**Creating a SOAP Web Service with Spring Boot Starter Web Services**

Jul 2, 202019 minute read

This guide will help you create a SOAP Web Service with Spring Boot Starter Web Services. We will take a Contract First approach by defining an XSD and exposing a WSDL from it.

## You will learn

1. What is a web service?
2. What are the different types of web services?
3. What is SOAP Web Service?
4. What is SOAP?
5. What is a SOAP Envelope?
6. What is SOAP Header and SOAP Body?
7. What is WSDL (Web Service Definition Language)?
8. What are the different parts of a WSDL?
9. What is Contract First Approach?
10. What is an XSD?
11. What is JAXB?
12. How do you configure a JAXB Plugin?
13. What is an Endpoint?
14. Can you show an example endpoint written with Spring Web Services?
15. What is a MessageDispatcherServlet?
16. How do you configure a MessageDispatcherServlet?
17. How do you generate a WSDL using Spring Web Services?

# What is a Web Service?

Service delivered over the web

Is this really a complete definition? Is everything that delivered over the web “Web Service”?

The key things to understand is

* Web services are designed for machine-to-machine (or application-to-application) interaction
* Web services should be interoperable - Not platform dependent
* Web services should allow communication over a network

# Types of Web Services

Not really types but a broad classification

* SOAP
* REST

These are not really mutually exclusive. Some SOAP services can actually be RESTful.

# What is SOAP?

SOAP was earlier an abbreviation for Simple Object Access Protocol. In SOAP, the request and response are in XML format. However, not all types of XML are valid SOAP Requests.

SOAP defines a standard XML format. We will use WSDL (Web Service Definition Language) to define the format of request xml and the response xml.

Now let’s say Facebook wants to know how to call the TODO Service? What should I give to the Facebook developer?

I will give him a WSDL of the Todo service where the following things will be explained

1. What are the different services (operations) exposed by the server?
2. How can a service (operation) be called? What URL to use? (Also called End Point).
3. What should the structure of request xml?
4. What should be the structure of response xml?

**SOAP format defines a SOAP-Envelope which envelopes (contains) the entire document.**

* **SOAP-Header** (**optional**) contains any information needed to identify the request. Also, part of the Header is authentication, authorization information (signatures, encrypted information etc.).
* **SOAP-Body** contains the real xml content of request or response.
* In case of error response, server responds back with **SOAP-Fault**.

**REST vs SOAP**

REST vs SOAP are not really comparable.

REST is an architectural style. SOAP is a message exchange format.

Let’s compare the popular implementations of REST and SOAP styles.

* RESTful Sample Implementation : JSON over HTTP
* SOAP Sample Implementation : XML over SOAP over HTTP

**Following are the important things to consider:**

* REST is built over simple HTTP protocol. SOAP services are more complex to implement and more complex to consume.
* REST has better performance and scalability. REST reads can be cached, SOAP based reads cannot be cached.
* REST permits many different data formats (JSON is the most popular choice) whereas SOAP only permits XML.
* SOAP services have well defined structure and interface (WSDL) and have a set of well-defined standards (WS-Security, WS-AtomicTransaction and WS-ReliableMessaging). Documentation standards with REST are evolving (We will use Swagger in this course).

**SOAP Service Examples**

|  |
| --- |
| Request <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">  <Body>  <getCourseDetailsRequest xmlns="http://in28minutes.com/courses">  <id>Course1</id>  </getCourseDetailsRequest>  </Body>  </Envelope> |
| Response <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  <SOAP-ENV:Header/>  <SOAP-ENV:Body>  <ns2:getCourseDetailsResponse xmlns:ns2="http://in28minutes.com/courses">  <ns2:course>  <ns2:id>Course1</ns2:id>  <ns2:name>Spring</ns2:name>  <ns2:description>10 Steps</ns2:description>  </ns2:course>  </ns2:getCourseDetailsResponse>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |
| Fault <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  <SOAP-ENV:Header/>  <SOAP-ENV:Body>  <SOAP-ENV:Fault>  <faultcode>SOAP-ENV:Server</faultcode>  <faultstring xml:lang="en">java.lang.NullPointerException</faultstring>  </SOAP-ENV:Fault>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |

