**Lab 7 SOAP web service ... in Spring Boot**

This lab will show you how you can create a SOAP web service using Spring Boot. We will create a SOAP web service for retrieving drone parts information. Where the ‘default’ with Spring Boot is SOAP version 1.1, this lab will implement a SOAP 1.2 version.

Starting point is ….

Furthermore, you have to have the provided VirtualBox machine up-and-running. You are logged in under user/password developer/welcome1.

# Overview

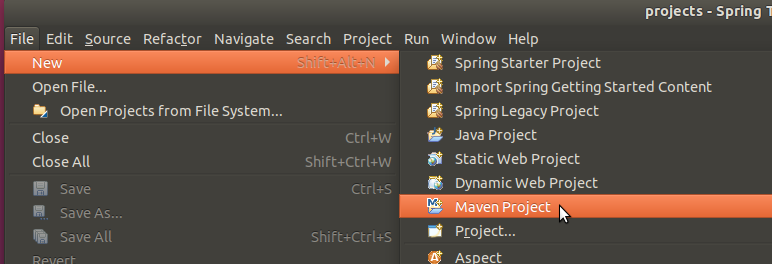
In this lab. We will start with a blacn maven project and add everyting that is needed for the SOAP web service. The following steps will be done in this lab:

* Section 2: Project and the pom.xml
* Section 3: XML schema for messages
* Section 4: JAXB: Java classes for parts.xsd
* Section 5: Parts repository
* Section 6: Web service endpoint
* Section 7: Web service configuration
* Section 8: Runnable application
* Section 9: Run and Test the Soap web service

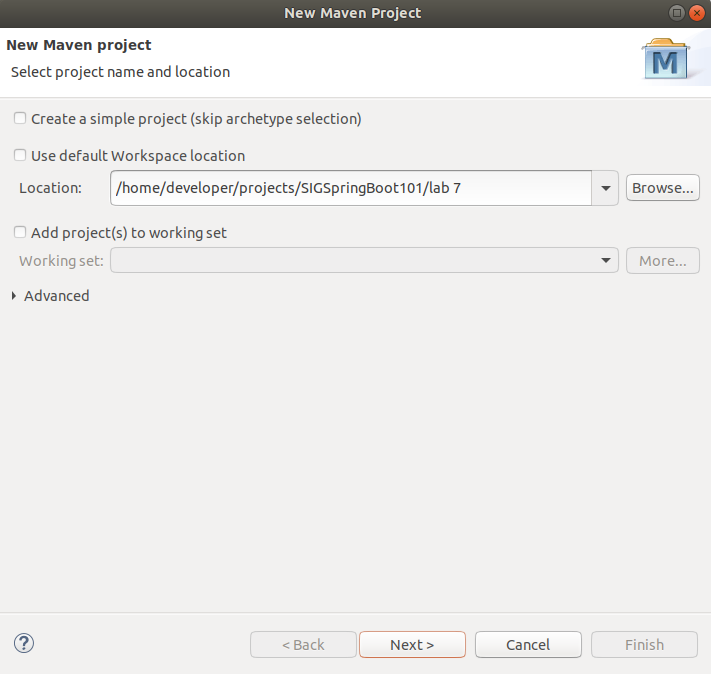
# Project and the pom.xml

First, we are going to create an Eclipse project and update the maven pom file.

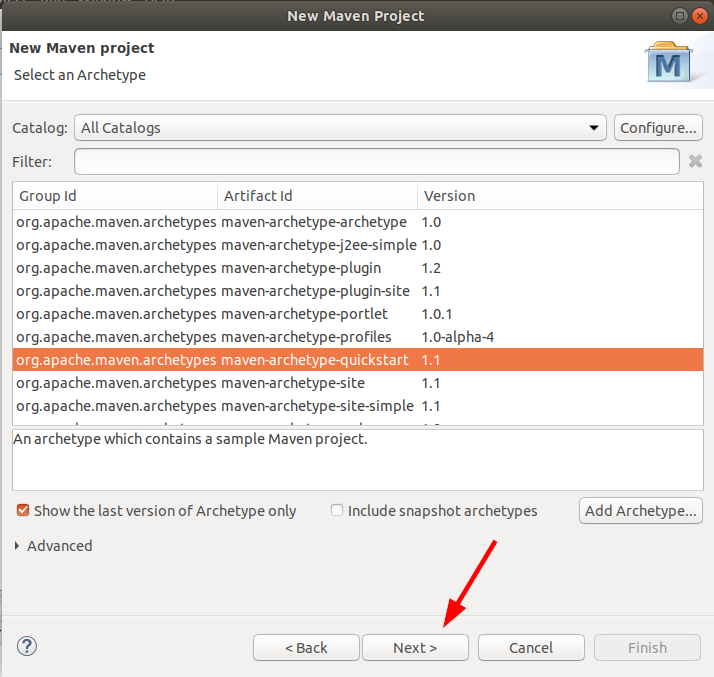
Start Eclipse STS and then create a maven project:



