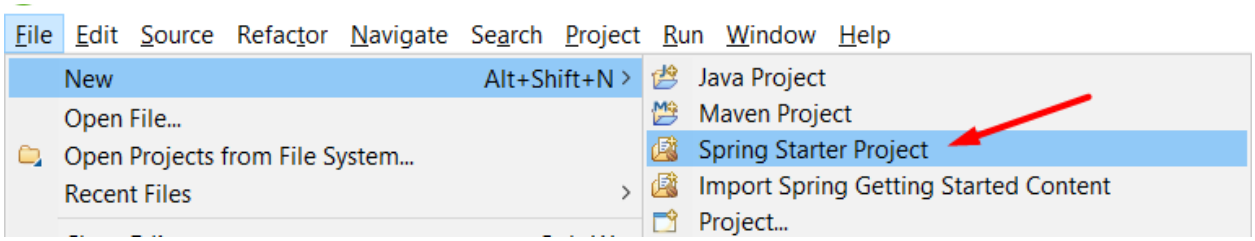
PROYECTO “singleton-demo”

Paso 1: Crear el Proyecto en STS

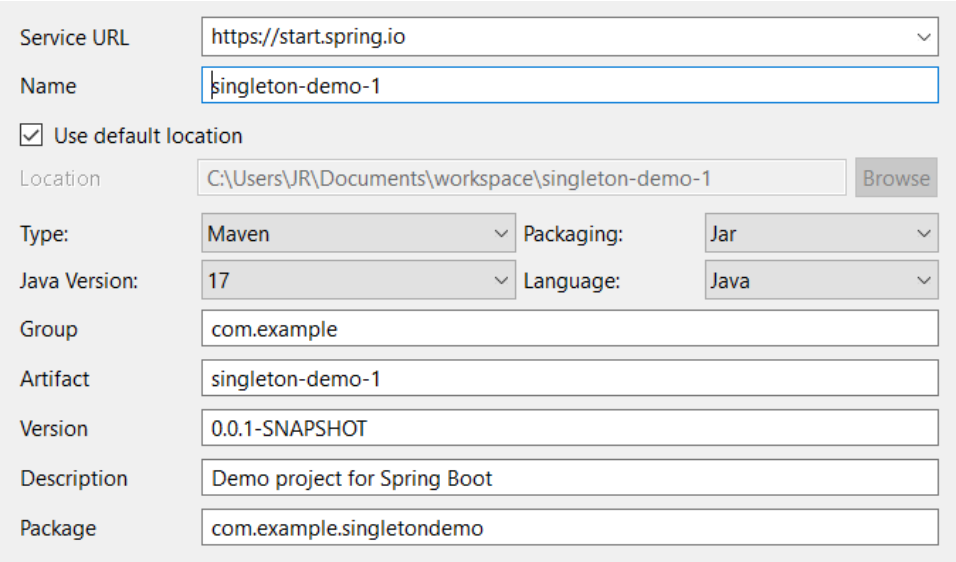
Abrir Spring Tool Suite (STS).

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

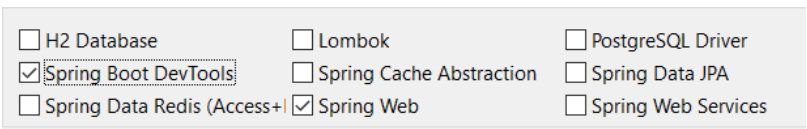


Completar los datos del proyecto:

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



- **Dependencies:**
  - **Spring Web (Para la API REST).**
  - **Spring Boot DevTools (Para recarga en vivo).**



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

Paso 2: Implementar el Patrón Singleton

Maneja la lógica y creamos el service

```java
package com.example.singletondemo.service;

public class SingletonService {
    // Instancia única de la clase
    private static SingletonService instance;

    // Constructor privado para evitar instanciación externa
    private SingletonService() {
        System.out.println("Nueva instancia creada!");
    }

    // Método estático para obtener la única instancia
    public static SingletonService getInstance() {
        if (instance == null) {
            instance = new SingletonService();
        }
        return instance;
    }

    public String getMessage() {
        return "Hola, soy un Singleton!";
    }
}
```

Explicación:

- **private static SingletonService instance:** Almacena la instancia única.
- **private SingletonService():** Evita que la clase sea instanciada con new.
- **public static SingletonService getInstance():** Crea una única instancia y la reutiliza.
- **getMessage():** Método de prueba para demostrar que siempre es la misma instancia.