**WSDL:** WSDL is used to define the structure of Request and the Structure of Response.

View-source: http://localhost:8080/ws/courses.wsdl

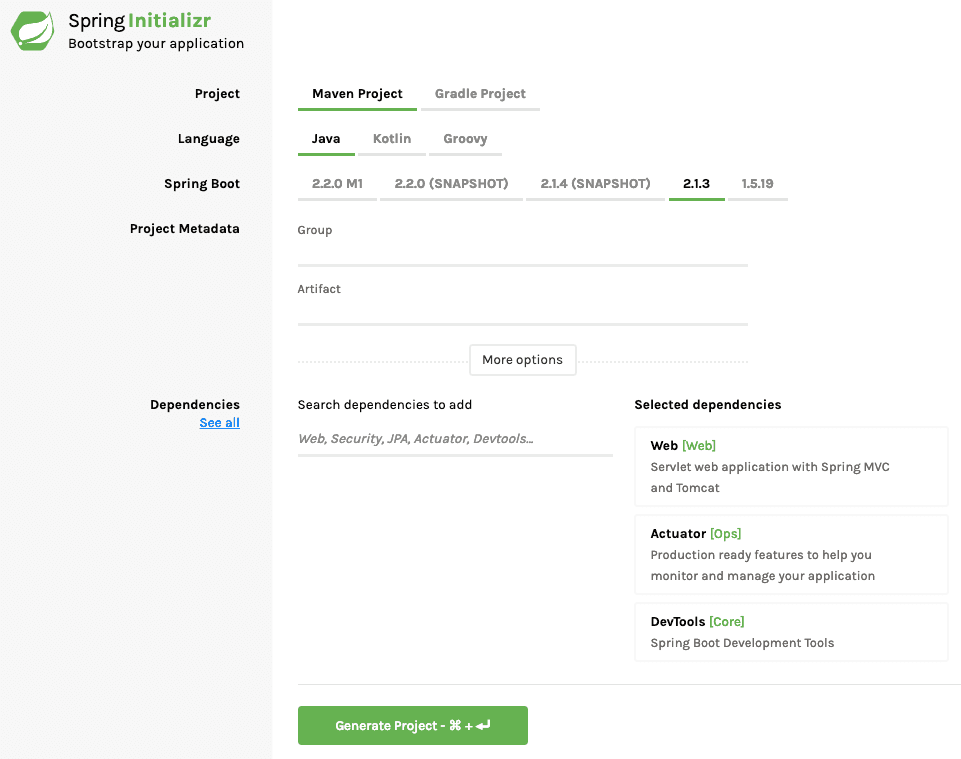
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="no"?><wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="http://in28minutes.com/courses" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://in28minutes.com/courses" targetNamespace="http://in28minutes.com/courses">  <wsdl:types>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://in28minutes.com/courses">  ---------------------------Request Body-----------------------------------------  <xs:element name="getCourseDetailsRequest">  <xs:complexType>  <xs:sequence>  <xs:element name="id" type="xs:string"/>  </xs:sequence>  </xs:complexType>  </xs:element>  ---------------------------Response Body-----------------------------------------  <xs:element name="getCourseDetailsResponse">  <xs:complexType>  <xs:sequence>  <xs:element name="course" type="tns:course"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:complexType name="course">  <xs:sequence>  <xs:element name="id" type="xs:string"/>  <xs:element name="name" type="xs:string"/>  <xs:element name="description" type="xs:string"/>  </xs:sequence>  </xs:complexType>  </xs:schema>  </wsdl:types>    <wsdl:message name="getCourseDetailsRequest">  <wsdl:part element="tns:getCourseDetailsRequest" name="getCourseDetailsRequest">  </wsdl:part>  </wsdl:message>  <wsdl:message name="getCourseDetailsResponse">  <wsdl:part element="tns:getCourseDetailsResponse" name="getCourseDetailsResponse">  </wsdl:part>  </wsdl:message>  ---------------------------SOAP Operation-----------------------------------------  <wsdl:portType name="CoursesPort">  <wsdl:operation name="getCourseDetails">  <wsdl:input message="tns:getCourseDetailsRequest" name="getCourseDetailsRequest">  </wsdl:input>  <wsdl:output message="tns:getCourseDetailsResponse" name="getCourseDetailsResponse">  </wsdl:output>  </wsdl:operation>  </wsdl:portType>  <wsdl:binding name="CoursesPortSoap11" type="tns:CoursesPort">  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>  <wsdl:operation name="getCourseDetails">  <soap:operation soapAction=""/>  <wsdl:input name="getCourseDetailsRequest">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="getCourseDetailsResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  </wsdl:binding>  <wsdl:service name="CoursesPortService">  <wsdl:port binding="tns:CoursesPortSoap11" name="CoursesPortSoap11">  <soap:address location="http://localhost:8080/ws"/>  </wsdl:port>  </wsdl:service>  </wsdl:definitions> |

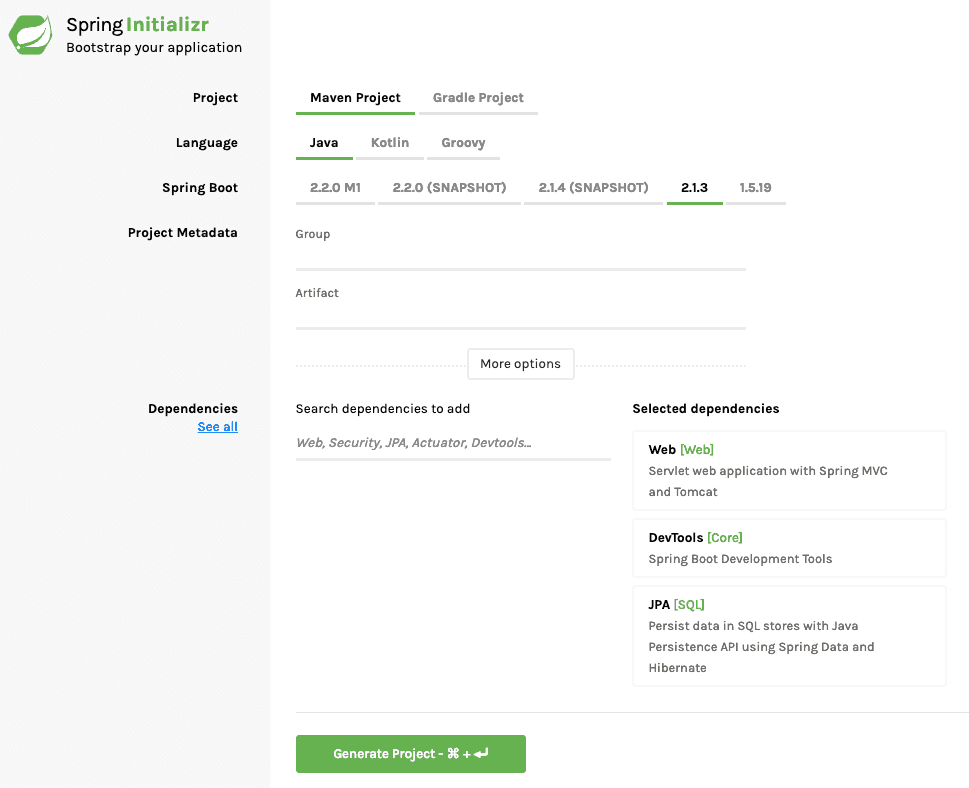
## Bootstrapping with Spring Initializer(Project creation)

Creating a SOAP Web service with Spring Initializr is a cake walk.

Spring Initializr [http://start.spring.io/](https://start.spring.io/) is great tool to bootstrap your Spring Boot projects.

You can create a wide variety of projects using Spring Initializr.





**Following steps have to be done for a Web Services project**

* Launch Spring Initializr and choose the following
  + Choose com.in28minutes.springboot.soap.web.services.example as Group
  + Choose spring-boot-tutorial-soap-web-services as Artifact
  + Choose following dependencies
    - Web Services
    - DevTools
* Click Generate Project.
* Import the project into Eclipse. File -> Import -> Existing Maven Project.

Do not forget to add Web Services as a dependency.

**Creating a SOAP Web Service with Spring Boot**

We will use a contract first approach and first define the XSD for the request and response.

