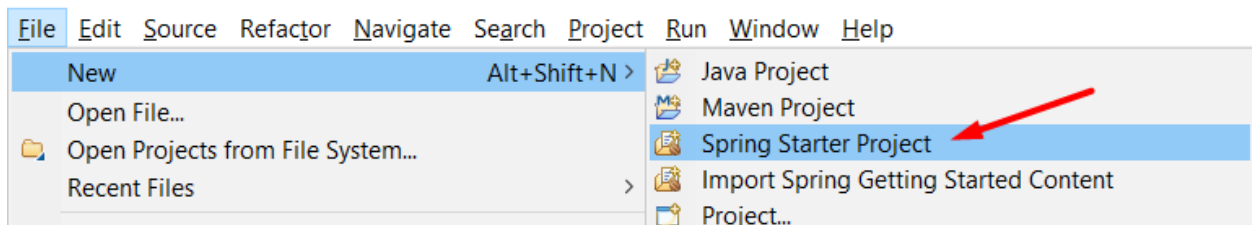


PROYECTO “factory method”

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:** factory-method-demo
- **Type:** Maven
- **Java Version:** 17
- **Group:** com.example
- **Artifact:** factory-method-demo
- **Package:** com.example.factorymethoddemo
- **Packaging:** Jar

A screenshot of the 'New Spring Starter Project' dialog box in STS. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' is 'factory-method-demo-1'. The 'Use default location' checkbox is checked. The 'Location' is 'C:\Users\JR\Documents\workspace\factory-method-demo-1'. The 'Type' is 'Maven', 'Packaging' is 'Jar', 'Java Version' is '17', and 'Language' is 'Java'. The 'Group' is 'com.example', 'Artifact' is 'factory-method-demo-1', 'Version' is '0.0.1-SNAPSHOT', 'Description' is 'Demo project for Spring Boot', and 'Package' is 'com.example.factorymethoddemo'.

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

A screenshot of the 'Dependencies' section in the STS dialog box. It shows a list of checkboxes for various dependencies: H2 Database, Spring Boot DevTools, Spring Data Redis (Access+), Lombok, Spring Cache Abstraction, Spring Web, PostgreSQL Driver, Spring Data JPA, and Spring Web Services. The 'Spring Web' checkbox is checked and highlighted with a dashed border.

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

Paso 2: Crear la Interfaz Vehículo

Creamos la clase vehículo en el paquete model.



```
package com.example.factorymethoddemo.model;

public interface Vehiculo {
    String conducir();
}
```

Explicación:

- **Define una interfaz Vehiculo** que será implementada por los diferentes tipos de vehículos.
- Obliga a que todos los vehículos implementen el **método conducir()**.

Paso 3: Crear las Clases Coche y Moto

Creamos la clase Coche



```
package com.example.factorymethoddemo.model;

public class Coche implements Vehiculo {
    @Override
    public String conducir() {
        return "Conduciendo un coche";
    }
}
```

Creamos la clase Moto



```
package com.example.factorymethoddemo.model;

public class Moto implements Vehiculo {
    @Override
    public String conducir() {
        return "Conduciendo una moto";
    }
}
```

Explicación:

- Ambas clases implementan Vehículo, pero personalizan el método conducir().
- Podemos agregar más clases (Camión, Bicicleta, etc.) sin modificar el código existente.

Paso 4: Crear la Clase VehiculoFactory (Factory Method)

Creamos la clase para el patrón Factory Method llamada VehiculoFactory



```
package com.example.factorymethoddemo.factory;

import com.example.factorymethoddemo.model.Coche;
import com.example.factorymethoddemo.model.Moto;
import com.example.factorymethoddemo.model.Vehiculo;

public class VehiculoFactory {

    public static Vehiculo crearVehiculo(String tipo) {
        if (tipo.equalsIgnoreCase("coche")) {
            return new Coche();
        } else if (tipo.equalsIgnoreCase("moto")) {
            return new Moto();
        } else {
            throw new IllegalArgumentException("Tipo de vehículo no soportado: " + tipo);
        }
    }
}
```

Explicación:

- Usamos if-else para la lógica.
- Crea una instancia de Coche o Moto según el tipo.
- Si el tipo no es reconocido, lanza una IllegalArgumentException.

Paso 5: Crear un Controlador REST VehiculoController

Creamos la clase para el patrón Factory Method llamada VehiculoFactory



```
package com.example.factorymethoddemo.controller;

import com.example.factorymethoddemo.factory.VehiculoFactory;
import com.example.factorymethoddemo.model.Vehiculo;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/vehiculo")
public class VehiculoController {

    @GetMapping("/{tipo}")
    public String obtenerVehiculo(@PathVariable String tipo) {
        Vehiculo vehiculo = VehiculoFactory.crearVehiculo(tipo);
        return vehiculo.conducir();
    }
}
```

Explicación:

- Exponemos un endpoint REST /vehiculo/{tipo}.
- Usamos VehiculoFactory para crear el vehículo correspondiente.
- Retornamos la respuesta del método conducir().

Paso 6: Ejecutar y Probar

- **GET** http://localhost:8080/vehiculo/coche

Respuesta esperada



Conduciendo un coche

GET

http://localhost:8080/vehiculo/coche

Send

Params

Authorization

Headers (6)

Body

Scripts

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK

128 ms

184 B

Save Response

Raw

Preview

Visualize

1 Conduciendo un coche