

PROGRAMACION N-CAPAS LABORATORIO #2



**Introducción a Spring Boot y Apache maven.
Configuración e implementación.
@GetMapping , @PostMapping**

Catedrático:

Lic. Juan Lozano

Instructores:

Karla Beatriz Morales Alfaro 00022516@uca.edu.sv
Sara Noemy Romero Menjivar 00030716@uca.edu.sv
Salvador Edgardo Campos Gómez 00117716@uca.edu.sv

Introducción a Spring Boot

¿Qué es Spring Boot?

Es un framework o herramienta que ayuda a construir aplicaciones basadas principalmente en microservicios de una forma más ágil.

➤ @GetMapping

Es una versión especializada de la anotación **@RequestMapping** que actúa como un acceso directo para **@RequestMapping (método = RequestMethod.GET)**.

➤ @PostMapping

Es una versión especializada de la anotación **@RequestMapping** que actúa como un acceso directo para **@RequestMapping (método = RequestMethod.POST)**

Pasos para instalar Spring Boot

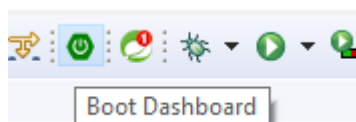
- **Spring Tools 4:** Ya trae soporte para Spring Boot.
- **Eclipse.**

1. Help > Eclipse Marketplace.
2. Buscar e instalar Spring Tools 4.



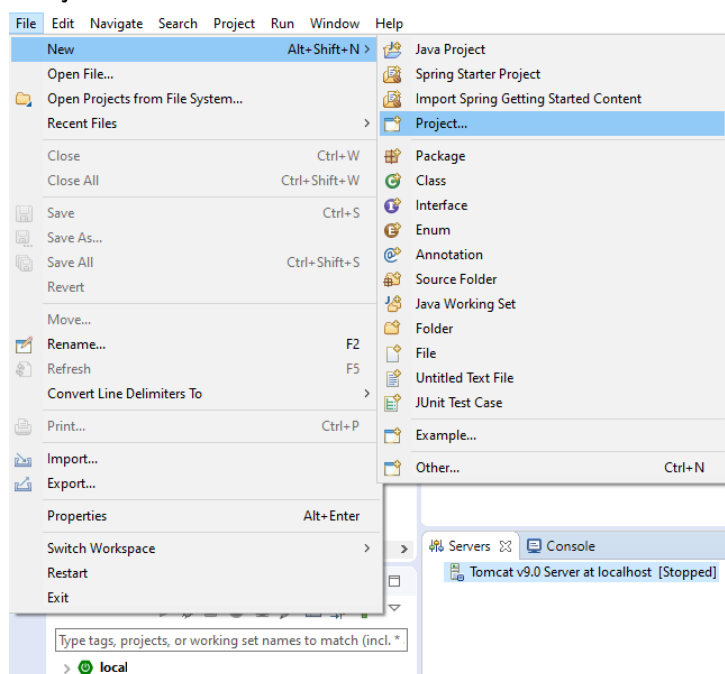
//La instalación tardará unos minutos y se deberá reiniciar Eclipse.

- Si la instalación fué exitosa en la barra del menú aparecerá un nuevo botón (Boot Dashboard).