**Define XSD for Request and Response**

/src/main/resources/student-details.xsd

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://in28minutes.com/students"  xmlns:tns="http://in28minutes.com/students" elementFormDefault="qualified">    <xs:element name="GetStudentDetailsRequest">  <xs:complexType>  <xs:sequence>  <xs:element name= "id" type="xs:int"/>  </xs:sequence>  </xs:complexType>  </xs:element>    <xs:element name="GetStudentDetailsResponse">  <xs:complexType>  <xs:sequence>  <xs:element name= "StudentDetails" type="tns:StudentDetails"/>  </xs:sequence>  </xs:complexType>  </xs:element>    <xs:complexType name="StudentDetails">  <xs:sequence>  <xs:element name="id" type="xs:int"/>  <xs:element name="name" type="xs:string"/>  <xs:element name="passportNumber" type="xs:string"/>  </xs:sequence>  </xs:complexType>    </xs:schema> |

We are creating a simple xsd defining the request GetStudentDetailsRequest and the response GetStudentDetailsResponse.

**Example request and response are shown below**

|  |
| --- |
| Request  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">  <Body>  <GetStudentDetailsRequest xmlns="http://in28minutes.com/students">  <id>1</id>  </GetStudentDetailsRequest>  </Body>  </Envelope> |
| Response  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  <SOAP-ENV:Header/>  <SOAP-ENV:Body>  <ns2:GetStudentDetailsResponse xmlns:ns2="http://in28minutes.com/students">  <ns2:StudentDetails>  <ns2:id>1</ns2:id>  <ns2:name>Adam</ns2:name>  <ns2:passportNumber>E1234567</ns2:passportNumber>  </ns2:StudentDetails>  </ns2:GetStudentDetailsResponse>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |

### Java API for XML Binding (JAXB) and Configuring JAXB 2 Maven Plugin

When we implement our code using Spring Web Services, following are the steps that are typically involved in processing a request

* Map Request XML to Java Request Objects
* Do the business logic and create the Java Response Objects
* Map the Response Object to a Response XML and return the response.

We do mapping from XML to Java and Java to XML. This is done using JAXB - Java API for XML Binding.

A Maven JAXB Plugin helps us in generating the Java objects based on the XSD. Let’s add it to our pom.xml.

|  |
| --- |
| <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>jaxb2-maven-plugin</artifactId>  <version>1.6</version>  <executions>  <execution>  <id>xjc</id>  <goals>  <goal>xjc</goal>  </goals>  </execution>  </executions>  <configuration>  <schemaDirectory>${project.basedir}/src/main/resources</schemaDirectory>  <outputDirectory>${project.basedir}/src/main/java</outputDirectory>  <clearOutputDir>false</clearOutputDir>  </configuration>  </plugin> |

Three important configurations

* <schemaDirectory>${project.basedir}/src/main/resources</schemaDirectory> - The location of XSD files.
* <outputDirectory>${project.basedir}/src/main/java</outputDirectory> - **Where do you want your Java code to be generated to?**
* <clearOutputDir>false</clearOutputDir> - **Should the output directory be cleaned every time? We use false because we write our java source code in the same directory.**

**Configuring an Endpoint for GetCourseDetailsRequest**

Endpoint is the component that receives the request, initiates the processing and sends the response back.

Let’s first create a bean for storing the Student details.

/src/main/java/com/in28minutes/springboot/soap/web/services/example/student/Student.java

|  |
| --- |
| package com.in28minutes.springboot.soap.web.services.example.student;  public class Student {  private Long id;  private String name;  private String passportNumber;  public Student() {  super();  }  public Student(Long id, String name, String passportNumber) {  super();  this.id = id;  this.name = name;  this.passportNumber = passportNumber;  }  public Student(String name, String passportNumber) {  super();  this.name = name;  this.passportNumber = passportNumber;  }  **// Getters and Setters omitted**  @Override  public String toString() {  return String.format("Student [id=%s, name=%s, passportNumber=%s]", id, name, passportNumber);  }  } |
| **/src/main/java/com/in28minutes/springboot/soap/web/services/example/student/StudentDetailsEndpoint.java**  package com.in28minutes.springboot.soap.web.services.example.student;  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.in28minutes.students.GetStudentDetailsRequest;  import com.in28minutes.students.GetStudentDetailsResponse;  import com.in28minutes.students.StudentDetails;  @Endpoint  public class StudentDetailsEndpoint {  @PayloadRoot(namespace = "http://in28minutes.com/students", localPart = "GetStudentDetailsRequest") @ResponsePayload  public GetStudentDetailsResponse processCourseDetailsRequest  (@RequestPayload GetStudentDetailsRequest request) {  GetStudentDetailsResponse response = new GetStudentDetailsResponse();  StudentDetails studentDetails = new StudentDetails();  studentDetails.setId(request.getId());  studentDetails.setName("Adam");  studentDetails.setPassportNumber("E1234567");    response.setStudentDetails(studentDetails);  return response;  }  } |

**Few important things to note:**

* @Endpoint - Annotation to indicate that this is a Web Service Endpoint.
* @PayloadRoot(namespace = "http://in28minutes.com/students", localPart = "GetStudentDetailsRequest") - Defines the details of the request that this method would handle. We will handle GetStudentDetailsRequest with the given namespace.
* @ResponsePayload - This method will return a response which would need to be converted to a response xml.
* public GetStudentDetailsResponse processCourseDetailsRequest(@RequestPayload GetStudentDetailsRequest request) - Method would handle the request. @RequestPayload indicates that this is got from the request.

### Configure the Message Dispatcher Servlet to receive the request

|  |
| --- |
| /src/main/java/com/in28minutes/springboot/soap/web/services/example/WebServiceConfig.java  @EnableWs  @Configuration  public class WebServiceConfig {  @Bean  public ServletRegistrationBean messageDispatcherServlet(ApplicationContext context) {  MessageDispatcherServlet messageDispatcherServlet = new MessageDispatcherServlet();  messageDispatcherServlet.setApplicationContext(context);  messageDispatcherServlet.setTransformWsdlLocations(true);  return new ServletRegistrationBean(messageDispatcherServlet, "/ws/\*");  }  } |

**Notes**

* @EnableWs - Enable SOAP Web Service features in this Spring Boot application.
* @Configuration - This class contains A spring configuration.
* @Bean public ServletRegistrationBean messageDispatcherServlet(ApplicationContext context) - We would want to create message dispatcher servlet to act as a front controller. return new ServletRegistrationBean(messageDispatcherServlet, "/ws/\*") - Configure the URL to the web services.