Paso 3: Crear el Controlador SingletonController

Expone los endpoints REST

```java
package com.example.singletondemo.controller;

import com.example.singletondemo.service.SingletonService;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/singleton")
public class SingletonController {

    @GetMapping
    public String getSingletonMessage() {
        SingletonService singleton = SingletonService.getInstance();
        return singleton.getMessage();
    }
}
```

Explicación:

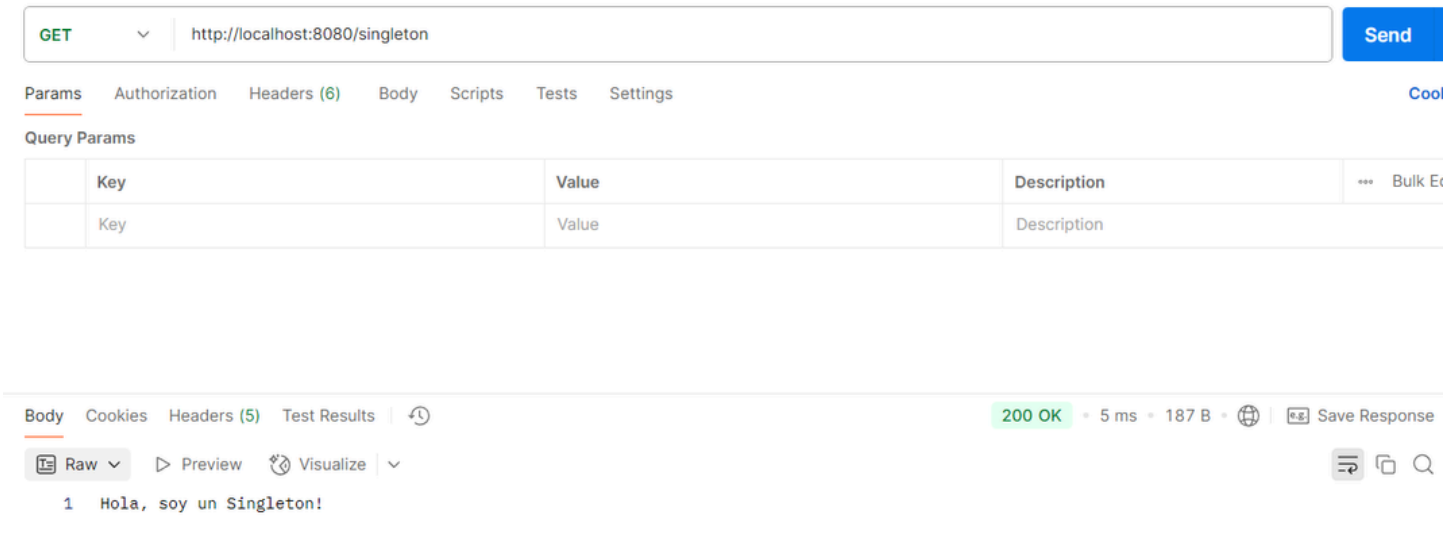
- **@RestController:** Exponemos un endpoint REST.
- **GET /singleton:** Devuelve el mensaje "Hola, soy un Singleton!".
- **Llama a SingletonService.getInstance():** Demuestra que siempre es la misma instancia.

Paso 3: Ejecutar y Probar

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

Respuesta esperada

```
Hola, soy un Singleton!
```



Paso 4: Conclusión

- Una instancia es un objeto creado a partir de una clase.
- Cada instancia tiene su propio estado y datos en memoria (excepto en el patrón Singleton).
- El Patrón Singleton evita múltiples instancias y garantiza que solo haya una.