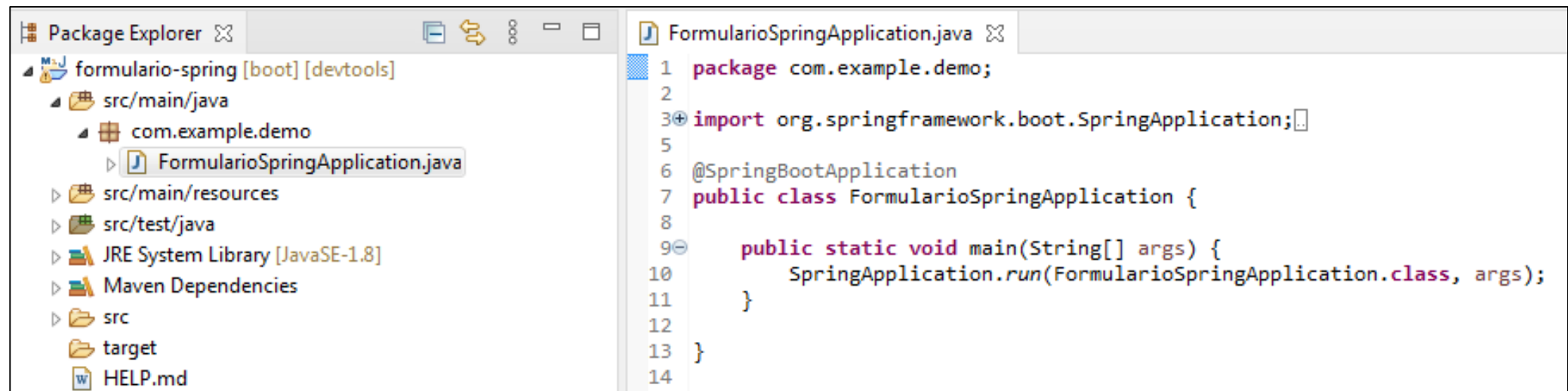
**Paso 3)** Probamos de ejecutar el proyecto, para ello levantamos el servidor Tomcat haciendo Run As/Spring Boot App. Una vez vemos que ha arrancado correctamente el servidor, vamos a un navegador y ponemos **localhost:8080**. Nos da error porque no tenemos ninguna página de inicio. Pero también significa que ya hay un servidor respondiendo en el puerto 8080.

The screenshot illustrates the process of running a Spring Boot application and the resulting error. It is divided into three main sections:

- IDE Run Menu:** On the left, the 'Run As' menu is open, showing options like 'Import...', 'Export...', 'Refresh', 'Close Project', 'Close Unrelated Projects', 'Assign Working Sets...', and 'Run As'. The 'Run As' option is selected, and a sub-menu is visible with 'Spring Boot App' (highlighted with the keyboard shortcut 'Alt+Shift+X') and 'Run Configurations...'. Other options in the sub-menu include '7 Maven clean', '8 Maven generate-sources', '9 Maven install', and 'Maven test'.
- Terminal Window:** On the right, a terminal window displays the output of the Spring Boot application. It shows the Spring Boot logo and version (v2.4.1), followed by a series of log messages indicating the application is starting successfully on port 8080. The logs include timestamps, log levels (INFO), and the names of the classes and methods being executed.
- Web Browser:** At the bottom, a web browser window is open to 'localhost:8080/'. The browser displays a 'Whitelabel Error Page' with the message: 'This application has no explicit mapping for /error, so you are seeing this as a fallback.' The timestamp 'Thu Jan 07 08:42:02 CET 2021' and the error details 'There was an unexpected error (type=Not Found, status=404)' are also visible.

