

HowToDoInJava

[Java 8](#)[Regex](#)[Concurrency](#)[Best Practices](#)[Spring Boot](#)[JUnit5](#)[Interview Questions](#)

Spring Boot 2 Tutorial

[Spring Boot – Introduction](#)[Spring Boot – Starter parent](#)[Spring Boot – Starter templates](#)[Spring Boot – Multi-module project](#)[Spring Boot – Annotations](#)[Spring Boot – Auto configuration](#)[Spring Boot – Logging](#)[Spring Boot – DevTools](#)[Spring Boot – WAR Packaging](#)[Spring Boot – Hibernate](#)[Spring Boot – REST API](#)[Spring Boot – ResponseBodyEmitter](#)[Spring Boot – SseEmitter](#)[Spring Boot – POST with Headers](#)[Spring Boot – Error Handling](#)[Spring Boot – HATEOAS](#)[Spring Boot – @Async](#)[Spring Boot – Async Controller](#)[Spring Boot – Caching](#)[Spring Boot – Retry](#)[Spring Boot – JUnit](#)[Spring Boot – BasicAuth](#)[Spring Boot – MockMvc](#)[Spring Boot – Mockito](#)[Spring Boot – H2 Database](#)[Spring Boot – Ehcache 3.x](#)[Spring Boot – Gson](#)

Spring Boot SOAP Webservice Example

By Sajal Chakraborty | Filed Under: [Spring Boot](#)

Learn to leverage Spring boot's simplicity to **create SOAP webservice** quickly. [REST](#) and [microservices](#) are gaining popularity everyday but still [SOAP](#) has its own place in some situations. In this **spring boot soap tutorial**, we will focus only in the Spring boot related configurations to see how easily we can create our **contract first SOAP webservice**.

We will build a simple contract first SOAP web service where we will implement *Student search functionality* with hard coded backend for demo purpose.

Table of Contents

- [1. Technology Stack](#)
- [2. Project Structure](#)
- [3. Create Spring Boot Project](#)
- [4. Create SOAP Domain and Generate Java Code](#)
- [5. Create SOAP WS Endpoint](#)
- [6. Add Configuration Beans](#)
- [7. Demo](#)
- [8. Summary](#)

1. Technology Stack

- JDK 1.8, Eclipse, Maven – Development environment
- Spring-boot – Underlying application framework
- [wsdl4j](#) – for publishing WSDL for our Service
- [SOAP-UI](#) – for testing our service
- [JAXB maven plugin](#) – for code generation

Search Tutorials



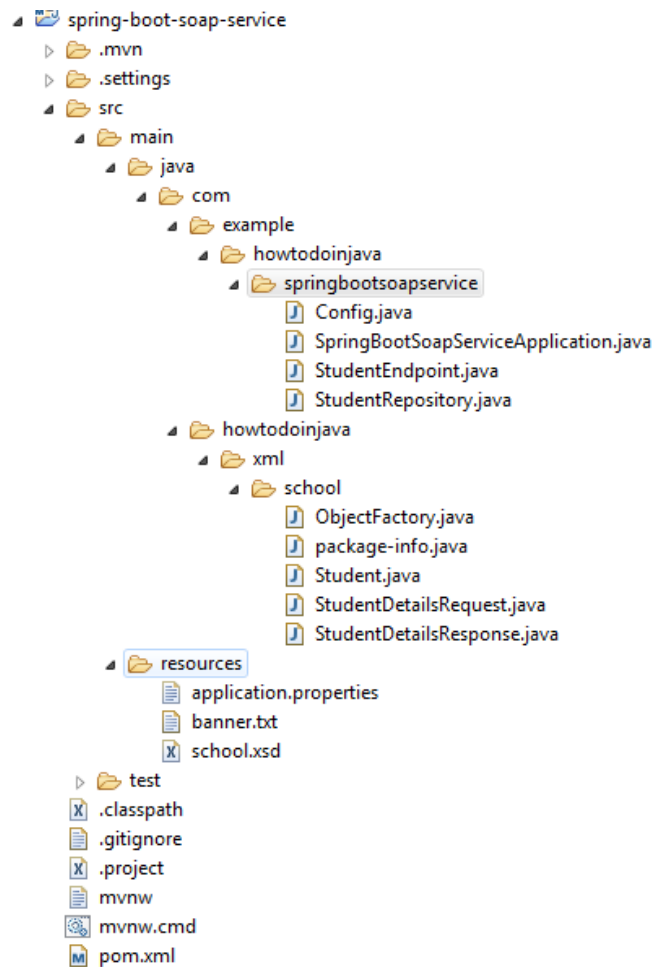
Spring Boot –
Performance Logs
Spring Boot – Test
Async Controller
Spring Boot –
TestRestTemplate
Spring Boot –
RestTemplate GET
Spring Boot –
RestTemplate POST
Spring – RestTemplate
timeout
Spring – RestTemplate
BasicAuth

