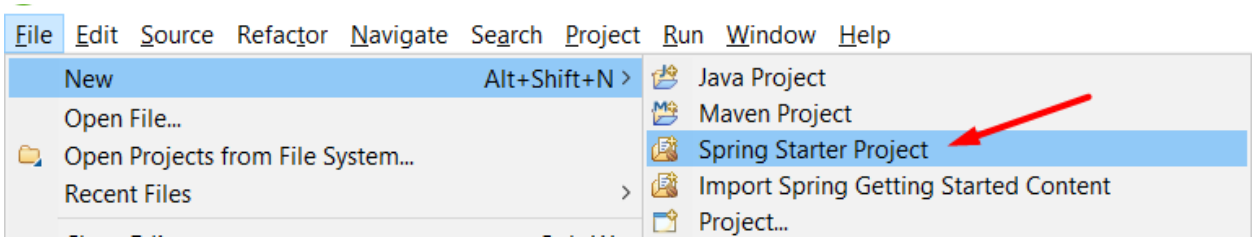
PROYECTO “builder-demo”

Paso 1: Crear el Proyecto en STS

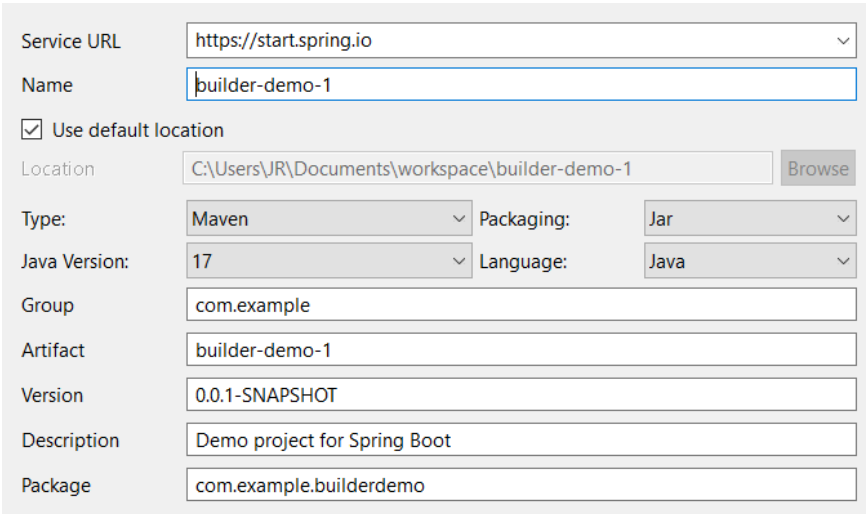
Abrir Spring Tool Suite (STS).

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

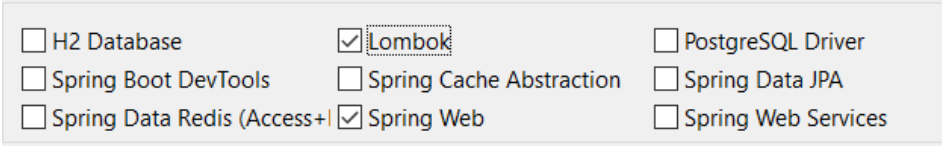


Completar los datos del proyecto:

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



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



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

Paso 2: Implementar el Patrón Builder

Creamos la clase persona en el paquete model.

```java
package com.example.builderdemo.model;

import com.fasterxml.jackson.annotation.JsonInclude;
import lombok.NoArgsConstructor;

@NoArgsConstructor
@JsonInclude(JsonInclude.Include.NON_NULL) // Ignorar valores nulos en la respuesta
public class Persona {
    private String nombre;
    private int edad;
    private String direccion;
    private String telefono;

    private Persona(PersonaBuilder builder) {
        this.nombre = builder.nombre;
        this.edad = builder.edad;
        this.direccion = builder.direccion;
        this.telefono = builder.telefono;
    }

    // Getters (necesarios para convertir JSON a objeto)
    public String getNombre() { return nombre; }
    public int getEdad() { return edad; }
    public String getDireccion() { return direccion; }
    public String getTelefono() { return telefono; }

    // Clase estática interna Builder
    public static class PersonaBuilder {
        private String nombre;
        private int edad;
        private String direccion;
        private String telefono;

        public PersonaBuilder setNombre(String nombre) {
            this.nombre = nombre;
            return this;
        }

        public PersonaBuilder setEdad(int edad) {
            this.edad = edad;
            return this;
        }

        public PersonaBuilder setDireccion(String direccion) {
            this.direccion = direccion;
            return this;
        }

        public PersonaBuilder setTelefono(String telefono) {
```

```
            this.telefono = telefono;
            return this;
        }

        public Persona build() {
            return new Persona(this);
        }
    }
}
```

Explicación:

Anotaciones para JSON (@JsonInclude, @NoArgsConstructor)

- @JsonInclude(JsonInclude.Include.NON_NULL): Evita que valores nulos aparezcan en el JSON de respuesta.
- @NoArgsConstructor: Agrega un constructor sin argumentos, permitiendo que se pueda instanciar Persona.

Constructor privado (private Persona(PersonaBuilder builder))

- Evita que Persona sea instanciada directamente con new Persona().
- **Obliga a usar el PersonaBuilder** para crear objetos.