### Spring Web Services Configuration to Generate WSDL

Lets add the wsdl4j dependency to our pom.xml.

/pom.xml

<dependency>

<groupId>wsdl4j</groupId>

<artifactId>wsdl4j</artifactId>

</dependency>

**Let’s enhance the WebServiceConfig to expose the WSDL.**

|  |
| --- |
| /src/main/java/com/in28minutes/springboot/soap/web/services/example/WebServiceConfig.java  @Bean(name = "students")  public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema studentsSchema) {  DefaultWsdl11Definition definition = new DefaultWsdl11Definition();  definition.setPortTypeName("StudentPort");  definition.setTargetNamespace("http://in28minutes.com/students");  definition.setLocationUri("/ws");  definition.setSchema(studentsSchema);  return definition;  }  @Bean  public XsdSchema studentsSchema() {  return new SimpleXsdSchema(new ClassPathResource("student-details.xsd"));  } |

**Notes**

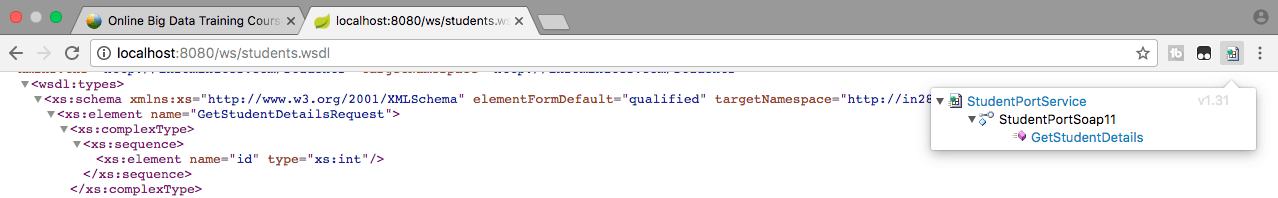
* @Bean(name = "students") - A spring bean. The name of the bean is the name of the wsdl in the URL.
* DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema studentsSchema)
* definition.setTargetNamespace("http://in28minutes.com/students") - Default name space
* definition.setLocationUri("/ws") - The url where we want to expose the wsdl at.
* definition.setSchema(studentsSchema) - We would create WSDL based on the xsd defined here - new SimpleXsdSchema(new ClassPathResource("student-details.xsd"))

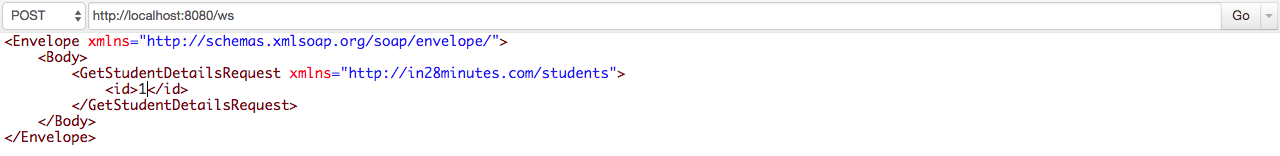
**URL of the WSDL -** [**http://localhost:8080/ws/students.wsdl**](http://localhost:8080/ws/students.wsdl)

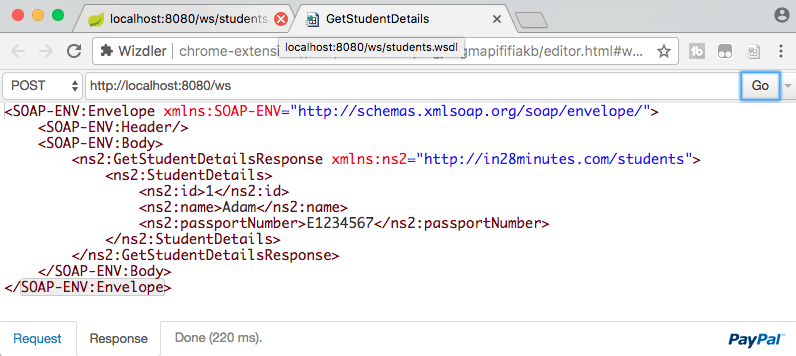
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="no"?><wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="http://in28minutes.com/students" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://in28minutes.com/students" targetNamespace="http://in28minutes.com/students">  <wsdl:types>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://in28minutes.com/students">    <xs:element name="GetStudentDetailsRequest">  <xs:complexType>  <xs:sequence>  <xs:element name="id" type="xs:int"/>  </xs:sequence>  </xs:complexType>  </xs:element>    <xs:element name="GetStudentDetailsResponse">  <xs:complexType>  <xs:sequence>  <xs:element name="StudentDetails" type="tns:StudentDetails"/>  </xs:sequence>  </xs:complexType>  </xs:element>    <xs:complexType name="StudentDetails">  <xs:sequence>  <xs:element name="id" type="xs:int"/>  <xs:element name="name" type="xs:string"/>  <xs:element name="passportNumber" type="xs:string"/>  </xs:sequence>  </xs:complexType>    </xs:schema>  </wsdl:types>  <wsdl:message name="GetStudentDetailsResponse">  <wsdl:part element="tns:GetStudentDetailsResponse" name="GetStudentDetailsResponse">  </wsdl:part>  </wsdl:message>  <wsdl:message name="GetStudentDetailsRequest">  <wsdl:part element="tns:GetStudentDetailsRequest" name="GetStudentDetailsRequest">  </wsdl:part>  </wsdl:message>  <wsdl:portType name="StudentPort">  <wsdl:operation name="GetStudentDetails">  <wsdl:input message="tns:GetStudentDetailsRequest" name="GetStudentDetailsRequest">  </wsdl:input>  <wsdl:output message="tns:GetStudentDetailsResponse"  name="GetStudentDetailsResponse">  </wsdl:output>  </wsdl:operation>  </wsdl:portType>  <wsdl:binding name="StudentPortSoap11" type="tns:StudentPort">  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>  <wsdl:operation name="GetStudentDetails">  <soap:operation soapAction=""/>  <wsdl:input name="GetStudentDetailsRequest">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="GetStudentDetailsResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  </wsdl:binding>  <wsdl:service name="StudentPortService">  <wsdl:port binding="tns:StudentPortSoap11" name="StudentPortSoap11">  <soap:address location="http://localhost:8080/ws"/>  </wsdl:port>  </wsdl:service>  </wsdl:definitions> |

### Executing Request using Wizdler

Install the chrome plugin Wizdler.