Spring Boot Tutorial

Spring Boot –
CommandLineRunner
Spring Boot –
Configure Jetty Server
Spring Boot – Tomcat
Default Port
Spring Boot – Change
Application Root
Spring Boot – SSL
[https]
Spring Boot – Get all
loaded beans
Spring Boot – Custom
PropertyEditor
Spring Boot –
@EnableScheduling
Spring Boot – Jersey
**Spring Boot – SOAP
Webservice**
Spring Boot – SOAP
Client
Spring Boot –
JMSTemplate
Spring Boot – REST
APIs
Spring Boot – JSP View
Spring Boot – Logging
yaml Config
Spring Boot – Logging
property Config
Spring Boot – Actuator
endpoints
Spring Boot – Role
Based Security
Spring Boot – RSS /
ATOM Feed

2. Project Structure

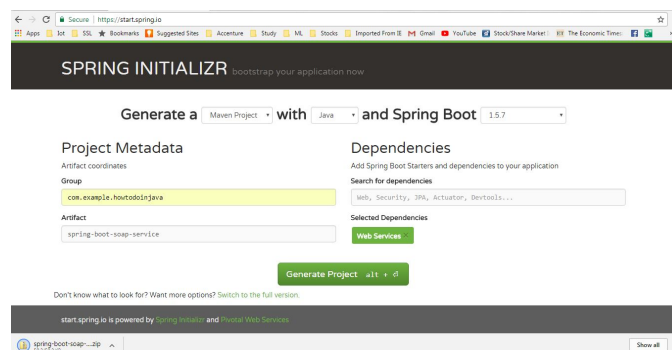
The classes and files created for this demo are shown below.



Spring Boot SOAP WS Project Structure

3. Create Spring Boot Project

Create one spring boot project from [SPRING INITIALIZR](https://start.spring.io) site with **Web Services** dependency only. After selecting the dependency and giving the proper maven GAV coordinates, download project in zipped format. Unzip and then import project in eclipse as maven project.



Generate Spring boot project



Spring Boot – Ehcache
2.x

Popular Tutorials

Java 8 Tutorial
Core Java Tutorial
Collections in Java
Java Concurrency
Spring Boot Tutorial
Spring AOP Tutorial
Spring MVC Tutorial
Spring Security Tutorial
Hibernate Tutorial
Jersey Tutorial
Maven Tutorial
Log4j Tutorial
Regex Tutorial

Add Wsd14j Dependency

Edit `pom.xml` and add this dependency to your project.

```
<dependency>
  <groupId>wsdl4j</groupId>
  <artifactId>wsdl4j</artifactId>
</dependency>
```

4. Create SOAP Domain model and Generate Java Code

As we are following the contract first approach to develop the service, we need to first create the domain (methods and parameters) for our service. For simplicity, we have kept both request and response in same [XSD](#), but in actual enterprise use case, we will have multiple XSDs importing each other to form the final definition.

