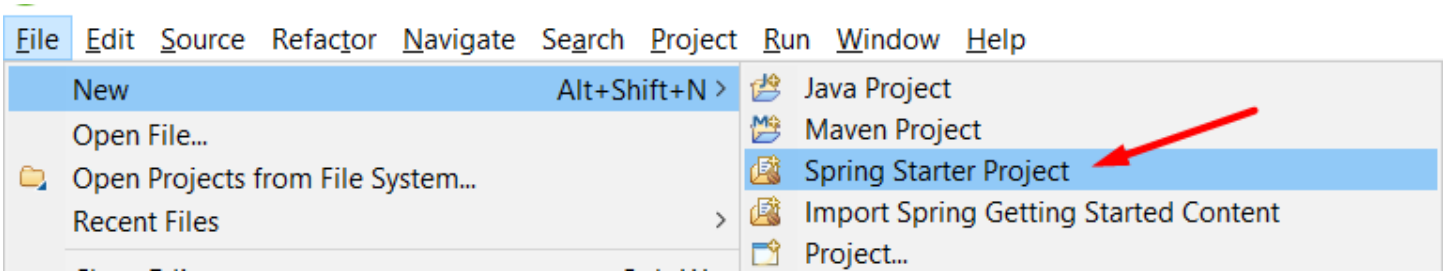
PROYECTO “SOAP-básico”

Paso 1: Crear el Proyecto en STS

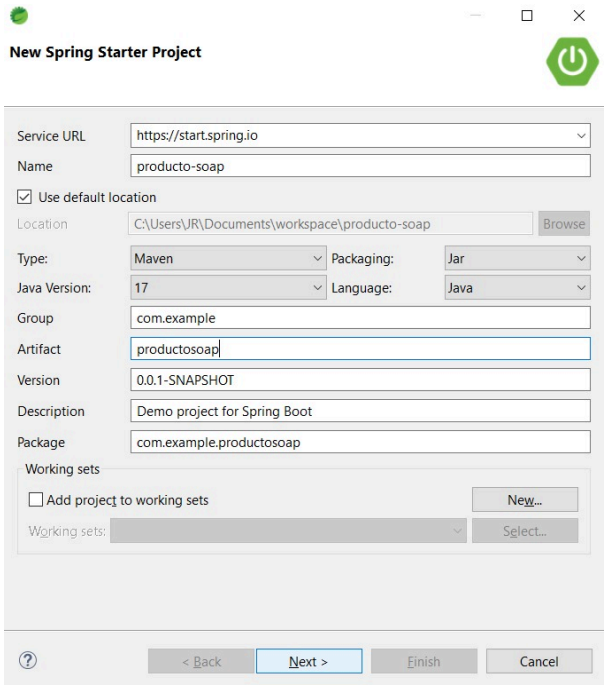
Abrir Spring Tool Suite (STS).

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

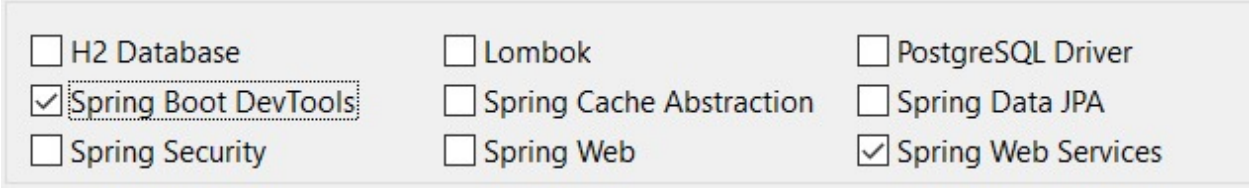


Completar los datos del proyecto:

- **Name:** producto-soap
- **Type:** Maven
- **Java Version:** 17
- **Packaging:** Jar



- **Dependencies:**
  - **Spring Web Services** (Soporte para servicios web)
  - **Spring Boot DevTools** (recarga automáticamente el proyecto ante un cambio)



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

Paso 2: Configuramos el pom

Pasamos a configurar el pom con la versión y librerías que necesitamos de la siguiente manera:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.5</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>productosoap</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>producto-soap</name>
    <description>Demo project for Spring Boot</description>
    <url />
    <licenses>
        <license />
    </licenses>
    <developers>
        <developer />
    </developers>
    <scm>
        <connection />
        <developerConnection />
        <tag />
        <url />
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web-services</artifactId>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>wsdl4j</groupId>
            <artifactId>wsdl4j</artifactId>
        </dependency>
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.3.1</version>
        </dependency>
        <dependency>
      <groupId>org.glassfish.jaxb</groupId>
      <artifactId>jaxb-runtime</artifactId>
</dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
</dependencies>

<build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>jaxb2-maven-plugin</artifactId>
                <version>2.5.0</version>
                <executions>
                    <execution>
                        <id>xjc</id>
                        <goals>
```