Once you install wizdler and launch the wsdl url http://localhost:8080/ws/students.wsdl, you would see a small icon at the corner of the chrome browser, which you can click to see the services that are part of the wsdl. Go ahead and click the Wizdler icon and click the service GetStudentDetails 

This would launch a window to execute the request. Change the id to 1. Click Go button at the top right corner of the screen. 

You should see the response as shown below. 

|  |
| --- |
| **Request**  <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">  <Body>  <GetStudentDetailsRequest xmlns="http://in28minutes.com/students">  <id>1</id>  </GetStudentDetailsRequest>  </Body>  </Envelope> |
| **Response**  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  <SOAP-ENV:Header/>  <SOAP-ENV:Body>  <ns2:GetStudentDetailsResponse xmlns:ns2="http://in28minutes.com/students">  <ns2:StudentDetails>  <ns2:id>1</ns2:id>  <ns2:name>Adam</ns2:name>  <ns2:passportNumber>E1234567</ns2:passportNumber>  </ns2:StudentDetails>  </ns2:GetStudentDetailsResponse>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |

## More SOAP Methods

You can enhance the endpoint to expose more operations. The steps would be

* Define the structure for Request and Response in XSD
* Enhance the Endpoint to process the Request
* Go ahead and test it.

Other thing you can work on is to remove the hardcoding and add business logic and persistence stuff using JPA.

Good Luck!

## Complete Code Example : /pom.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 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.in28minutes.springboot.soap.web.services.example</groupId>  <artifactId>spring-boot-tutorial-soap-web-services</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>spring-boot-tutorial-soap-web-services</name>  <description>**SOAP Web Services with Spring Boot**</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>**spring-boot-starter-parent**</artifactId>  <version>2.3.1.RELEASE</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  **<java.version>1.8</java.version>**  <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>**spring-boot-starter-web-services**</artifactId>  </dependency>  <dependency>  <groupId>wsdl4j</groupId>  **<artifactId>wsdl4j</artifactId>**  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>**spring-boot-devtools**</artifactId>  <scope>runtime</scope>  </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>1.6</version>  <executions>  <execution>  <id>xjc</id>  <goals>  <goal>xjc</goal>  </goals>  </execution>  </executions>  <configuration>  **<schemaDirectory>${project.basedir}/src/main/resources</schemaDirectory>**  **<outputDirectory>${project.basedir}/src/main/java</outputDirectory>**  <clearOutputDir>false</clearOutputDir>  </configuration>  </plugin>  </plugins>  </build>  <repositories>  <repository>  <id>spring-snapshots</id>  <name>Spring Snapshots</name>  <url>https://repo.spring.io/snapshot</url>  <snapshots>  <enabled>true</enabled>  </snapshots>  </repository>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories>  <pluginRepositories>  <pluginRepository>  <id>spring-snapshots</id>  <name>Spring Snapshots</name>  <url>https://repo.spring.io/snapshot</url>  <snapshots>  <enabled>true</enabled>  </snapshots>  </pluginRepository>  <pluginRepository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </pluginRepository>  </pluginRepositories>  </project> |

<!-- http://localhost:8080/ws/students.wsdl -->

**/src/main/java/com/in28minutes/springboot/soap/web/services/example/SpringBootTutorialSoapWebServicesApplication.java**

package com.in28minutes.springboot.soap.web.services.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

**@SpringBootApplication**

public class SpringBootTutorialSoapWebServicesApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootTutorialSoapWebServicesApplication.class, args);

}

}

### /src/main/java/com/in28minutes/springboot/soap/web/services/example/student/Student.java

package com.in28minutes.springboot.soap.web.services.example.student;

public class Student {

private Long id;

private String name;

private String passportNumber;

public Student() {

super();

}

public Student(Long id, String name, String passportNumber) {

super();

this.id = id;

this.name = name;

this.passportNumber = passportNumber;

}

public Student(String name, String passportNumber) {

super();

this.name = name;

this.passportNumber = passportNumber;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPassportNumber() {

return passportNumber;

}

public void setPassportNumber(String passportNumber) {

this.passportNumber = passportNumber;

}

@Override

public String toString() {

return String.format("Student [id=%s, name=%s, passportNumber=%s]", id, name, passportNumber);

}

}

### /src/main/java/com/in28minutes/springboot/soap/web/services/example/student/StudentDetailsEndpoint.java

package com.in28minutes.springboot.soap.web.services.example.student;

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.in28minutes.students.GetStudentDetailsRequest;

import com.in28minutes.students.GetStudentDetailsResponse;

import com.in28minutes.students.StudentDetails;

**@Endpoint**

public class StudentDetailsEndpoint {

@**PayloadRoot**(namespace = "http://in28minutes.com/students", localPart = "GetStudentDetailsRequest")

@**ResponsePayload**

public **GetStudentDetailsResponse** processCourseDetailsRequest(@**RequestPayload** **GetStudentDetailsRequest** request) {

GetStudentDetailsResponse response = new GetStudentDetailsResponse();

StudentDetails studentDetails = new StudentDetails();

studentDetails.setId(request.getId());

studentDetails.setName("Adam");

studentDetails.setPassportNumber("E1234567");

response.setStudentDetails(studentDetails);

return response;

}

}

### /src/main/java/com/in28minutes/springboot/soap/web/services/example/WebServiceConfig.java

package com.in28minutes.springboot.soap.web.services.example;

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.transport.http.MessageDispatcherServlet;

import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;

import org.springframework.xml.xsd.SimpleXsdSchema;

import org.springframework.xml.xsd.XsdSchema;

@EnableWs

@Configuration

public class WebServiceConfig {

@Bean

public ServletRegistrationBean messageDispatcherServlet(ApplicationContext context) {

MessageDispatcherServlet messageDispatcherServlet = new MessageDispatcherServlet();

messageDispatcherServlet.setApplicationContext(context);

messageDispatcherServlet.setTransformWsdlLocations(true);

return new ServletRegistrationBean(messageDispatcherServlet, "/ws/\*");

}

@Bean(name = "students")

public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema studentsSchema) {

DefaultWsdl11Definition definition = new DefaultWsdl11Definition();

definition.setPortTypeName("StudentPort");

definition.setTargetNamespace("http://in28minutes.com/students");

**definition.setLocationUri("/ws");**

definition.setSchema(studentsSchema);

return definition;

}

@Bean

public **XsdSchema** studentsSchema() {

return new SimpleXsdSchema(**new ClassPathResource("student-details.xsd"));**

}

}

### /src/main/java/com/in28minutes/students/GetStudentDetailsRequest.java

//

// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, v2.2.7

// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>

// Any modifications to this file will be lost upon recompilation of the source schema.