`student.xsd`

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

  <xs:element name="StudentDetailsRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="StudentDetailsResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Student"
          type="tns:Student"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Student">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="standard" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

```

        <xs:element name="address" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

</xs:schema>

```

Place above file in **resources** folder of the project.

Add JAXB maven plugin for XSD to Java object generation

We will use **jaxb2-maven-plugin** to generate the domain classes efficiently. We need to now add the below maven plug in to the plugin section of project's **pom.xml** file.

```

<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>

```

The plugin uses **XJC** tool as code generation engine. XJC compiles an XML schema file into fully annotated Java classes.

Now execute above maven plugin to *generate java code from XSD*.

5. Create SOAP Webservice Endpoint

StudentEndpoint class will handle all the incoming requests for the service and will delegate the call to the finder method of the data repository.

```

package com.example.howtodoinjava.springbootsoapservice;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ws.server.endpoint.annotation.Endpoint;
;
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
import org.springframework.ws.server.endpoint.annotation.RequestP

```

```

ayload;
import
org.springframework.ws.server.endpoint.annotation.Response
Payload;
import com.howtodoinjava.xml.school.StudentDetailsRequest;
import
com.howtodoinjava.xml.school.StudentDetailsResponse;

@Endpoint
public class StudentEndpoint
{
    private static final String NAMESPACE_URI =
"https://www.howtodoinjava.com/xml/school";

    private StudentRepository studentRepository;

    @Autowired
    public StudentEndpoint(StudentRepository
StudentRepository) {
        this.studentRepository = studentRepository;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart =
"StudentDetailsRequest")
    @ResponsePayload
    public StudentDetailsResponse
getStudent(@RequestPayload StudentDetailsRequest request)
{
        StudentDetailsResponse response = new
StudentDetailsResponse();
        response.setStudent(studentRepository.findStudent(
request.getName()));

        return response;
    }
}

```

Here few details about the annotations –

1. `@Endpoint` registers the class with Spring WS as a potential candidate for processing incoming SOAP messages.
2. `@PayloadRoot` is then used by Spring WS to pick the handler method based on the message's namespace and localPart. Please note the Namespace URL and Request Payload root request mentioned in this annotation.
3. `@RequestPayload` indicates that the incoming message will be mapped to the method's request parameter.
4. The `@ResponsePayload` annotation makes Spring WS map the returned value to the response payload.

Create Data Repository

As mentioned, we will use the hardcoded data as backend for this demo, let's add one class called `StudentRepository.java` with Spring `@Repository` annotation. It will simply hold data in `HashMap` and will also give one finder method called `findStudent()`.

Read More – [@Repository Annotations](#)

```

package com.example.howtodoinjava.springbootsoapservice;

import java.util.HashMap;
import java.util.Map;
import javax.annotation.PostConstruct;
import org.springframework.stereotype.Component;
import org.springframework.util.Assert;
import com.howtodoinjava.xml.school.Student;

@Component
public class StudentRepository {
    private static final Map<String, Student> students =
new HashMap<>();

    @PostConstruct
    public void initData() {

        Student student = new Student();
        student.setName("Sajal");
        student.setStandard(5);
        student.setAddress("Pune");
        students.put(student.getName(), student);

        student = new Student();
        student.setName("Kajal");
        student.setStandard(5);
        student.setAddress("Chicago");
        students.put(student.getName(), student);

        student = new Student();
        student.setName("Lokesh");
        student.setStandard(6);
        student.setAddress("Delhi");
        students.put(student.getName(), student);

        student = new Student();
        student.setName("Sukesh");
        student.setStandard(7);
        student.setAddress("Noida");
        students.put(student.getName(), student);
    }

    public Student findStudent(String name) {
        Assert.notNull(name, "The Student's name must not
be null");
        return students.get(name);
    }
}