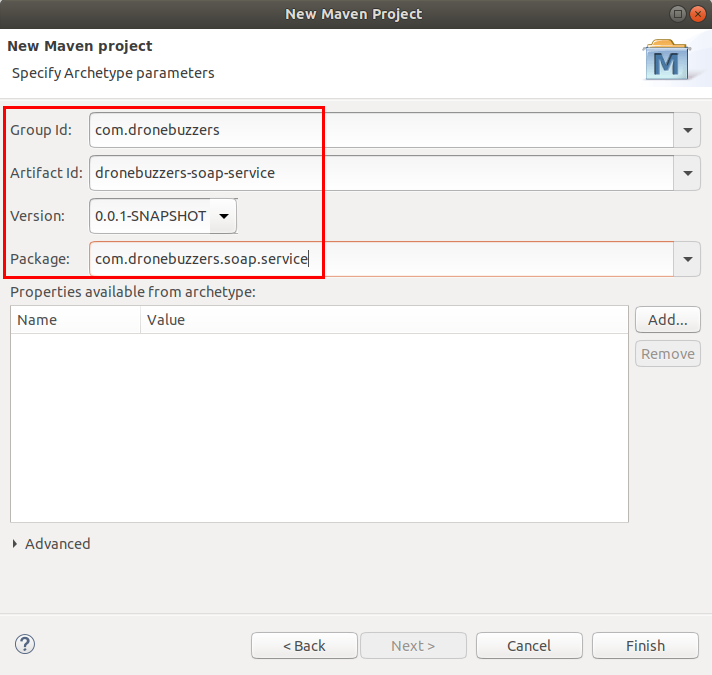
Complete the pop-up like shown below:



Click Next and set the archetype like shown below:



Click Next and complete like shown below



Click Finish

Now, the project is created.

Next, the pom.xml will be updated. The project needs to include spring-ws-core and wsdl4j as a dependency.

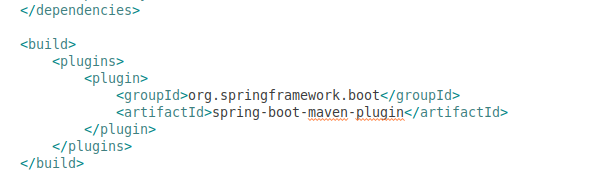
Edit pom.xml in source code view and make the following changes:

(or copy the pom.xml from <workspace>/SIGSpringBoot101/lab 7/input/pom.xml)

* Add the Spring Boot parent
* Add the Spring Boot dependencies, including those for the SOAP web service and wsdl’s
* Add the java 1.8 property



* … and add the build plugin:



Finally, delete the generated App.java class (located in package: com.dronebuzzers.soap.service)

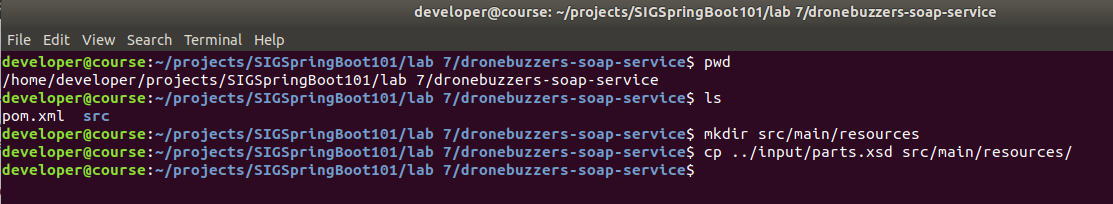
# XML schema for messages

For a SOAP web service, we need an XML schema file that defines the business objects and the request and response messages that are used in the SOAP web service.