// Generated on: 2017.11.28 at 02:57:40 PM IST

//

package com.in28minutes.students;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlType;

/\*\*

\* <p>Java class for anonymous complex type.

\*

\* <p>The following schema fragment specifies the expected content contained within this class.

\*

\* <pre>

\* &lt;complexType>

\* &lt;complexContent>

\* &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">

\* &lt;sequence>

\* &lt;element name="id" type="{http://www.w3.org/2001/XMLSchema}int"/>

\* &lt;/sequence>

\* &lt;/restriction>

\* &lt;/complexContent>

\* &lt;/complexType>

\* </pre>

\*

\*

\*/

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "", propOrder = {

"id"

})

@XmlRootElement(name = "GetStudentDetailsRequest")

public class GetStudentDetailsRequest {

protected int id;

/\*\*

\* Gets the value of the id property.

\*

\*/

public int getId() {

return id;

}

/\*\*

\* Sets the value of the id property.

\*

\*/

public void setId(int value) {

this.id = value;

}

}

### /src/main/java/com/in28minutes/students/GetStudentDetailsResponse.java

//

// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, v2.2.7

// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>

// Any modifications to this file will be lost upon recompilation of the source schema.

// Generated on: 2017.11.28 at 02:57:40 PM IST

//

package com.in28minutes.students;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlType;

/\*\*

\* <p>Java class for anonymous complex type.

\*

\* <p>The following schema fragment specifies the expected content contained within this class.

\*

\* <pre>

\* &lt;complexType>

\* &lt;complexContent>

\* &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">

\* &lt;sequence>

\* &lt;element name="StudentDetails" type="{http://in28minutes.com/students}StudentDetails"/>

\* &lt;/sequence>

\* &lt;/restriction>

\* &lt;/complexContent>

\* &lt;/complexType>

\* </pre>

\*

\*

\*/

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "", propOrder = {

"studentDetails"

})

@XmlRootElement(name = "GetStudentDetailsResponse")

public class GetStudentDetailsResponse {

@XmlElement(name = "StudentDetails", required = true)

protected StudentDetails studentDetails;

/\*\*

\* Gets the value of the studentDetails property.

\*

\* @return

\* possible object is

\* {@link StudentDetails }

\*

\*/

public StudentDetails getStudentDetails() {

return studentDetails;

}

/\*\*

\* Sets the value of the studentDetails property.

\*

\* @param value

\* allowed object is

\* {@link StudentDetails }

\*

\*/

public void setStudentDetails(StudentDetails value) {

this.studentDetails = value;

}

}

### /src/main/java/com/in28minutes/students/ObjectFactory.java

//

// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, v2.2.7

// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>

// Any modifications to this file will be lost upon recompilation of the source schema.

// Generated on: 2017.11.28 at 02:57:40 PM IST

//

package com.in28minutes.students;

import javax.xml.bind.annotation.XmlRegistry;

/\*\*

\* This object contains factory methods for each

\* Java content interface and Java element interface

\* generated in the com.in28minutes.students package.

\* <p>An ObjectFactory allows you to programatically

\* construct new instances of the Java representation

\* for XML content. The Java representation of XML

\* content can consist of schema derived interfaces

\* and classes representing the binding of schema

\* type definitions, element declarations and model

\* groups. Factory methods for each of these are

\* provided in this class.

\*

\*/

@XmlRegistry

public class ObjectFactory {

/\*\*

\* Create a new ObjectFactory that can be used to create new instances of schema derived classes for package: com.in28minutes.students

\*

\*/

public ObjectFactory() {

}

/\*\*

\* Create an instance of {@link GetStudentDetailsResponse }

\*

\*/

public GetStudentDetailsResponse createGetStudentDetailsResponse() {

return new GetStudentDetailsResponse();

}

/\*\*

\* Create an instance of {@link StudentDetails }

\*

\*/

public StudentDetails createStudentDetails() {

return new StudentDetails();

}

/\*\*

\* Create an instance of {@link GetStudentDetailsRequest }

\*

\*/

public GetStudentDetailsRequest createGetStudentDetailsRequest() {

return new GetStudentDetailsRequest();

}

}

### /src/main/java/com/in28minutes/students/package-info.java

//

// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, v2.2.7

// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>

// Any modifications to this file will be lost upon recompilation of the source schema.

// Generated on: 2017.11.28 at 02:57:40 PM IST

//

@javax.xml.bind.annotation.XmlSchema(namespace = "http://in28minutes.com/students", elementFormDefault = javax.xml.bind.annotation.XmlNsForm.QUALIFIED)

package com.in28minutes.students;

### /src/main/java/com/in28minutes/students/StudentDetails.java

//

// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, v2.2.7

// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>

// Any modifications to this file will be lost upon recompilation of the source schema.

// Generated on: 2017.11.28 at 02:57:40 PM IST

//

package com.in28minutes.students;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlType;

/\*\*

\* <p>Java class for StudentDetails complex type.

\*

\* <p>The following schema fragment specifies the expected content contained within this class.

\*

\* <pre>

\* &lt;complexType name="StudentDetails">

\* &lt;complexContent>

\* &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">

\* &lt;sequence>

\* &lt;element name="id" type="{http://www.w3.org/2001/XMLSchema}int"/>

\* &lt;element name="name" type="{http://www.w3.org/2001/XMLSchema}string"/>

\* &lt;element name="passportNumber" type="{http://www.w3.org/2001/XMLSchema}string"/>

\* &lt;/sequence>

\* &lt;/restriction>

\* &lt;/complexContent>

\* &lt;/complexType>

\* </pre>

\*

\*

\*/

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "StudentDetails", propOrder = {

"id",

"name",

"passportNumber"

})

public class StudentDetails {

protected int id;

@XmlElement(required = true)

protected String name;

@XmlElement(required = true)

protected String passportNumber;

/\*\*

\* Gets the value of the id property.

\*

\*/

public int getId() {

return id;

}

/\*\*

\* Sets the value of the id property.