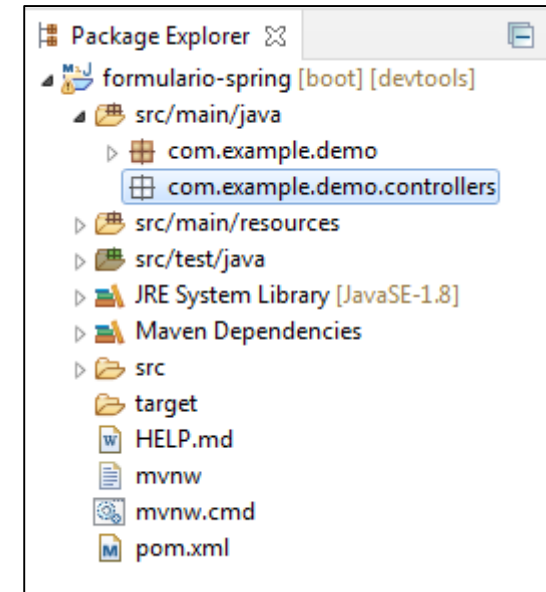
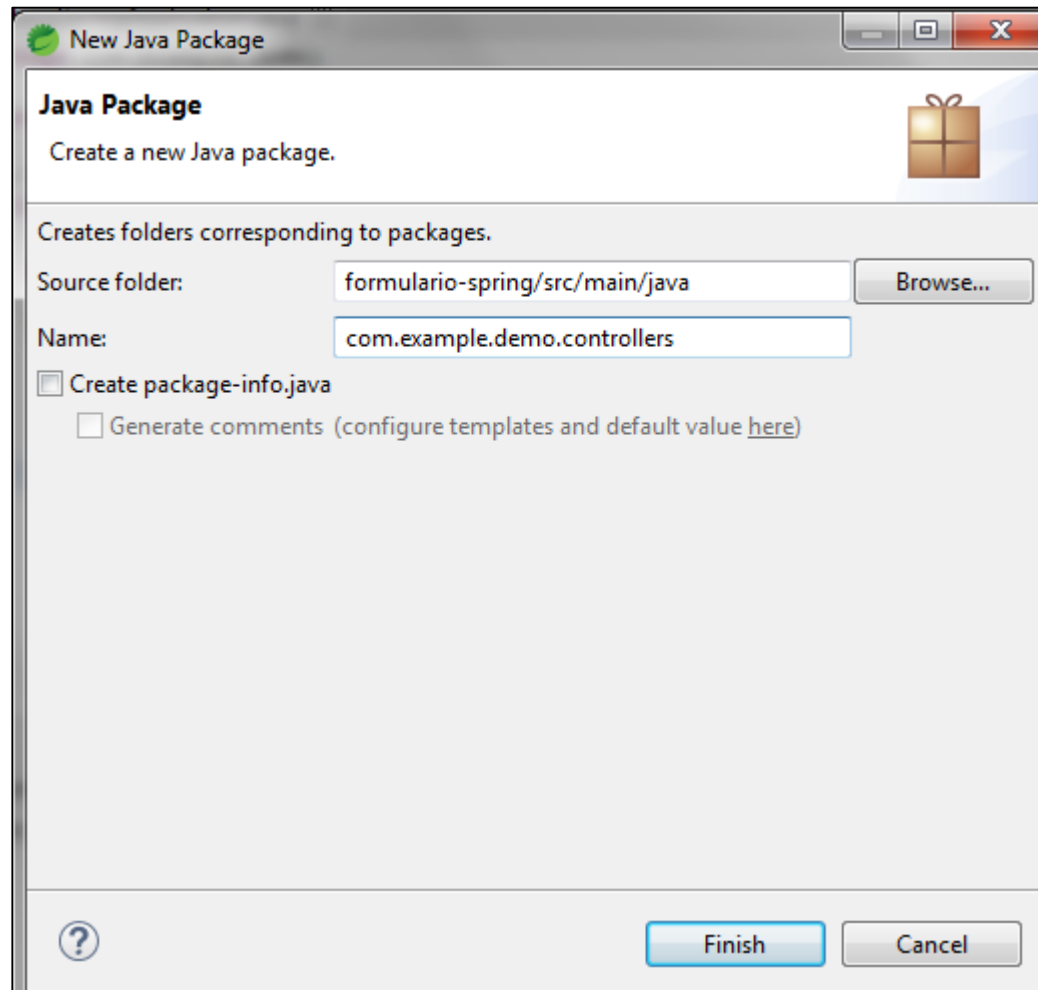
# 1. CREACION PROYECTO

**Paso 4)** Podemos observar en el package raíz indicado al principio en la creación del proyecto, la clase generada automáticamente que inicia nuestro servidor y la aplicación:



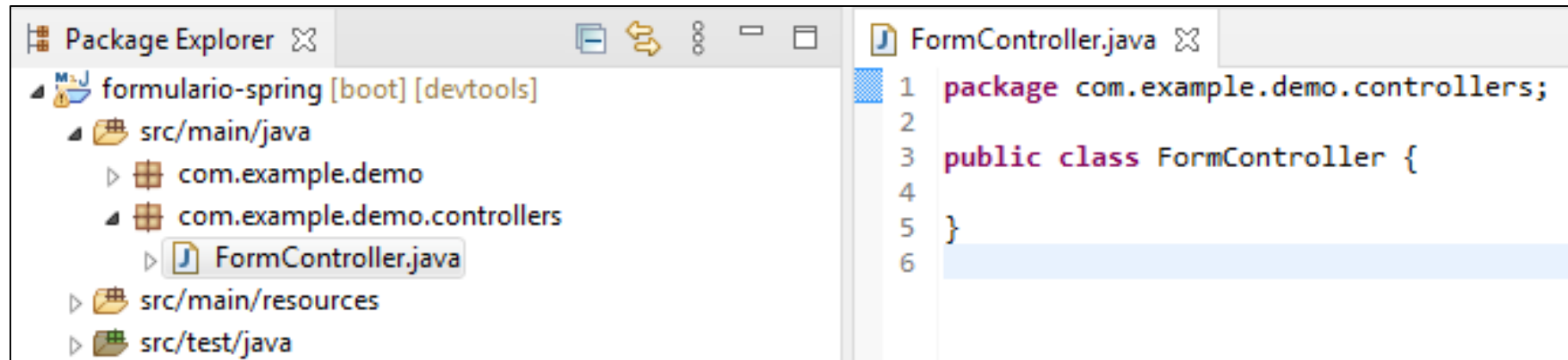
## 2. CONTROLADOR

**Paso 1)** Generamos un package dentro del existente con la extensión controllers:



## 2. CONTROLADOR

**Paso 2)** Dentro de este package creamos una clase a la que le pondremos la etiqueta de controlador.





## 2. CONTROLADOR

**Paso 3)** Creamos 2 métodos handlers, uno que muestra el formulario en pantalla (con método Get) y otro para procesarlo (con método Post).

- Ambos métodos están en la raíz del proyecto
- Cada método redirigirá hacia un fichero html: formulario.html y resultado.html

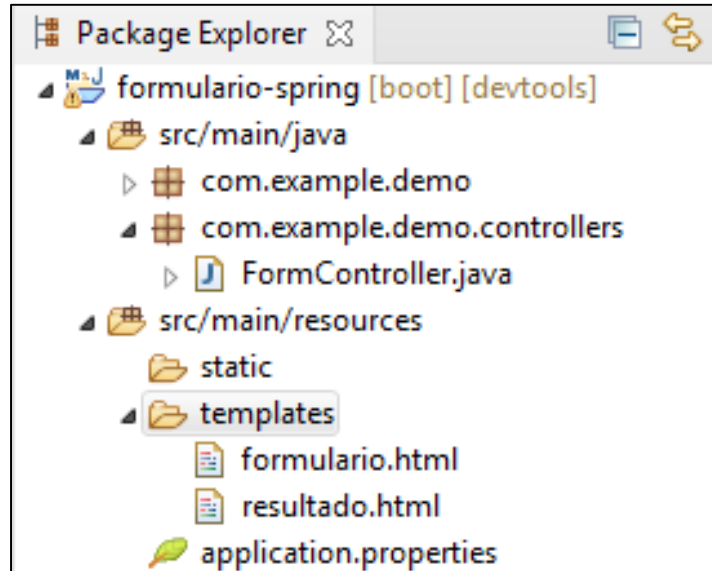
```
FormController.java
1 package com.example.demo.controllers;
2
3 import org.springframework.stereotype.Controller;
4
5 @Controller
6 public class FormController {
7
8     @GetMapping("/")
9     public String form(Model model) {
10         return "formulario";
11     }
12
13     @PostMapping("/")
14     public String procesar(Model model) {
15         return "resultado";
16     }
17
18 }
19
20
21
22
23
```

### 3. CREACION DE VISTAS

**Paso 1)** En resources/templates vamos a crear las dos plantillas necesarias:

- formulario.html
- resultado.html

Formulario.html no va a presentar cambios entre las 3 formas de paso de parámetros que vamos a presentar