```
                        <goal>xjc</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <sources>
<source>${project.basedir}/src/main/resources/producto.xsd</source>
                </sources>
                <outputDirectory>${project.basedir}/src/main/java</outputDirectory>
                <clearOutputDir>false</clearOutputDir>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>
```

Configuración Principal:

- **Version**: Usaremos la versión padre **2.7.5**
- **Java Version**: usaremos el jdk 17

Dependencias Clave:

- **spring-boot-starter-web-services:** Soporte para servicios web SOAP
- **spring-boot-devtools:** Herramientas de desarrollo
- **wsdl4j:** Para trabajar con WSDL
- **jaxb-api y jaxb-runtime:** Para el binding XML-Java (marshalling/unmarshalling)
- **spring-boot-starter-test:** Para pruebas

Plugins Importantes:

- **spring-boot-maven-plugin:** Para empaquetar la aplicación Spring Boot
- **jaxb2-maven-plugin:** Para generar clases Java a partir del XSD**:**
  - Toma producto.xsd como fuente
  - Genera las clases en src/main/java

Paso 3: Creamos el esquema Producto.xsd

Creamos el siguiente esquema XML que define la estructura de los mensajes SOAP en la siguiente ruta src/main/resources/producto.xsd y quedaría de esta forma:

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://ejemplo.com/productos"
        xmlns:tns="http://ejemplo.com/productos"
        elementFormDefault="qualified">

    <xs:element name="getProductoRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="id" type="xs:int"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="getProductoResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="nombre" type="xs:string"/>
                <xs:element name="precio" type="xs:double"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Características Principales:

- **Target Namespace:** http://ejemplo.com/productos

Elementos Definidos:

- **getProductoRequest:**
  - Contiene un elemento id de tipo entero
- **getProductoResponse:**
  - Contiene nombre (string) y precio (double)

Paso 4: Creamos la configuración del SOAP

Ahora, vamos a configurar el SOAP en la ruta src/main/java/com/example/productosoap/config con el nombre de clase java: WebServiceConfig.java.

```java
package com.example.productosoap.config;

import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.ws.config.annotation.EnableWs;
import org.springframework.ws.config.annotation.WsConfigurerAdapter;
import org.springframework.ws.transport.http.MessageDispatcherServlet;
import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;

@Configuration
@EnableWs
public class WebServiceConfig extends WsConfigurerAdapter {

    @Bean
    public ServletRegistrationBean<MessageDispatcherServlet> messageDispatcherServlet(ApplicationContext ctx) {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(ctx);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean<>(servlet, "/ws/*");
    }

    @Bean(name = "productos")
    public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema productoSchema) {
        DefaultWsdl11Definition wsdl = new DefaultWsdl11Definition();
        wsdl.setPortTypeName("ProductoPort");
        wsdl.setLocationUri("/ws");
        wsdl.setTargetNamespace("http://ejemplo.com/productos");
        wsdl.setSchema(productoSchema);
        return wsdl;
    }

    @Bean
    public XsdSchema productoSchema() {
        return new SimpleXsdSchema(new ClassPathResource("producto.xsd"));
    }
}
```

Anotaciones:

- **@Configuration:** Clase de configuración Spring
- **@EnableWs:** Habilita soporte para servicios web SOAP

Beans Configurados:

1. **MessageDispatcherServlet**:
   - Maneja peticiones SOAP
   - Mapeado a "/ws/*"
   - Configurado para transformar ubicaciones WSDL
2. **DefaultWsdl11Definition**:
   - Genera automáticamente el WSDL
   - Puerto: "ProductoPort"
   - Ubicación: "/ws"
   - Namespace: "http://ejemplo.com/productos"
   - Usa el esquema producto.xsd
3. **XsdSchema**:
   - Carga el esquema XSD desde classpath ("producto.xsd")

Paso 5: Creamos el endpoint

Ahora, vamos a crear el a consumir del SOAP en la ruta src/main/java/com/example/productosoap/endpoint con el nombre de clase java: ProductoEndpoint.java.

```java
package com.example.productosoap.endpoint;

import org.springframework.ws.server.endpoint.annotation.Endpoint;
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
import org.springframework.ws.server.endpoint.annotation.RequestPayload;
import org.springframework.ws.server.endpoint.annotation.ResponsePayload;

import com.ejemplo.productos.GetProductoRequest;
import com.ejemplo.productos.GetProductoResponse;

@Endpoint
public class ProductoEndpoint {

    private static final String NAMESPACE_URI = "http://ejemplo.com/productos";

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "getProductoRequest")
    @ResponsePayload
    public GetProductoResponse getProducto(@RequestPayload GetProductoRequest request) {
        GetProductoResponse response = new GetProductoResponse();

        // Simulación: si id = 1, retorna producto ejemplo
        if (request.getId() == 1) {
            response.setNombre("Laptop");
            response.setPrecio(999.99);
        } else {
            response.setNombre("Producto no encontrado");
            response.setPrecio(0.0);
        }

        return response;
    }
}
```

Anotaciones Clave:

- **@Endpoint:** Marca la clase como un endpoint SOAP
- **@PayloadRoot:** Define qué mensaje manejar (basado en namespace y localPart)
- **@RequestPayload:** Indica que el parámetro viene en el cuerpo SOAP
- **@ResponsePayload:** Indica que el retorno va en el cuerpo SOAP

Lógica del Servicio:

- Recibe un GetProductoRequest con un ID
- Si el ID es 1, retorna un producto de ejemplo ("Laptop", 999.99)
- Para otros IDs, retorna "Producto no encontrado" con precio 0.0

Paso 6: Consumimos o probamos el endpoint

Ahora, con la ruta http://localhost:8080/ws, la escribes en el SOAP UI, agregas el Id 1 en:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:prod="http://ejemplo.com/productos">
  <soapenv:Header/>
  <soapenv:Body>
    <prod:getProductoRequest>
      <prod:id>1</prod:id>
    </prod:getProductoRequest>
  </soapenv:Body>
</soapenv:Envelope>
```