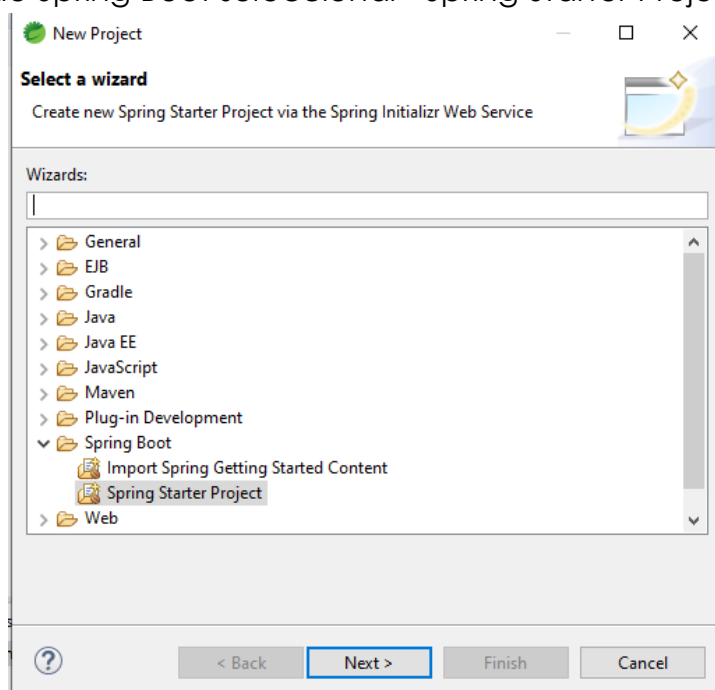


Crear un proyecto con Spring Boot en Eclipse y SpringTools4.

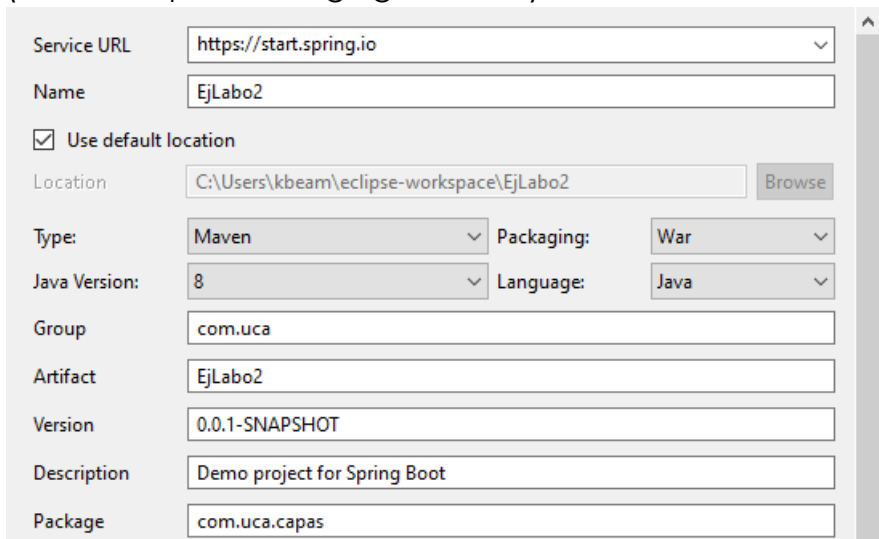
- File > New > Project...



- En la carpeta de Spring Boot seleccionar "Spring Starter Project"