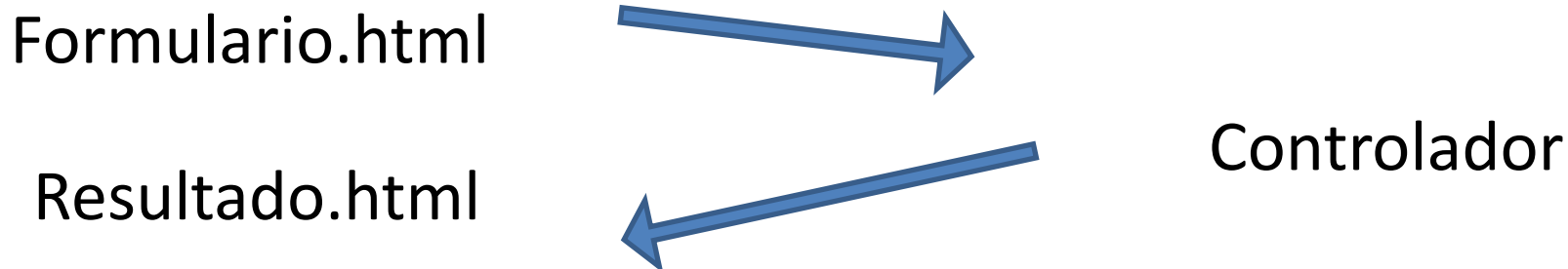
```
formulario.html
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title th:text="${titulo}"></title>
6 </head>
7 <body>
8     <h3 th:text="${titulo}"></h3>
9     <form th:action="@{/}" method="post">
10         Username<input type="text" name="username" id="username">
11         Password<input type="password" name="password" id="password">
12         Correo<input type="text" name="email" id="email">
13         <input type="submit" value="Enviar">
14     </form>
15 </body>
16 </html>
```

## 4. PASO DE PARAMETROS

---

**Paso 1)** Veremos 3 métodos de paso de parámetros y el paso de un ArrayList y despliegue de un html mediante un foreach:

- 1º método: Los datos del formularios son capturados de forma individual por el controlador y este los envía de la misma forma a la vista.
- 2º método: Los datos del formulario son capturados de forma individual por el controlador y éste los envía de forma unificada dentro de una clase a la vista.
- 3º método: Los datos del formulario son capturados y reenviados por el controlador de forma unificada dentro de una clase.



## 4. PRIMER METODO PASO PARAMETROS

**Paso 2)** En el método del controlador que procesa el formulario, insertaremos tantos parámetros con la etiqueta `@RequestParam`, como elementos html (básicamente input type) hayamos definido en formulario.html.

```
<form th:action="@{/}" method="post">
    Username<input type="text" name="username" id="username">
    Password<input type="password" name="password" id="password">
    Correo<input type="text" name="email" id="email">
    <input type="submit" value="Enviar">
</form>
```

Es muy importante que coincida el valor del atributo name del input type con el nombre de la variable usada para recuperar su valor en el controlador.

```
FormController.java
1 package com.example.demo.controllers;
2
3 import org.springframework.stereotype.Controller;
4
5
6 @Controller
7 public class FormController {
8
9     @GetMapping("/")
10    public String form(Model model) {
11        model.addAttribute("titulo", "Envio formulario");
12        return "formulario";
13    }
14
15    @PostMapping("/")
16    public String procesar(Model model,
17        @RequestParam(name="username") String username,
18        @RequestParam String password,
19        @RequestParam String email) {
20
21        model.addAttribute("titulo", "Resultado formulario");
22        model.addAttribute("username", username);
23        model.addAttribute("password", password);
24        model.addAttribute("email", email);
25        return "resultado";
26    }
27 }
28
29
30 }
```

## 4. PRIMER METODO PASO PARAMETROS

**Paso 3)** Los valores capturados por el controlador son empaquetados en la estructura Model (una especie de HashMap, compuesta de clave y valor) y de esta forma son enviados a la plantilla resultado.html.

```
FormController.java
1 package com.example.demo.controllers;
2
3 import org.springframework.stereotype.Controller;
4
5
6
7
8
9 @Controller
10 public class FormController {
11
12     @GetMapping("/")
13     public String form(Model model) {
14         model.addAttribute("titulo", "Envio formulario");
15         return "formulario";
16     }
17
18     @PostMapping("/")
19     public String procesar(Model model,
20         @RequestParam (name="username") String username,
21         @RequestParam String password,
22         @RequestParam String email) {
23
24         model.addAttribute("titulo", "Resultado formulario");
25         model.addAttribute("username", username);
26         model.addAttribute("password", password);
27         model.addAttribute("email", email);
28         return "resultado";
29     }
30 }
31
```