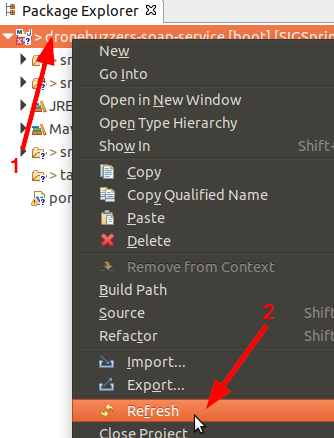
The XSD is available:

<workspace>/SIGSpringBoot101/lab 7/input/parts.xsd

You need to import it into the Eclipse project in the resources directory:



Refresh the project in Eclipse by right-clicking it and click Refresh:



Now, open the parts.xsd so you can examine the message definitions:



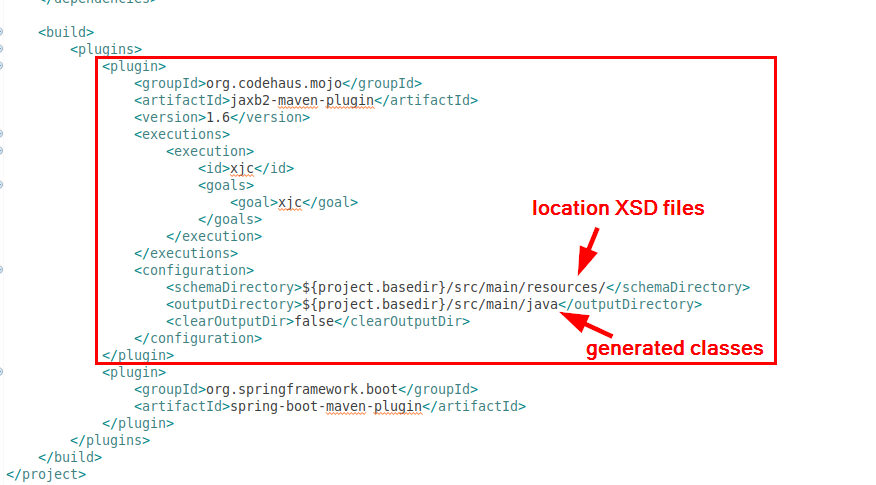
… and a bit further down the file the business object definitions:



# JAXB: Java classes for parts.xsd

In this section, we will use JACB to create the java classes for the messages and business objects defined in parts.xsd.