\*

\*/

public void setId(int value) {

this.id = value;

}

/\*\*

\* Gets the value of the name property.

\*

\* @return

\* possible object is

\* {@link String }

\*

\*/

public String getName() {

return name;

}

/\*\*

\* Sets the value of the name property.

\*

\* @param value

\* allowed object is

\* {@link String }

\*

\*/

public void setName(String value) {

this.name = value;

}

/\*\*

\* Gets the value of the passportNumber property.

\*

\* @return

\* possible object is

\* {@link String }

\*

\*/

public String getPassportNumber() {

return passportNumber;

}

/\*\*

\* Sets the value of the passportNumber property.

\*

\* @param value

\* allowed object is

\* {@link String }

\*

\*/

public void setPassportNumber(String value) {

this.passportNumber = value;

}

}

### /src/main/resources/application.properties

### /src/main/resources/student-details.xsd

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://in28minutes.com/students"

xmlns:tns="http://in28minutes.com/students" elementFormDefault="qualified">

<xs:element name="GetStudentDetailsRequest">

<xs:complexType>

<xs:sequence>

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

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="GetStudentDetailsResponse">

<xs:complexType>

<xs:sequence>

<xs:element name= "StudentDetails" type="tns:StudentDetails"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:complexType name="StudentDetails">

<xs:sequence>

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

<xs:element name="name" type="xs:string"/>

<xs:element name="passportNumber" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:schema>

### /src/test/java/com/in28minutes/springboot/soap/web/services/example/SpringBootTutorialSoapWebServicesApplicationTests.java

package com.in28minutes.springboot.soap.web.services.example;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootTutorialSoapWebServicesApplicationTests {

@Test

public void contextLoads() {

}

}

Step1- Create the POM.xml with minimum following dependencies.

Pom.xml: Following are main dependencies

* 1. <artifactId>spring-boot-starter-parent</artifactId> parent
  2. <artifactId>spring-boot-starter-web-services</artifactId> for soap service
  3. <artifactId>wsdl4j</artifactId>To generate WSDL. Generate WSDL
  4. <artifactId>spring-boot-devtools</artifactId> to auto start of tomcat server.
  5. <artifactId>spring-boot-starter-test</artifactId> for unit testing
  6. <artifactId>spring-boot-maven-plugin</artifactId> inside build->plugin tag. The **Spring Boot Maven Plugin** provides **Spring Boot** support in Apache **Maven**. It allows us to package executable jar or war archives, run **Spring Boot** applications, generate build information and start your **Spring Boot** application prior to running integration tests.
  7. <artifactId>jaxb2-maven-plugin</artifactId> inside build->plugin tag. This **JAXB (**Java API for XML Binding ) **plugin** used to generate Java classes from XML Schemas (XSD)(and optionally binding files) and to create XML Schemas from annotated Java classes
  8. <goal>xjc</goal> Compiles an XML schema file into fully annotated Java classes.
  9. <schemaDirectory>${project.basedir}/src/main/resources</schemaDirectory>. Specify the place or path of XSD location in the project
  10. <outputDirectory>${project.basedir}/src/main/java</outputDirectory>. Specify the path of java files generated from XSD

Step2- Now create an XSD in the resource folder of the application

### Step3- Create WebServiceConfig class and

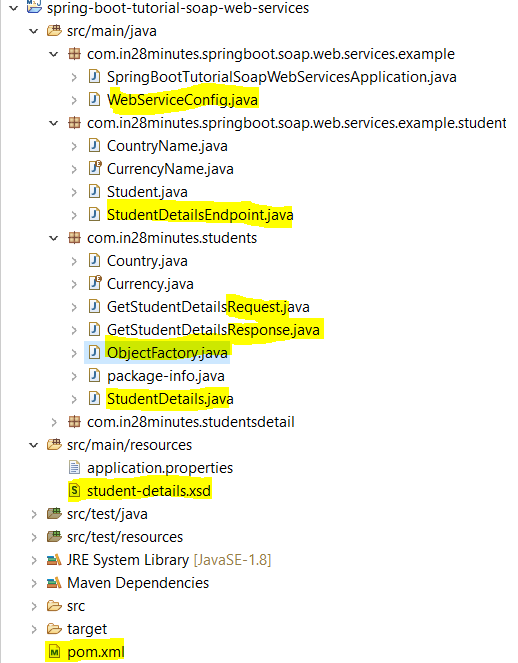
|  |
| --- |
| Configure the Message Dispatcher Servlet to receive the request.   @Bean  **public** ServletRegistrationBean messageDispatcherServlet(ApplicationContext context) {  MessageDispatcherServlet messageDispatcherServlet = **new** MessageDispatcherServlet();  messageDispatcherServlet.setApplicationContext(context);  messageDispatcherServlet.setTransformWsdlLocations(**true**);  **return** **new** ServletRegistrationBean(messageDispatcherServlet, "/**root**/\*");  }   * Servlet for simplified dispatching of Web service messages. * This MessageDispatcherServlet is an alternative of Spring-MVC DispatcherServlet used in SOAP service * This servlet automatically detects EndpointAdapters, EndpointMappings, and Endpo**intExceptionResolvers by type.** * This servlet also automatically detects any WsdlDefinition defined in its application context. |
| 1. **Configure the WSDL name and its context root to be exposed**     @Bean(name = "studentsDetail")  **public** DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema studentsSchema) {  DefaultWsdl11Definition definition = **new** DefaultWsdl11Definition();  definition.setPortTypeName("StudentPort");  definition.setTargetNamespace("http://in28minutes.com/students");  definition.setLocationUri("/**ws**");  definition.setSchema(studentsSchema);  **return** definition;  }  @Bean  **public** XsdSchema studentsSchema() {  **return** **new** SimpleXsdSchema(**new** ClassPathResource("student-details.xsd"));  }   * @Bean(name = "studentsDetail") - A spring bean. The name of the bean is the name of the wsdl in the URL. * DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema studentsSchema) * definition.setTargetNamespace("http://in28minutes.com/students") - Default name space * definition.setLocationUri("/ws") - The url where we want to expose the wsdl at. * definition.setSchema(studentsSchema) - We would create WSDL based on the xsd defined here - new SimpleXsdSchema(new ClassPathResource("student-details.xsd")) |

This WSDL is exposed under the bean name: for example, a WsdlDefinition bean named ‘studentsDetail ‘will be exposed as **studentsDetail.wsdl** in this servlet's context: **http://localhost:8080/ root/ studentsDetail.wsdl**. When the transformWsdlLocations init-param is set to true in this servlet's configuration in web.xml, all location attributes in the WSDL definitions will reflect the URL of the incoming request.