```

6. Add SOAP Webservice Configuration Beans

Create a class with `@Configuration` annotation to hold bean definitions.

```

package com.example.howtodoinjava.springbootsoapservice;

import
org.springframework.boot.web.servlet.ServletRegistrationBe
an;
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;

@EnableWs
@Configuration
public class Config extends WsConfigurerAdapter
{
    @Bean
    public ServletRegistrationBean
messageDispatcherServlet(ApplicationContext
applicationContext)
    {
        MessageDispatcherServlet servlet = new
MessageDispatcherServlet();
        servlet.setApplicationContext(applicationContext);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean(servlet,
"/service/*");
    }

    @Bean(name = "studentDetailsWsdl")
    public DefaultWsdl11Definition
defaultWsdl11Definition(XsdSchema countriesSchema)
    {
        DefaultWsdl11Definition wsdl11Definition = new
DefaultWsdl11Definition();
        wsdl11Definition.setPortTypeName("StudentDetailsPo
rt");
        wsdl11Definition.setLocationUri("/service/student-
details");
        wsdl11Definition.setTargetNamespace("https://www.h
owtodoinjava.com/xml/school");
        wsdl11Definition.setSchema(countriesSchema);
        return wsdl11Definition;
    }

    @Bean
    public XsdSchema countriesSchema()
    {
        return new SimpleXsdSchema(new
ClassPathResource("school.xsd"));
    }
}

```

- **Config** class extends **WsConfigurerAdapter** which configures annotation driven Spring-WS programming model.
- **MessageDispatcherServlet** – Spring-WS uses it for handling SOAP requests. We need to inject **ApplicationContext** to this servlet so that Spring-WS find other beans. It also declares the URL mapping for the requests.
- **DefaultWsdl11Definition** exposes a standard WSDL 1.1 using **XsdSchema**. The bean name **studentDetailsWsdl** will

be the **wsdl name** that will be exposed. It will be available under

<http://localhost:8080/service/studentDetailsWsd1.wsdl>.

This is the simplest approach to expose the contract first wsdl in spring.

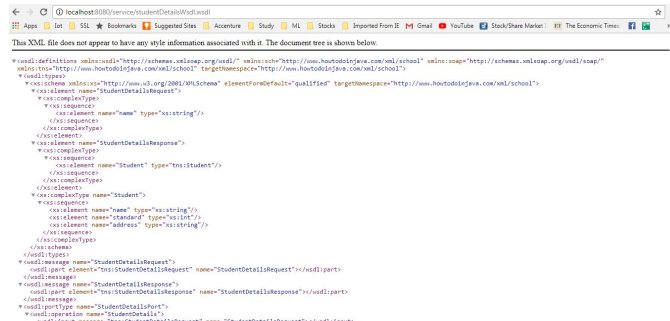
This configuration also uses the WSDL location servlet transformation `servlet.setTransformWsd1Locations(true)` internally. If we see the exported WSDL, the **soap:address** will have the **localhost** address. Similarly, if we instead visit the WSDL from the public facing IP address assigned to the deployed machine, we will see that address instead of **localhost**. So the endpoint URL is dynamic based on the deployment environment.

7. Spring boot SOAP webservice demo

Do maven build using `mvn clean install` and start the application using `java -jar target\spring-boot-soap-service-0.0.1-SNAPSHOT.jar` command. This will bring up one tomcat server in default port **8080** and application will be deployed in it.

1) Now go to

<http://localhost:8080/service/studentDetailsWsd1.wsdl> to see if the WSDL is coming properly.



WSDL generated

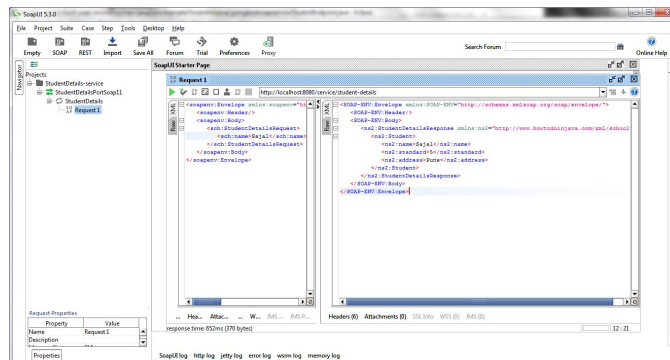
2) Once we have the successful WSDL generated, we can use that WSDL to create a project in SOAP ui and test the application. Sample Request and response is given below.

Request:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sch="https://www.howtodoinjava.com/xml/school">
  <soapenv:Header/>
  <soapenv:Body>
    <sch:StudentDetailsRequest>
      <sch:name>Sajal</sch:name>
    </sch:StudentDetailsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```


Response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:StudentDetailsResponse
      xmlns:ns2="https://www.howtodoinjava.com/xml/school">
      <ns2:Student>
        <ns2:name>Sajal</ns2:name>
        <ns2:standard>5</ns2:standard>
        <ns2:address>Pune</ns2:address>
      </ns2:Student>
    </ns2:StudentDetailsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**SOAP UI Example**

8. Summary

In above example, we learned to **create SOAP webservice using Spring Boot**. We also learned to **generate java code from WSDL**. We learned about beans which are needed to process the SOAP requests.

Feel free to drop a comment if you face any difficulty in running above project.

[Source code for this Article](#)

Happy Learning !!

Read More:

[Spring boot soap web service client example](#)



Aceite cartão com a

Ad Chegou a Total: a má
sem aluguel e com 3x m

SumUp

Confira Agora

**Top 20 Java
Exception
Handling Best
Practices**

**Microservices -
Definition,
Principles and
Benefits**

**Multithreading -
Difference
between lock and
monitor in java**

**Microservices
Tutorials**

**Fork/Join
Framework
Tutorial:
ForkJoinPool...**

**Object level lock
vs Class level lock
in Java**

**Spring WebFlux
Tutorial - Spring
Boot WebFlux
Example**

**RxJa
Wha**

Feedback, Discussion and Comments

[chhabra84](#)

June 7, 2019

Getting Error. Please help

```
<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">unexpected element
(uri:"https://www.howtodoinjava.com/xml/school",
local:"StudentDetailsRequest"). Expected elements are &:
{https://www.howtodoinjava.com/xml/school}StudentDetail:
</faultstring>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

[Reply](#)

Rajendra

May 28, 2019

Is there any way to get header values in the Endpoint? We are working on one project where we have done above part, but one issue is that we need to read header information passing from soap UI. Could anyone help us how to read header values from soap UI to Endpoint @RequestPayload method?

[Reply](#)

vivek kumar

March 21, 2019

Hi Sajal,

I tried your above soap example and it was really helpful for me but now i want to fetch list of data from database then what will be process for it. I means we want to connect with database and communicate with database.

Thanks

vivek kumar

[Reply](#)

vivek kumar

March 20, 2019

hii all

When we generate WSDL file from jar. Now what will be URL to execute the given request in SOAP interface and how to run this project

[Reply](#)

vivek

March 19, 2019

how to connect with database of soup web service

[Reply](#)

Nay Myo Kyaw

March 11, 2019

Hi Sajal,

I have tried your example and it is really helpful. Next thing I want to achieve is to send an object of Student data to the java app via SOAP message. So the app can save Student data in the database. The other way around of your example. Can you help us to provide an example of that? Thanks in advance.

[Reply](#)

Saurabh Singh

February 13, 2019

Hi,

I am following this tutorial from beginning.

can anyone explain what is the need of this SOAP Web service feature. In Spring boot using RestController we are exposing everything as resource which is the ultimate need. So why do we need development using JAX-RS or JAX-WS approach. I am very confused.

Thanks.

[Reply](#)

satra

September 21, 2018

I think here we are generating wsdl from java ..not java objects from wsdl rather java objects are generated from XSD. But in summary its mentioned like "We also learned to generate java code from WSDL." which seems to be

confusing. Let me know if i miss anything to make above comments.

[Reply](#)

Meenal

July 30, 2018

Getting below error in @Bean(name = "studentDetailsWsd"): BeanException: Error creating bean with name 'studentDetailsWsd' defined in class path resource [com/example/howtodoinjava/springbootsoapervice/Config.class]: Invocation of init method failed; nested exception is WSDLException: faultCode=CONFIGURATION_ERROR: No Java extensionType found to represent a '{http://www.w3.org/2001/XMLSchema}schema' element in the context of a 'javax.wsdl.Types':

Please guide!!

[Reply](#)

Meenal

July 30, 2018

I am getting error regarding WsConfigurerAdapter while doing maven install. Please help me!!

[Reply](#)

Meenal

July 29, 2018

I am getting error:- java.lang.IllegalStateException: Failed to load ApplicationContext
Caused by:
org.springframework.beans.factory.BeanDefinitionStoreException: Failed to parse configuration class [com.example.howtodoinjava.springbootsoapervice.SpringBootSoapServiceApplication]; nested exception is java.io.FileNotFoundException: class path resource [org/springframework/ws/config/annotation/WsConfigurerAdapter.class] cannot be opened because it does not exist
Caused by: java.io.FileNotFoundException: class path resource

[org/springframework/ws/config/annotation/WsConfigurerAdapter.class]
cannot be opened because it does not exist

Initially I was getting error for EnableWs imports so I have added few External jars which started supporting EnableWs and WsConfigurerAdapter. Please guide me as I am new to this.

[Reply](#)

Wasim

May 23, 2018

Hi Lokesh,

I am getting this below error while building the project :

Caused by: java.lang.IllegalArgumentException: xsd 'class path resource [school.xsd]' does not exist

Let me know if i am missing anything here.

Thanks

[Reply](#)

dev5264

May 31, 2018

rename it to student.xsd in countriesSchema method of Config class.

[Reply](#)

chen

May 15, 2018

how to change Response content "****" to ***

[Reply](#)

kahn

April 12, 2018

Hi,

The class StudentEndpoint's source code

```
private static final String NAMESPACE_URI =  
    "https://www.howtodoinjava.com/xml/school";
```

should be

```
private static final String NAMESPACE_URI =  
    "https://www.howtodoinjava.com/xml/school";
```

in my opinion.

The difference between them is [https -> http]. Because, the school.xsd file's targetNamespace is [https://www.howtodoinjava.com/xml/school].

[Reply](#)

Witold Kaczurba

April 4, 2018

The attached zip uses .M4 snapshot of spring-boot-starter-parent. It may not be available from the mavencentral, but simply changing it to .RELEASE should solve the issue.

```
org.springframework.boot  
spring-boot-starter-parent  
2.0.0.RELEASE
```

[Reply](#)

javageek

March 9, 2018

One small correction :

```
return new SimpleXsdSchema(new  
    ClassPathResource("school.xsd"));
```

Please change school.xsd to student.xsd

[Reply](#)

disappointed

March 6, 2018

is it me or it doesn't compile?

[Reply](#)

[Lokesh Gupta](#)

March 6, 2018

What compilation error you are getting?

[Reply](#)

[Kishore](#)

December 15, 2017

Thanks for the tutorial , very easy to follow and get the SOAP service up and running in spring boot. Appreciate your time.

[Reply](#)

Ask Questions & Share Feedback

Please do not submit a comment only to say "Thank you".

Comment

*Want to Post Code Snippets or XML content? Please use [java] ... [/java] tags otherwise code may not appear partially or even fully. e.g.

```
[java]
public static void main (String[] args) {
...
}
[/java]
```

Name *

Email *

Website

POST COMMENT

Meta Links

- Advertise
- Contact Us
- Privacy policy
- About Me

Recommended Reading

- 10 Life Lessons
- Secure Hash Algorithms
- How Web Servers work?
- How Java I/O Works Internally?
- Best Way to Learn Java
- Java Best Practices Guide
- Microservices Tutorial
- REST API Tutorial
- How to Start New Blog