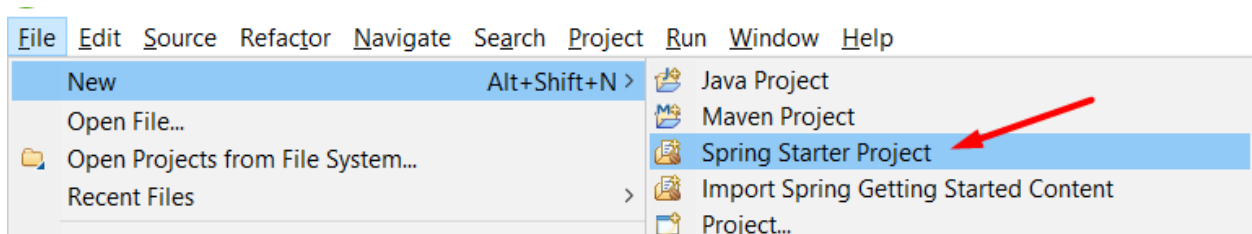


PROYECTO “singleton-persona”

Paso 1: Crear el Proyecto en STS

Abrir Spring Tool Suite (STS).

Ir a **File → New → Spring Starter Project**



Completar los datos del proyecto:

- **Name:** singleton-persona
- **Type:** Maven
- **Java Version:** 17
- **Group:** com.example
- **Artifact:** singleton-demo
- **Package:** com.example.singletonpersona
- **Packaging:** Jar

Service URL	<input type="text" value="https://start.spring.io"/>		
Name	<input type="text" value="singleton-persona-1"/>		
<input checked="" type="checkbox"/> Use default location			
Location	<input type="text" value="C:\Users\JR\Documents\workspace\singleton-persona-1"/>	<input type="button" value="Browse"/>	
Type:	<input type="text" value="Maven"/>	Packaging:	<input type="text" value="Jar"/>
Java Version:	<input type="text" value="17"/>	Language:	<input type="text" value="Java"/>
Group	<input type="text" value="com.example"/>		
Artifact	<input type="text" value="singleton-persona-1"/>		
Version	<input type="text" value="0.0.1-SNAPSHOT"/>		
Description	<input type="text" value="Demo project for Spring Boot"/>		
Package	<input type="text" value="com.example.singletonpersona"/>		

- **Dependencies:**
 - **Spring Web (Para la API REST).**

<input type="checkbox"/> H2 Database	<input type="checkbox"/> Lombok	<input type="checkbox"/> PostgreSQL Driver
<input type="checkbox"/> Spring Boot DevTools	<input type="checkbox"/> Spring Cache Abstraction	<input type="checkbox"/> Spring Data JPA
<input type="checkbox"/> Spring Data Redis (Access+I	<input checked="" type="checkbox"/> Spring Web	<input type="checkbox"/> Spring Web Services

Hacer clic en **Finish** para generar el proyecto.

Paso 2: Implementar el Patrón Singleton

Maneja la lógica y creamos el service



```
package com.example.singletonpersona.service;

public class PersonaSingleton {
    private static PersonaSingleton instance;
    private String nombre;

    private PersonaSingleton() {
        this.nombre = "Sin nombre"; // Valor inicial
    }

    public static PersonaSingleton getInstance() {
        if (instance == null) {
            instance = new PersonaSingleton();
        }
        return instance;
    }

    public String getNombre() {
        return nombre;
    }

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

Explicación:

- **private static PersonaSingleton instance:** Guarda la única instancia.
- **private PersonaSingleton():** Evita que se creen múltiples instancias.
- **getInstance():** Crea la instancia si no existe, **y siempre devuelve la misma.**
- **setNombre(nuevoNombre):** Permite cambiar el nombre en la instancia única.

Paso 3: Crear el Controlador PersonaController

Expone los endpoints REST



```
package com.example.singletonpersona.controller;

import com.example.singletonpersona.service.PersonaSingleton;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/persona")

public class PersonaController {

    @GetMapping
    public String getNombre() {
        return "Nombre actual: " + PersonaSingleton.getInstance().getNombre();
    }

    @PostMapping("/{nuevoNombre}")
    public String setNombre(@PathVariable String nuevoNombre) {
        PersonaSingleton.getInstance().setNombre(nuevoNombre);
        return "Nuevo nombre asignado: " + nuevoNombre;
    }
}
```

Explicación:

- **GET /persona:** Retorna el **nombre actual** de la única instancia.
- **POST /persona/{nuevoNombre}:** Cambia el nombre en la instancia **Singleton**.

Paso 4: Ejecutar y Probar

- **GET** http://localhost:8080/persona

Respuesta esperada



Nombre actual: Sin nombre

GET

http://localhost:8080/persona

Send

ParamsAuthorizationHeaders (6)BodyScriptsTestsSettingsCookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (5)Test Results


200 OK • 141 ms • 189 B • Save Response

RawPreviewVisualize

1 Nombre actual: Sin nombre

- **POST** http://localhost:8080/persona/Juan

Respuesta esperada



Nuevo nombre asignado: Juan

POST http://localhost:8080/persona/Juan **Send**

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK • 14 ms • 191 B Save Response

Raw Preview Visualize

1 Nuevo nombre asignado: Juan

Paso 5: Conclusión

- El patrón Singleton garantiza que haya una única instancia de Persona.
- El nombre cambia globalmente porque siempre usamos la misma instancia.
- Es ideal para casos donde necesitamos un estado global compartido.