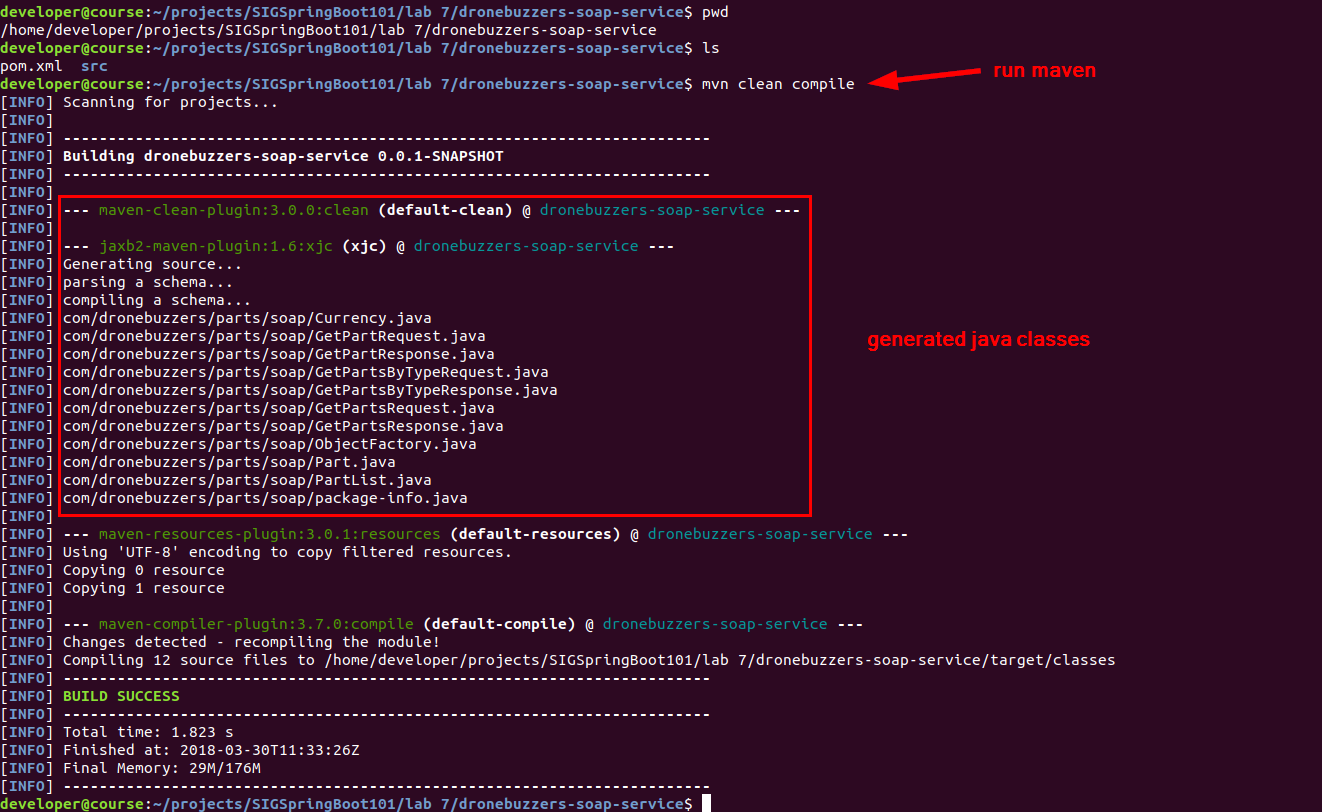
First, add the jaxb maven plugin in the pom.xml:



Either type it in by hand… or copy it from the file:

<workspace>/SIGSpringBoot101/lab 7/input/jaxb-plugin.xml

Next, generate the domain classes with the jaxb plugin. Don’t forget to save the updated pom.xml first! The command to run the jaxb plugin is mvn clean compile:



Note that the location / packages for the generated sources reflect the namespace ‘com.dronesbuzzers.parts.soap’.

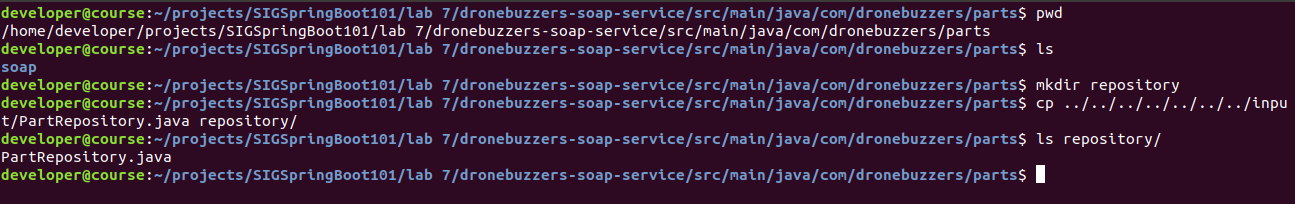
So far, all pretty straightforward jaxb stuff.

# Parts repository

As the parts web service also needs access to parts data, we need to have a parts repository. For this lab, we created a simple java class that will be used as a repository. Nothing fancy, just copy it from:

<workspace>/SIGSpringBoot101/lab 7/input/PartRepository.java

Mind to put it in the right directory: repository



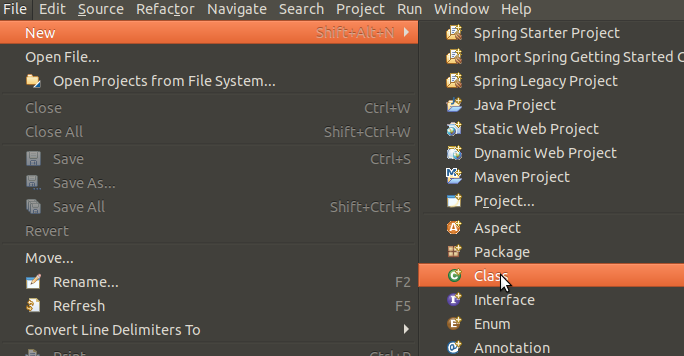
The PartRepository class has hard-coded drone parts and some methods for retrieving information about them.