Step4- **Configuring an Endpoint** where we will define the method or operation defined in XSD

Endpoint is the component that receives the request, initiates the processing and sends the response back.

Step5- Create a **SpringBoot launcher class**



|  |
| --- |
| **Step1- Create the POM.xml with minimum following dependencies**  <?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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.in28minutes.springboot.soap.web.services.example</groupId>  <artifactId>spring-boot-tutorial-soap-web-services</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.0.RELEASE</version>  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  <!-- <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version> -->  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web-services</artifactId>  </dependency>  <dependency>  <groupId>wsdl4j</groupId>  <artifactId>wsdl4j</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  </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>1.6</version>  <executions>  <execution>  <id>ID1</id>  <goals>  <goal>xjc</goal>  </goals>  </execution>  </executions>  <configuration>  <schemaDirectory>${project.basedir}/src/main/resources</schemaDirectory>  <outputDirectory>${project.basedir}/src/main/java</outputDirectory>  <clearOutputDir>false</clearOutputDir>  </configuration>  </plugin>  </plugins>  </build>  </project>  <!-- http://localhost:8080/ws/students.wsdl --> |
| **Step2- Now create an XSD in the resource folder of the application**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <xs:schema xmlns:xs=*"http://www.w3.org/2001/XMLSchema"*  targetNamespace=*"http://in28minutes.com/studentsDetail"*  xmlns:tns=*"http://in28minutes.com/studentsDetail"* elementFormDefault=*"qualified"*>    <xs:element name=*"GetStudentDetailsRequest"*>  <xs:complexType>  <xs:sequence>  <xs:element name= *"id"* type=*"xs:int"*/>  </xs:sequence>  </xs:complexType>  </xs:element>    <xs:element name=*"GetStudentDetailsResponse"*>  <xs:complexType>  <xs:sequence>  <xs:element name= *"StudentDetails"* type=*"tns:StudentDetails"*/>  </xs:sequence>  </xs:complexType>  </xs:element>    <xs:complexType name=*"StudentDetails"*>  <xs:sequence>  <xs:element name=*"id"* type=*"xs:int"*/>  <xs:element name=*"name"* type=*"xs:string"*/>  <xs:element name=*"passportNumber"* type=*"xs:string"*/>  <xs:element name=*"age"* type=*"xs:int"*/>  <xs:element name=*"country"* type=*"tns:country"*/>  </xs:sequence>  </xs:complexType>  <xs:complexType name=*"country"*>  <xs:sequence>  <xs:element name=*"name"* type=*"xs:string"*/>  <xs:element name=*"population"* type=*"xs:int"*/>  <xs:element name=*"capital"* type=*"xs:string"*/>  <xs:element name=*"currency"* type=*"tns:currency"*/>  </xs:sequence>  </xs:complexType>  <xs:simpleType name=*"currency"*>  <xs:restriction base=*"xs:string"*>  <xs:enumeration value=*"GBP"*/>  <xs:enumeration value=*"EUR"*/>  <xs:enumeration value=*"PLN"*/>  </xs:restriction>  </xs:simpleType>  </xs:schema> |
| Step3- Create WebServiceConfig class and package com.in28minutes.springboot.soap.web.services.example;  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.transport.http.MessageDispatcherServlet;  import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;  import org.springframework.xml.xsd.SimpleXsdSchema;  import org.springframework.xml.xsd.XsdSchema;  @EnableWs  @Configuration  public class WebServiceConfig {  @Bean  public ServletRegistrationBean messageDispatcherServlet(ApplicationContext context) {  MessageDispatcherServlet messageDispatcherServlet = new MessageDispatcherServlet();  messageDispatcherServlet.setApplicationContext(context);  messageDispatcherServlet.setTransformWsdlLocations(true);  return new ServletRegistrationBean(messageDispatcherServlet, "/root/\*");  }  @Bean(name = "studentsDetail")  public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema studentsSchema) {  DefaultWsdl11Definition definition = new DefaultWsdl11Definition();  definition.setPortTypeName("StudentPort");  definition.setTargetNamespace("http://in28minutes.com/students");  definition.setLocationUri("/ws");  definition.setSchema(studentsSchema);  return definition;  }  @Bean  public XsdSchema studentsSchema() {  return new SimpleXsdSchema(new ClassPathResource("student-details.xsd"));  }  } |
| **Step5- Create a SpringBoot launcher class**  package com.in28minutes.springboot.soap.web.services.example;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SpringBootTutorialSoapWebServicesApplication {  public static void main(String[] args) {  SpringApplication.run(SpringBootTutorialSoapWebServicesApplication.class, args);  }  } |
| **Step4- Configuring an Endpoint where we will define the method or operation defined in XSD**  package com.in28minutes.springboot.soap.web.services.example.student;  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.in28minutes.students.Country;  import com.in28minutes.students.Currency;  import com.in28minutes.students.GetStudentDetailsRequest;  import com.in28minutes.students.GetStudentDetailsResponse;  import com.in28minutes.students.StudentDetails;  @Endpoint  public class StudentDetailsEndpoint {  @PayloadRoot(namespace = "http://in28minutes.com/students", localPart = "GetStudentDetailsRequest")  @ResponsePayload  public GetStudentDetailsResponse processCourseDetailsRequest(@RequestPayload GetStudentDetailsRequest request) {  GetStudentDetailsResponse response = new GetStudentDetailsResponse();  System.out.println("Insed endppoint class");  StudentDetails studentDetails = new StudentDetails();  try {  studentDetails.setId(request.getId());  studentDetails.setName("Adam");  studentDetails.setPassportNumber("E1234567");  studentDetails.setAge(30);  Country countryObj= new Country();  countryObj.setCapital("Delhi");  countryObj.setName("India");  countryObj.setPopulation(450000);  studentDetails.setCountry(countryObj);  studentDetails.getCountry().setCurrency(Currency.GBP);  } catch (Exception e) {  e.getStackTrace();  }  response.setStudentDetails(studentDetails);    return response;  }  } |