3. Crear proyecto de la siguiente manera y luego dar Next:
(Verificar que Packaging sea War)



Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

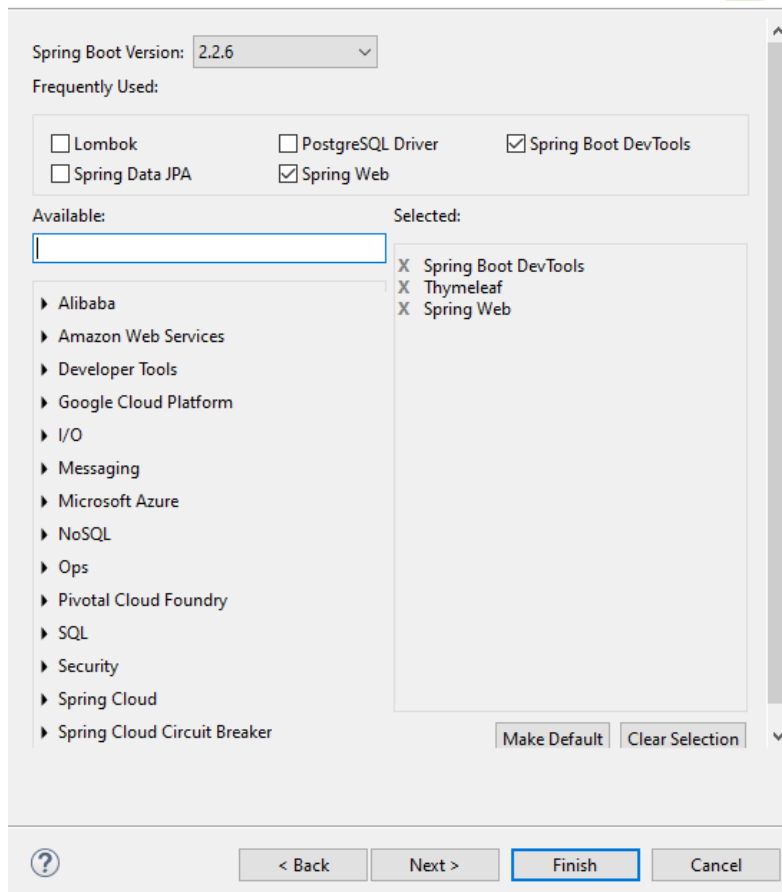
Version:

Description:

Package:

4. Agregar las dependencias (según lo que se seleccione, se crearán las dependencias en el pom.xml)

New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

☐ Lombok ☐ PostgreSQL Driver ☒ Spring Boot DevTools

☐ Spring Data JPA ☒ Spring Web

Available:

Selected:

X Spring Boot DevTools

X Thymeleaf

X Spring Web

Alibaba

Amazon Web Services

Developer Tools

Google Cloud Platform

I/O

Messaging

Microsoft Azure

NoSQL

Ops

Pivotal Cloud Foundry

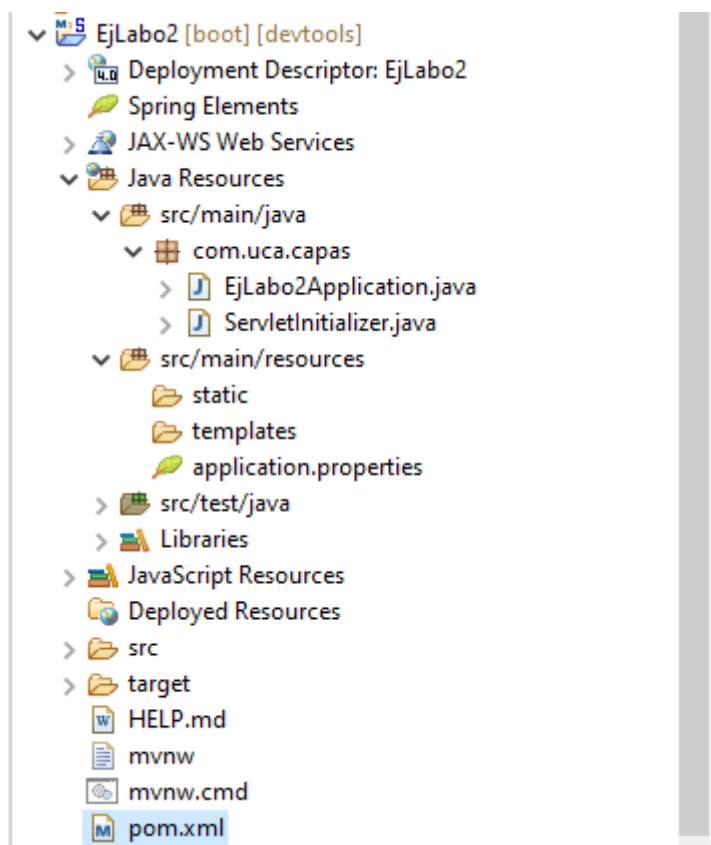
SQL

Security

Spring Cloud

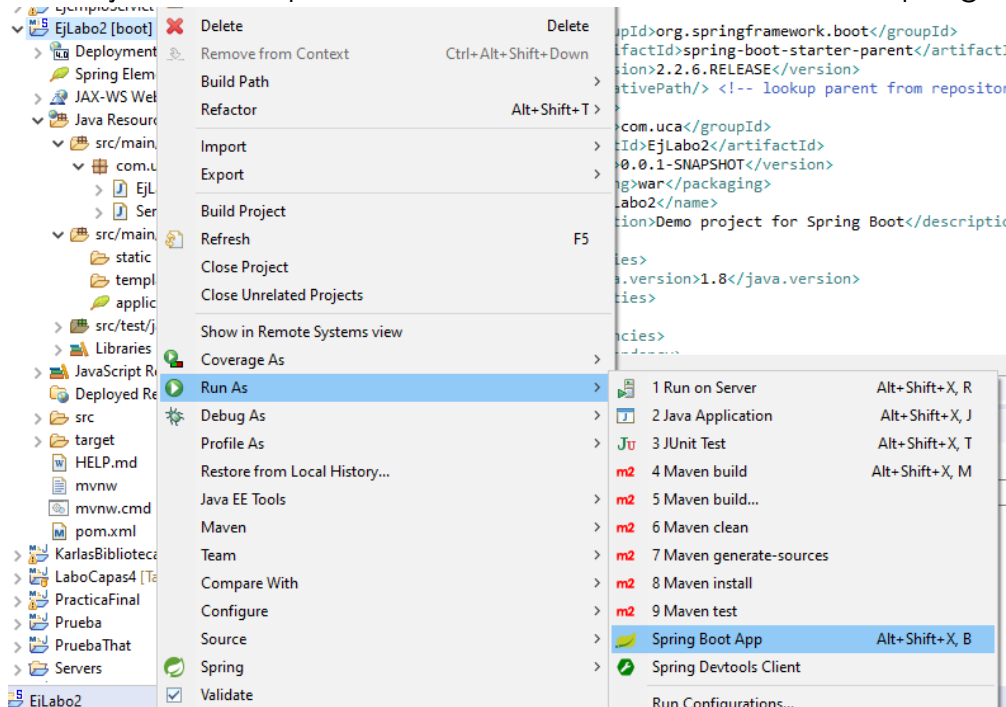
Spring Cloud Circuit Breaker

Se creará el proyecto de la siguiente manera:

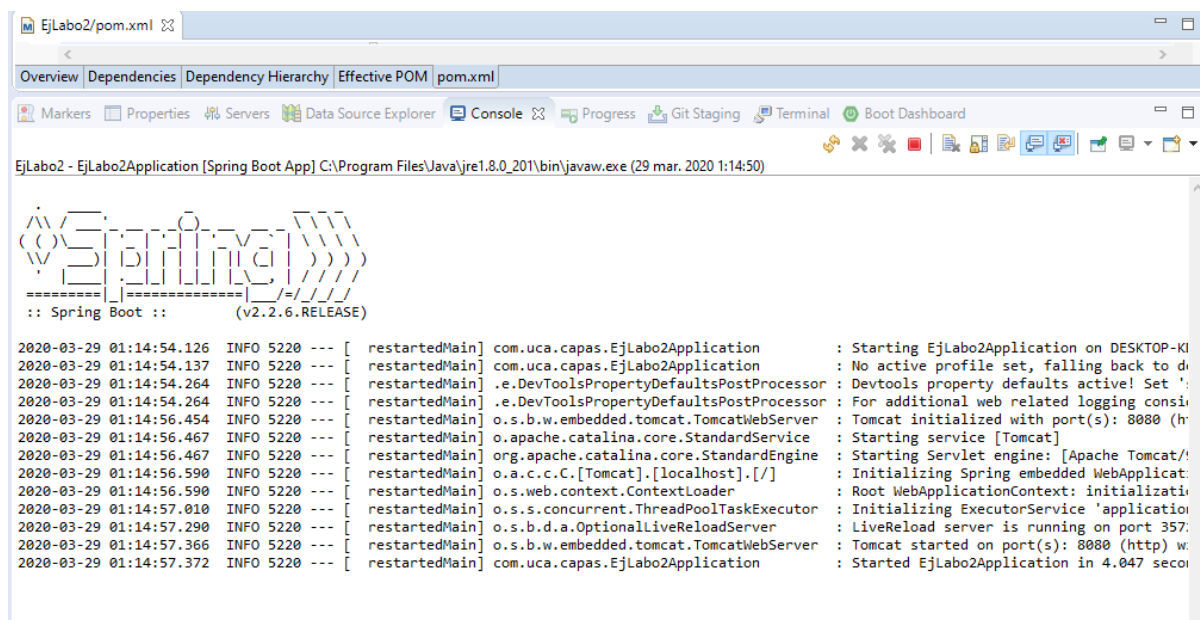


De no crearse de esta manera dar click en el botón “Boot Dashboard”

5. Para ejecutar la aplicación dar click derecho Run As... “Spring Boot App”



Si todo se ejecuta correctamente, en la consola se mostrará el siguiente mensaje:



```

EjLabo2/pom.xml
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Markers Properties Servers Data Source Explorer Console Progress Git Staging Terminal Boot Dashboard
EjLabo2 - EjLabo2Application [Spring Boot App] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (29 mar. 2020 1:14:50)

:: Spring Boot :: (v2.2.6.RELEASE)

2020-03-29 01:14:54.126 INFO 5220 --- [ restartedMain] com.uca.capas.EjLabo2Application : Starting EjLabo2Application on DESKTOP-KI
2020-03-29 01:14:54.137 INFO 5220 --- [ restartedMain] com.uca.capas.EjLabo2Application : No active profile set, falling back to d
2020-03-29 01:14:54.264 INFO 5220 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set '
2020-03-29 01:14:54.264 INFO 5220 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consi
2020-03-29 01:14:56.454 INFO 5220 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (h
2020-03-29 01:14:56.467 INFO 5220 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Starting service [Tomcat]
2020-03-29 01:14:56.467 INFO 5220 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/!
2020-03-29 01:14:56.590 INFO 5220 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicat
2020-03-29 01:14:56.590 INFO 5220 --- [ restartedMain] o.s.web.context.ContextLoader : Root WebApplicationContext: initializati
2020-03-29 01:14:57.010 INFO 5220 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'application
2020-03-29 01:14:57.290 INFO 5220 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 357
2020-03-29 01:14:57.366 INFO 5220 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) w
2020-03-29 01:14:57.372 INFO 5220 --- [ restartedMain] com.uca.capas.EjLabo2Application : Started EjLabo2Application in 4.047 secon
  
```

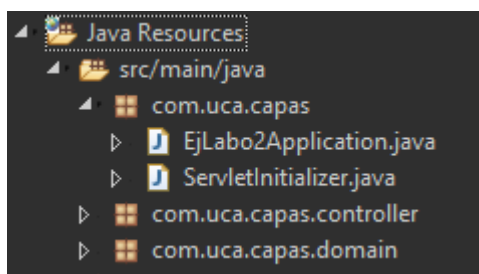
Actividad 30%:

Hacer una aplicación web con Spring Boot, en la cual, por medio de un formulario pueda ingresar los datos de un usuario de la siguiente manera:

- Nombre del usuario.
- Ocupación del usuario.

Luego, al presionar el botón “Enviar” devuelva los datos registrados del usuario. Utilizar las anotaciones **@GetMapping** y **@PostMapping**.

Teniendo implementado Spring Boot en la aplicación web, se crearán dos paquetes en `src/main/java`, el primer paquete se llamará **“com.uca.capas.controller”** y el segundo paquete se llamará **“com.uca.capas.domain”**.

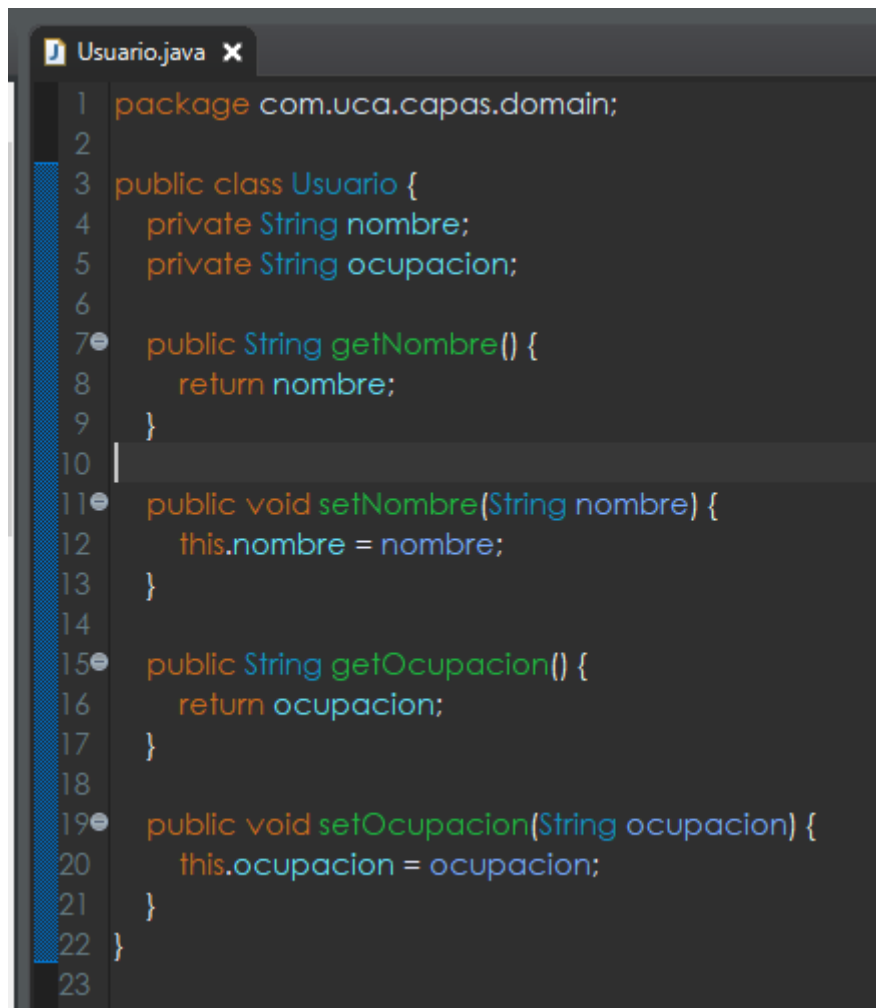


Dentro del paquete **“com.uca.capas.controller”** crear una clase llamada **“MainController.java”**

Dentro del paquete **“com.uca.capas.domain”** crear una clase llamada **“Usuario.java”**

Usuario.java

Se crea la clase donde se almacenan los datos del usuario.



```
1 package com.uca.capas.domain;
2
3 public class Usuario {
4     private String nombre;
5     private String ocupacion;
6
7     public String getNombre() {
8         return nombre;
9     }
10
11     public void setNombre(String nombre) {
12         this.nombre = nombre;
13     }
14
15     public String getOcupacion() {
16         return ocupacion;
17     }
18
19     public void setOcupacion(String ocupacion) {
20         this.ocupacion = ocupacion;
21     }
22 }
23
```

MainController.java

En la clase MainController se hará la implementación de las anotaciones **@GetMapping** y **@PostMapping**.

@GetMapping: Devuelve página agregarUsuario.html la cual contiene el formulario.

@PostMapping: Devuelve la página mostrarMensaje.html donde se muestran los datos del usuario ingresado.

```

MainController.java X
1 package com.uca.capas.controller;
2
3 import org.springframework.stereotype.Controller;
4
5
6
7
8
9 @Controller
10 public class MainController {
11
12     @GetMapping("/agregarUsuario")
13     public String enviarForm(Usuario usuario) {
14
15         return "agregarUsuario";
16     }
17
18     @PostMapping("/agregarUsuario")
19     public String procesarForm(Usuario usuario) {
20
21         return "mostrarMensaje";
22     }
23 }
24

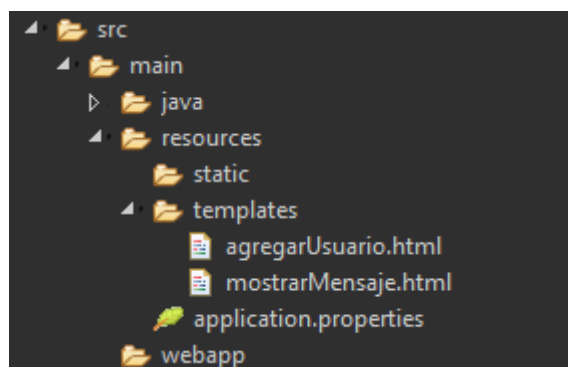
```

Dentro de la carpeta src > main > resources > templates crear:

agregarUsuario.html

Dentro de la carpeta src > main > resources > templates crear:

mostrarMensaje.html



agregarUsuario.html

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <h1>Agregar usuario</h1>
9
10 <form action="#" th:action="@{/agregarUsuario}" th:object="${usuario}" method="post">
11 <p>
12     Nombre: <input type="text" th:field="*{nombre}">
13 </p>
14 <p>
15     Ocupacion: <input type="text" th:field="*{ocupacion}">
16 </p>
17 <p>
18     <input type="submit" value="Enviar"/> <input type="reset" value="Borrar">
19 </p>
20 </form>
21 </body>
22 </html>

```

agregarUsuario.html es la vista que contiene el formulario, el th:object hace referencia al formulario de usuario. Este no es el nombre de la clase, es un identificador.

Por ejemplo:

Nombre: <input type="text" th:field="*{nombre}">

Pudo haberse referenciado por el identificador:

<p th:text="Nombre: ' + \${usuario.nombre}'"></p>

Nota:

Es importante tomar en cuenta que en **th:field** se ocupa el nombre definido en el **get()** y **set()** de la clase usuario, en la cual se toma la primera letra en minúscula.

getNombre() = nombre

mostrarMensaje.html

De la misma manera por medio del identificador “**usuario.nombre**” **getNombre()** se obtienen los datos del usuario que se ha ingresado para mostrarlos.

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <h1>Datos del usuario</h1>
9
10 <p th:text="Nombre: ' + ${usuario.nombre}"></p>
11 <p th:text="Ocupacion: ' + ${usuario.ocupacion}"></p>
12 <a href="/agregarUsuario">Regresar</a>
13 </body>
14 </html>

```

Luego se ejecutará la aplicación como un “Spring Boot App” y en buscador se pondrá lo que se definió en el @GetMapping, este en el controlador se configuro para que devuelva el **agregarUsuario.html** en el cual nos mostrará el formulario para ingresar los datos del usuario.

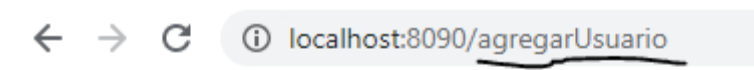
← → ↻ ⓘ localhost:8090/agregarUsuario

Agregar usuario

Nombre:

Ocupacion:

Luego al presionar “Enviar” nos mostrará los datos del usuario ingresados por medio de **@PostMapping**, de la misma manera en el controlador se configuró que @PostMapping nos devuelva el **mostrarMensaje.html** donde se muestran los datos del usuario ingresado.



Datos del usuario

Nombre: Karla Morales

Ocupacion: Analista BI

[Regresar](#)

TAREA 70%:

Crear una aplicación web con Spring boot y elaborar un login con @GetMapping y @PostMapping, dicho login debe recibir usuario y contraseña por medio de un formulario y al presionar "Enviar" debe mostrar si las credenciales son correctas o incorrectas.