Refresh the project!

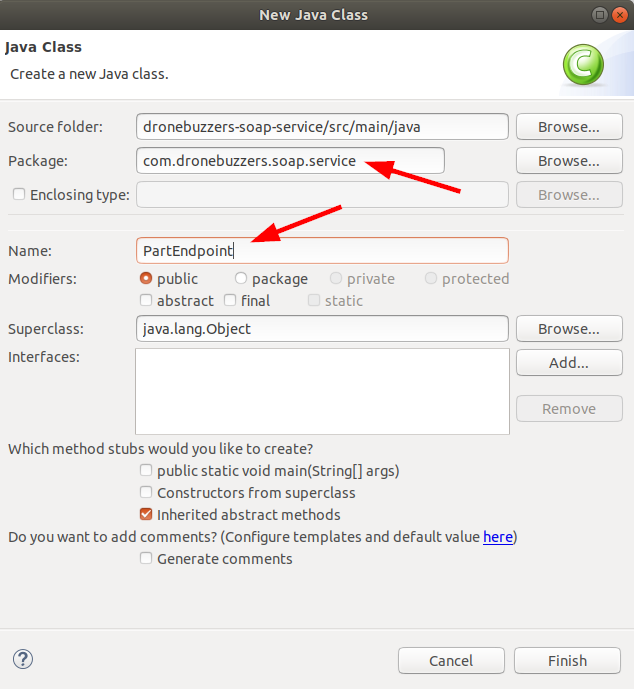
# Web service endpoint

A SOAP web service has an endpoint. It is a simple java class with some Spring Boot annotations.

Create a new java class by right-clicking the project, select New and then Class:



Complete like shown below:



Click Finish

In the resulting file, copy the contents of the file:

<workspace>/SIGSpringBoot101/lab 7/input/PartEndpoint.java

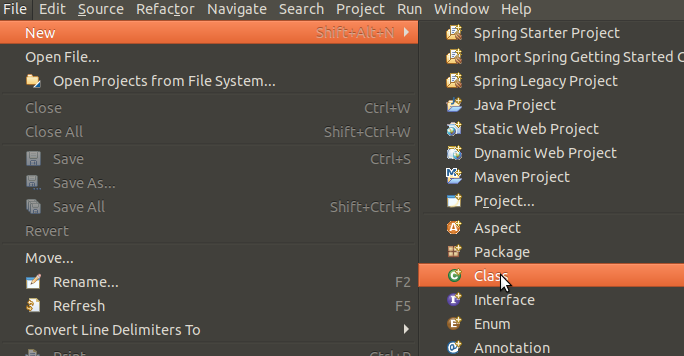
And examine the – annotations in – the code:



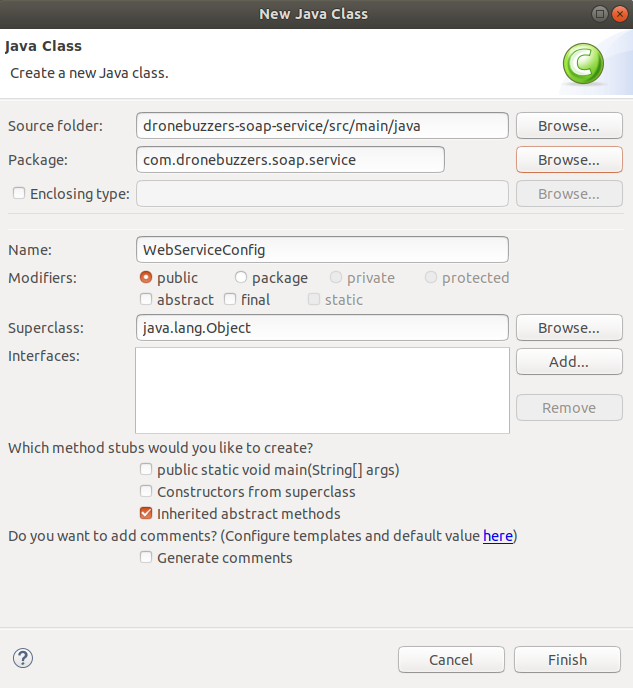
# Web service configuration

The previous section defined what the web service is doing on its endpoint. Now, we configure the web service endpoint itself.