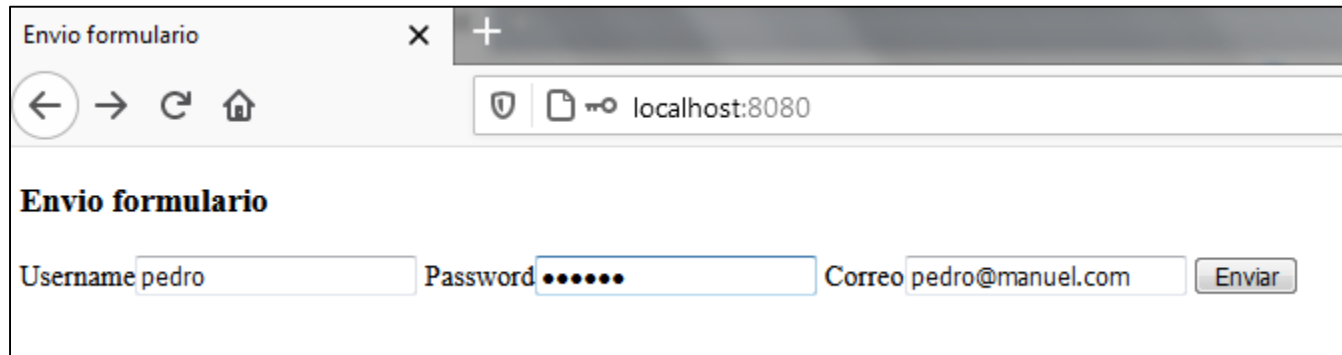
## 4. PRIMER METODO PASO PARAMETROS

**Paso 4)** El archivo resultado.html debe recoger y mostrar las 3 variables que le pasa el controlador mediante la anotacion de thymeleaf

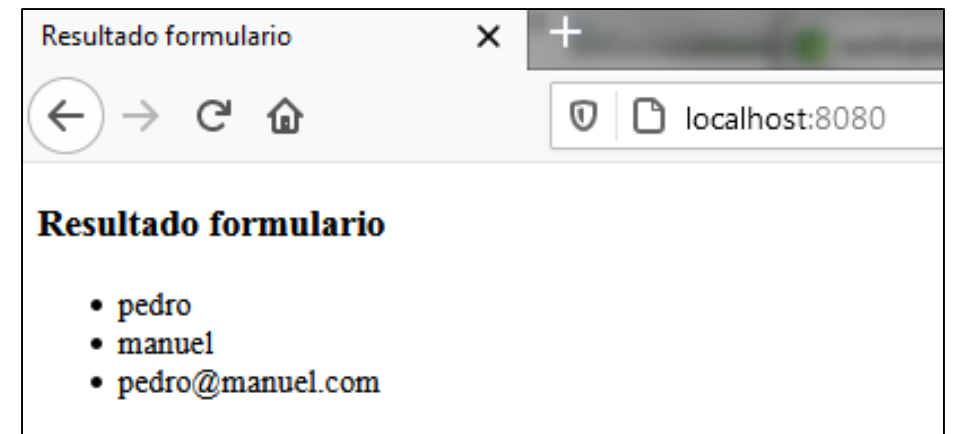
```
resultado.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="ISO-8859-1">
5     <title th:text="${titulo}"></title>
6 </head>
7 <body>
8     <h3 th:text="${titulo}"></h3>
9     <ul>
10         <li th:text="${username}"></li>
11         <li th:text="${password}"></li>
12         <li th:text="${email}"></li>
13     </ul>
14 </body>
15 </html>
```

## 4. PRIMER METODO PASO PARAMETROS

**Paso 5)** Ejecutamos el proyecto y vemos el funcionamiento típico de un formulario jee:



A screenshot of a web browser window with the title 'Envio formulario'. The address bar shows 'localhost:8080'. The page content includes a heading 'Envio formulario' and a form with three input fields: 'Username' containing 'pedro', 'Password' containing seven dots, and 'Correo' containing 'pedro@manuel.com'. An 'Enviar' button is located to the right of the 'Correo' field.