Detalles:

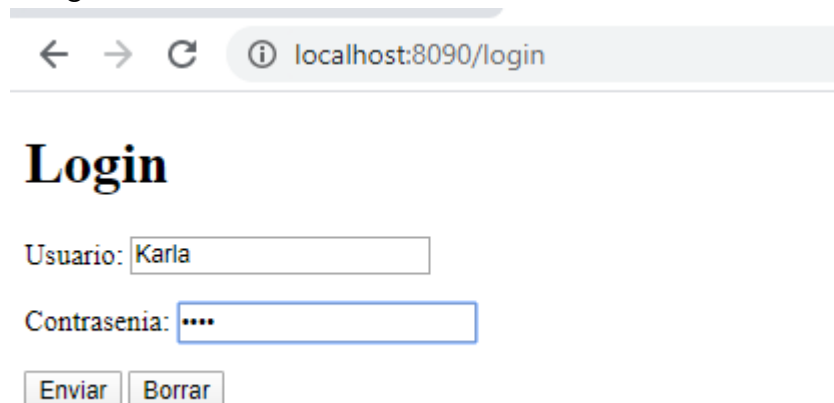
1. El **proyecto** debe ser nombrado **tareaLabo2**.
2. Crear 3 paginas html:
 - a. Login.html (En esta página debe crearse el formulario).
 - b. MostrarMensajeV.html (**En esta página debe mostrarse "credenciales correctas"**).
 - c. MostrarMensajeF.html (**En esta página debe mostrarse "credenciales incorrectas"**).
3. En el main controller configurar (**Como se muestra en las imágenes**):
 - a. "/login".
 - b. "/validacion".
4. Las credenciales pueden ir quemadas, estas dejarlas comentadas en el código.
5. Subir el proyecto a github.

ENTREGABLES:

- ✓ Actividad 30% funcional, subir el proyecto a github y adjuntar el link al entregable **Actividad Laboratorio 2**.
- ✓ Tarea 70% funcional, subir el proyecto a github y adjuntar el link a entregable **Tarea laboratorio 2**.

IMÁGENES DEMOSTRATIVAS DE TAREA:

Imagen 1.



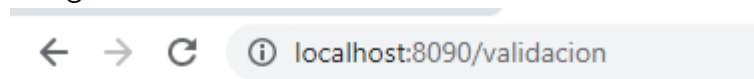
← → ↻ ⓘ localhost:8090/login

Login

Usuario:

Contraseña:

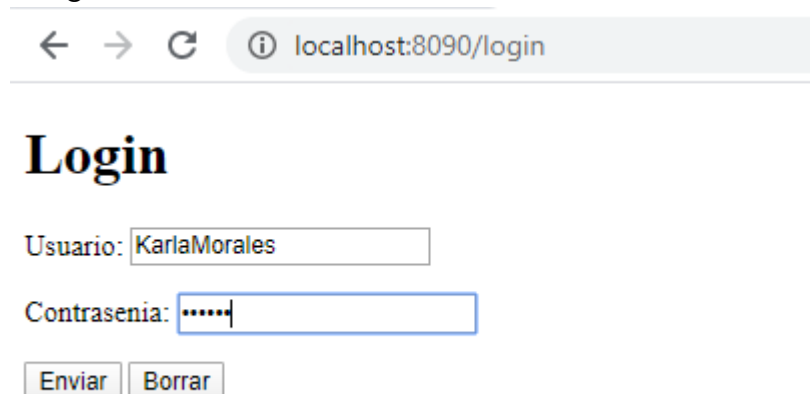
Imagen 2.



← → ↻ ⓘ localhost:8090/validacion

Credenciales incorrectas

Imagen 3.



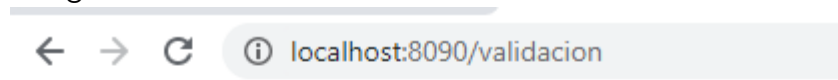
← → ↻ ⓘ localhost:8090/login

Login

Usuario:

Contraseña:

Imagen 4.



← → ↻ ⓘ localhost:8090/validacion

Credenciales correctas

Referencias:

<https://github.com/KarlaMorales97/Laboratorio2-nCapas.git>

<https://marketplace.eclipse.org/>

<https://www.arquitecturajava.com/que-es-un-java-maven-artifact/>

<https://howtodoinjava.com/spring5/webmvc/controller-getmapping-postmapping/>