Create a new java class by right-clicking the project, select New and then Class:



Complete like shown below:



Click Finish

In the resulting file, copy the contents of the file:

<workspace>/SIGSpringBoot101/lab 7/input/WebServiceConfig.java

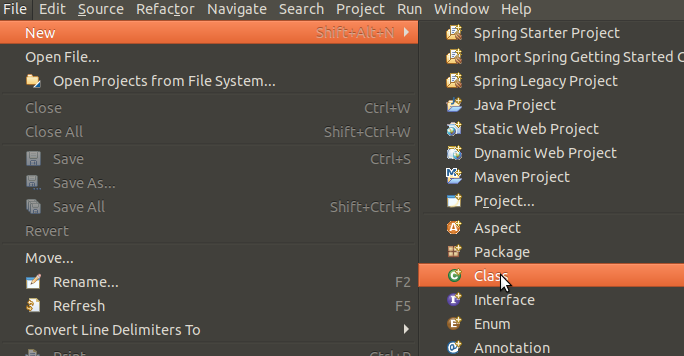
And examine the annotations in the code:



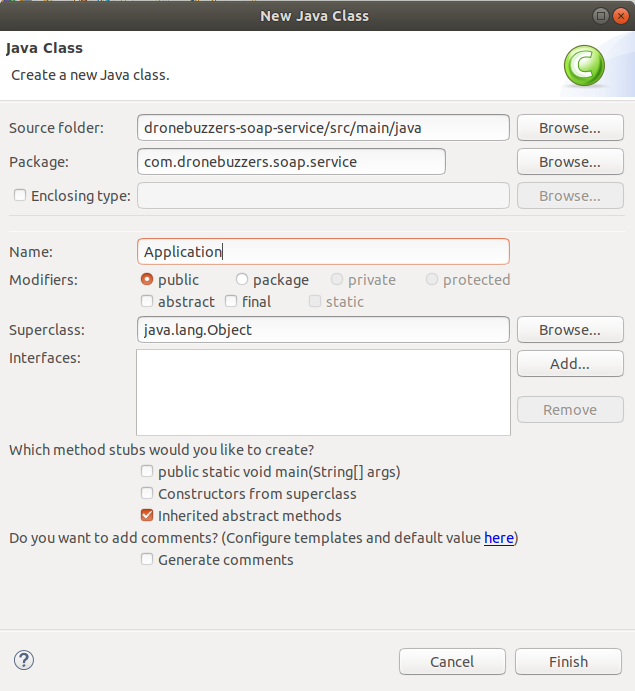
# Runnable application

In order to turn this project into a runnable application, create the class Application.

Create a new java class by right-clicking the project, select New and then Class:



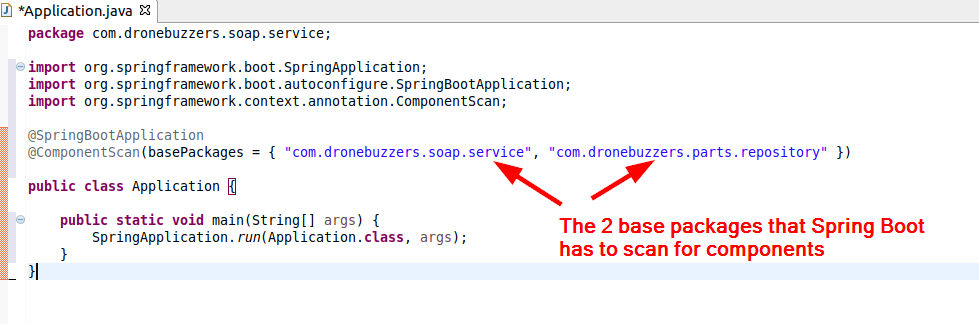
Complete like shown below:



In the resulting file, copy the contents of the file:

<workspace>/SIGSpringBoot101/lab 7/input/Application.java

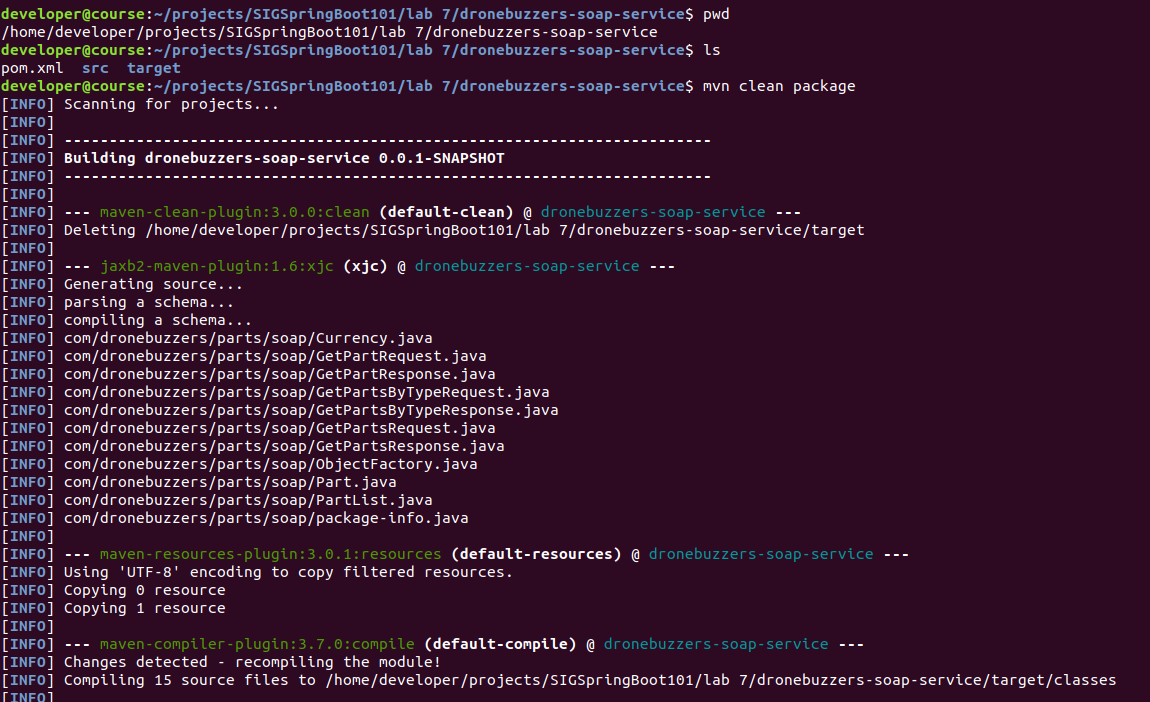
And examine the annotations in the code:



After saving all files in Eclipse, the implementation of the Soap web service is finished.