Métodos Getter (getNombre(), getEdad(), etc.)

- Son necesarios para acceder a los valores de Persona.

Clase Interna PersonaBuilder (Patrón Builder)

- Contiene métodos setNombre(), setEdad(), etc., para **construir el objeto paso a paso**.
- build(): Devuelve la instancia final de Persona.

Paso 3: Crear el service PersonaService

Creamos el service

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

import com.example.builderdemo.model.Persona;
import org.springframework.stereotype.Service;

@Service
public class PersonaService {
    public Persona crearPersona(String nombre, int edad, String direccion, String telefono) {
        return new Persona.PersonaBuilder()
                .setNombre(nombre)
                .setEdad(edad)
                .setDireccion(direccion)
                .setTelefono(telefono)
                .build();
    }
}
```

Explicación:

- **@Service:** Indica que esta clase es un **servicio** en Spring Boot.

Método crearPersona()

- Usa el **Builder** para **crear una instancia de Persona paso a paso**.
- Retorna el objeto Persona **ya construido**.

Paso 4: Crear el controller PersonaController

Creamos el controller

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

import com.example.builderdemo.model.Persona;
import com.example.builderdemo.service.PersonaService;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/persona")
public class PersonaController {
    private final PersonaService personaService;

    public PersonaController(PersonaService personaService) {
        this.personaService = personaService;
    }

    @PostMapping
    public Persona crearPersona(@RequestBody Persona persona) {
        return personaService.crearPersona(
            persona.getNombre(),
            persona.getEdad(),
            persona.getDireccion(),
            persona.getTelefono()
        );
    }
}
```

Explicación:

- **@RestController:** Indica que es un controlador REST en Spring Boot.
- **@RequestMapping("/persona"):** Define el endpoint base /persona.

Método crearPersona(@RequestBody Persona persona)

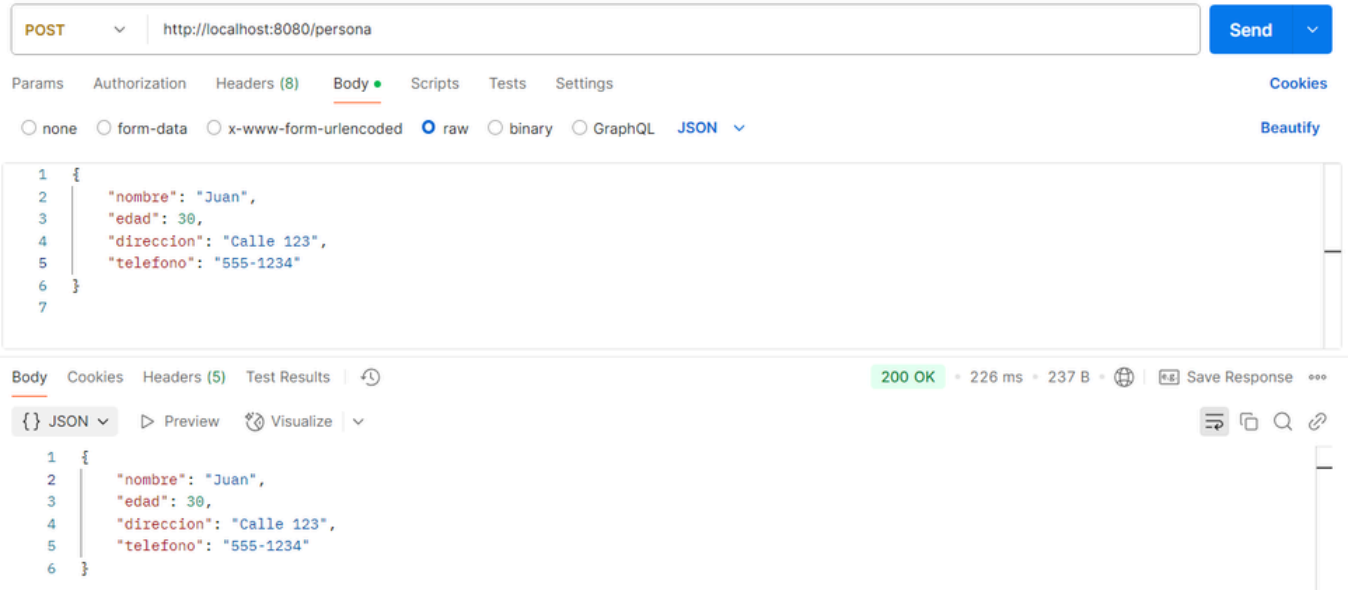
- Recibe un **JSON con datos de una persona**.
- Llama a PersonaService para **crear una persona usando el Builder**.
- Devuelve el objeto Persona construido.

Paso 5: Ejecutar y Probar

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

Respuesta esperada

```
{ "nombre": "Juan", "edad": 30, "direccion": "Calle 123", "telefono": "555-1234" }
```



Paso 6: Conclusión

Componente	Propósito
Persona	Clase con **Patrón Builder** para construir objetos fácilmente.
PersonaBuilder	Permite construir una Persona paso a paso.
PersonaService	Lógica de negocio para crear personas usando el Builder.
PersonaController	Expone un **endpoint REST** (POST /persona) para recibir datos en JSON.