A screenshot of a web browser window with the title 'Resultado formulario'. The address bar shows 'localhost:8080'. The page content includes a heading 'Resultado formulario' and a bulleted list of the submitted data:

- pedro
- manuel
- pedro@manuel.com



## 4. SEGUNDO METODO PASO PARAMETROS

**Paso 1)** Ahora creamos una clase POJO o Bean del formulario: Usuario. Debe tener exactamente los mismos atributos que los valores name del fomrulario

```
<form th:action="@{/}" method="post">
  Username<input type="text" name="username" id="username">
  Password<input type="password" name="password" id="password">
  Correo<input type="text" name="email" id="email">
  <input type="submit" value="Enviar">
</form>
```

```
Usuario.java
1 package com.example.demo.beans;
2
3 public class Usuario {
4     private String username;
5     private String password;
6     private String email;
7
8     public String getUsername() {
9         return username;
10    }
11    public void setUsername(String username) {
12        this.username = username;
13    }
14    public String getPassword() {
15        return password;
16    }
17    public void setPassword(String password) {
18        this.password = password;
19    }
20    public String getEmail() {
21        return email;
22    }
23    public void setEmail(String email) {
24        this.email = email;
25    }
26 }
```



## 4. SEGUNDO METODO PASO PARAMETROS

**Paso 2)** En este el controlador recoge los parámetros del formulario de forma individual pero después los integra dentro de un objeto de la clase Usuario, que es el que finalmente envía la vista

```
FormController.java
1 package com.example.demo.controllers;
2
3 import org.springframework.stereotype.Controller;
10
11 @Controller
12 public class FormController {
13
14     @GetMapping("/")
15     public String form(Model model) {
16         model.addAttribute("titulo", "Envio formulario");
17         return "formulario";
18     }
19
20     @PostMapping("/")
21     public String procesar(Model model,
22         @RequestParam (name="username") String username,
23         @RequestParam String password,
24         @RequestParam String email) {
25
26         Usuario usuario = new Usuario();
27         usuario.setUsername(username);
28         usuario.setPassword(password);
29         usuario.setEmail(email);
30         model.addAttribute("titulo", "Resultado formulario");
31         model.addAttribute("usuario", usuario);
32
33         //model.addAttribute("username", username);
34         //model.addAttribute("password", password);
35         //model.addAttribute("email", email);
36         return "resultado";
37     }
38 }
39
```

## 4. SEGUNDO METODO PASO PARAMETROS

**Paso 3)** Como ahora los datos pasados a resultado.html han sido encapsulados en la clase Usuario, debemos llamarlos usando la clase Usuario:

```
resultado.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="ISO-8859-1">
5     <title th:text="${titulo}"></title>
6 </head>
7 <body>
8     <h3 th:text="${titulo}"></h3>
9     <ul>
10         <li th:text="${usuario.username}"></li>
11         <li th:text="${usuario.password}"></li>
12         <li th:text="${usuario.email}"></li>
13     </ul>
14 </body>
15 </html>
```

## 4. TERCER MÉTODO PASO PARAMETROS

**Paso 1)** En esta tercera forma directamente ponemos como parámetro principal de la función procesar a un objeto de la clase usuario, indicando que los datos del formulario ya vendrán encapsulados directamente en este objeto:

Detalles importantes a tener en cuenta:

- El formulario vista no se debe de tocar (maravilla de spring, en struts se debe de tocar el <form> indicando que se va a encapsular en tal clase-bean).
- La clase usuario debe de ser una copia de los name de los atributos, si no la cosa no funciona.
- El código queda muy compacto

```
FormController.java
1 package com.example.demo.controllers;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.PostMapping;
7
8 import com.example.demo.beans.Usuario;
9
10 @Controller
11 public class FormController {
12
13     @GetMapping("/")
14     public String form(Model model) {
15         model.addAttribute("titulo", "Envio formulario");
16         return "formulario";
17     }
18
19     @PostMapping("/")
20     public String procesar(Usuario usuario, Model model) {
21
22         model.addAttribute("titulo", "Resultado formulario");
23         model.addAttribute("usuario", usuario);
24
25         return "resultado";
26     }
27 }
28
```

## 4. TERCER METODO PASO PARAMETROS

**Paso 2)** La vista resultado no se modifica y permanece exactamente igual

```
resultado.html ✕
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="ISO-8859-1">
5     <title th:text="${titulo}"></title>
6 </head>
7 <body>
8     <h3 th:text="${titulo}"></h3>
9     <ul>
10         <li th:text="${usuario.username}"></li>
11         <li th:text="${usuario.password}"></li>
12         <li th:text="${usuario.email}"></li>
13     </ul>
14 </body>
15 </html>
```