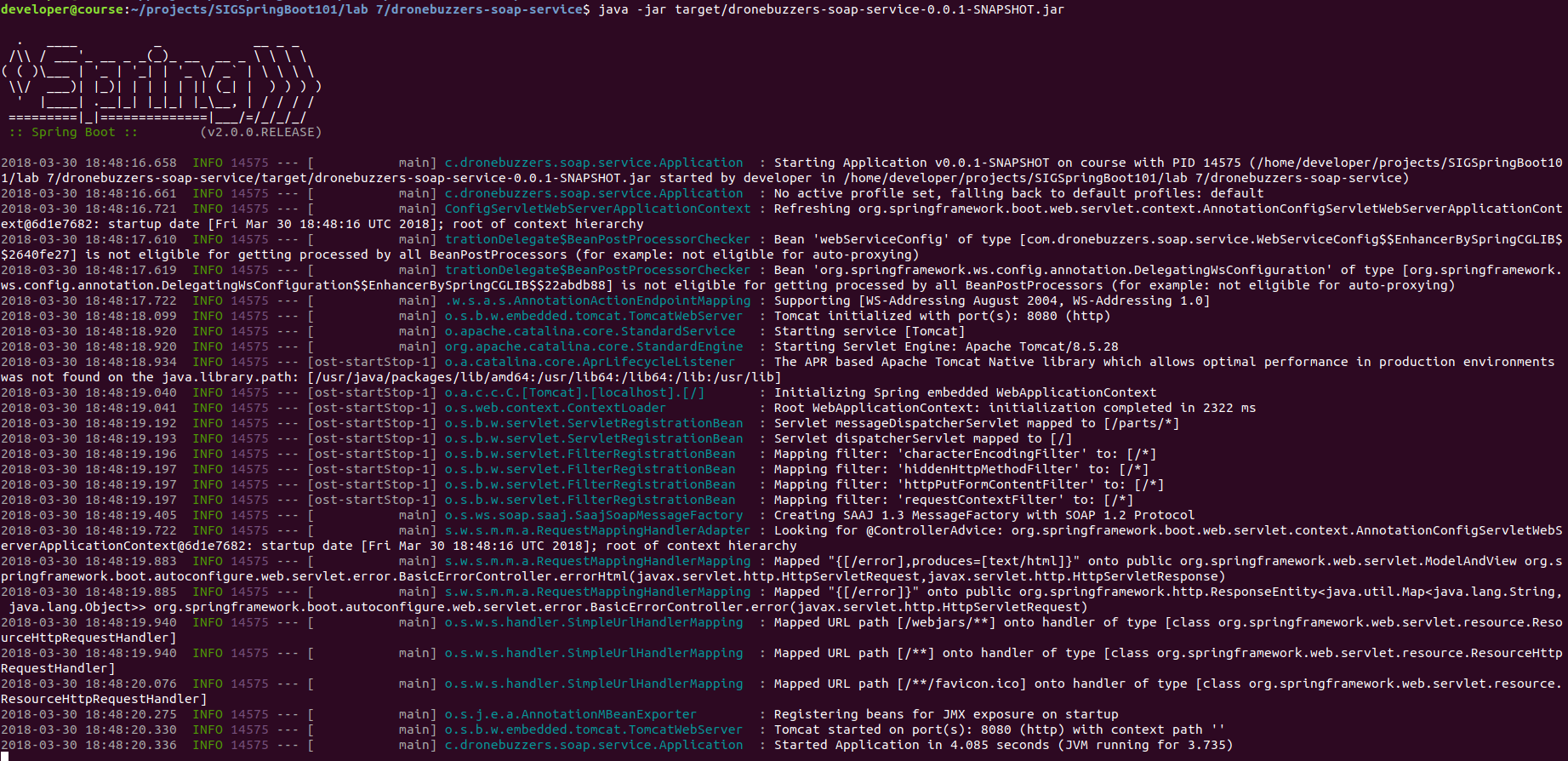
# Run and Test the Soap web service

Let’s start the web service from the command line, using mvn clean package:



Next, the web service can be started with the command:

java -jar target/dronebuzzers-soap-service-0.0.1-SNAPSHOT.jar



With the service up and running, you can have a look at the WSDL by

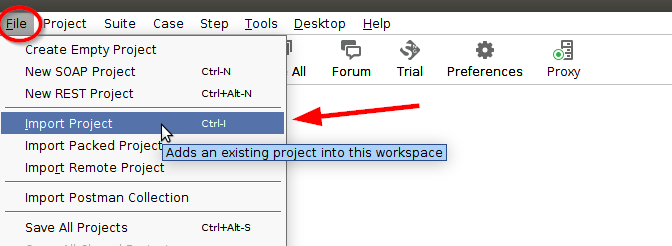
<http://localhost:8080/parts/soap/parts.wsdl>

The testing work can be done with SoapUI. Start SoapUI by clicking on 

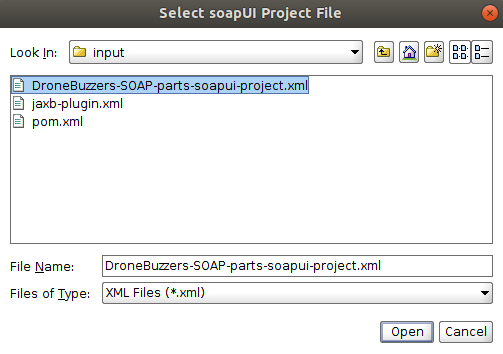
SoapUI will start. Next, import the sample project:

<workspace>/SIGSpringBoot101/lab 7/input/DroneBuzzers-SOAP-parts-soapui-project.xml

In SoapUI, click File – Import Project:



Select the sample project:



And open it. Next, the 3 operations can be used for testing the parts Soap web service:

