By Alx: *June 2019*

Soap with Spring boot - WSDL First

Goal

In this tutorial we want to show how to build a saop web serivce with spring boot. Spring boot uses Spring-WS, which allows only contract-first. Hence we need to start from a contract definition, either from a `xml schema (xsd)` or from `WSDL`. We will flow the contract-first with wsdl approach

Used technologies

JDK 1.8
Maven 3.2

WSDL First

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="someName"
    targetNamespace="http://www.wstutorial.com/ws/TutorialService" xmlns="http://schemas.xmlsoap.org
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.wstutorial.com/ws/Tutor

    <types>
        <xs:schema targetNamespace="http://www.wstutorial.com/ws/TutorialService">
            <xs:complexType name="statusCode">
                <xs:sequence>
                    <xs:element name="code" type="xs:long" />
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="TutorialType">
                <xs:all>
                    <xs:element name="id" type="xs:long" />
                    <xs:element name="name" type="xs:string" />
                    <xs:element name="author" type="xs:string" />
                </xs:all>
            </xs:complexType>

            <xs:element name="updateTutorialRequest">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="tutorialType" type="tns:TutorialType" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="updateTutorialResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="statusCode" type="tns:statusCode" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="deleteTutorialRequest">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:long" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="deleteTutorialResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="statusCode" type="tns:statusCode" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="getTutorialsRequest">
                <xs:complexType>
```

```
                <xs:sequence>
                    <xs:element name="id" type="xs:long" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>

        <xs:element name="getTutorialsResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element type="tns:TutorialType" minOccurs="0"
                        maxOccurs="unbounded" name="tutorials" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>

        <xs:complexType name="TutorialTypes">
            <xs:sequence>
                <xs:element type="tns:TutorialType" minOccurs="0"
                    maxOccurs="unbounded" name="tutorials" />
            </xs:sequence>
        </xs:complexType>

        <xs:element name="tutorialFault" type="xs:string" />
    </xs:schema>
</types>

<message name="tutorialFault">
    <part name="params" element="tns:tutorialFault" />
</message>

<message name="getTutorialsRequestMsg">
    <part name="params" element="tns:getTutorialsRequest" />
</message>
<message name="getTutorialsResponseMsg">
    <part name="params" element="tns:getTutorialsResponse" />
</message>

<message name="deleteTutorialRequestMsg">
    <part name="params" element="tns:deleteTutorialRequest" />
</message>
<message name="deleteTutorialResponseMsg">
    <part name="params" element="tns:deleteTutorialResponse" />
</message>

<message name="updateTutorialRequestMsg">
    <part name="params" element="tns:updateTutorialRequest" />
</message>
<message name="updateTutorialResponseMsg">
    <part name="params" element="tns:updateTutorialResponse" />
</message>

<portType name="TutorialServicePortType">
    <operation name="deleteTutorial">
        <input message="tns:deleteTutorialRequestMsg" />
        <output message="tns:deleteTutorialResponseMsg" />
        <fault name="fault" message="tns:tutorialFault" />
    </operation>
```

```
        <operation name="updateTutorial">
            <input message="tns:updateTutorialRequestMsg" />
            <output message="tns:updateTutorialResponseMsg" />
            <fault name="fault" message="tns:tutorialFault" />
        </operation>
        <operation name="getTutorials">
            <input message="tns:getTutorialsRequestMsg" />
            <output message="tns:getTutorialsResponseMsg" />
            <fault name="fault" message="tns:tutorialFault" />
        </operation>
    </portType>

    <binding name="tutorialServiceSOAPBinding" type="tns:TutorialServicePortType">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
        <operation name="deleteTutorial">
            <soap:operation soapAction="deleteTutorial" />
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="fault">
                <soap:fault name="fault" use="literal" />
            </fault>
        </operation>
        <operation name="updateTutorial">
            <soap:operation soapAction="updateTutorial" />
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="fault">
                <soap:fault name="fault" use="literal" />
            </fault>
        </operation>
        <operation name="getTutorials">
            <soap:operation soapAction="getTutorials" />
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="fault">
                <soap:fault name="fault" use="literal" />
            </fault>
        </operation>
    </binding>
    <service name="TutorialService">
        <port name="TutorialServicePort" binding="tns:tutorialServiceSOAPBinding">
            <soap:address
                location="http://localhost:8080/wsdlfirst/tutorialService" />
        </port>
```

```
    </service>
</definitions>
```

The wsdl document contains the five standard elements: types, message, portType, binding and service.
The wsdl provides 3 operations: deleteTutorial, updateTutorial and getTutorials

Maven file

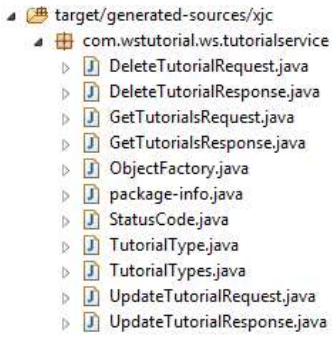
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xs
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.wstutorial.ws</groupId>
    <artifactId>soap-spring-boot</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.4.1.RELEASE</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web-services</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.jvnet.jaxb2.maven2</groupId>
                <artifactId>maven-jaxb2-plugin</artifactId>
                <version>0.13.2</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>generate</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <schemaDirectory>${project.basedir}/src/main/resources/wsdl</schemaDirectory>
                    <schemaIncludes>
                        <include>*.wsdl</include>
                    </schemaIncludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

The only used dependency is`spring-boot-starter-web-services`, it includes the needed dependencies for using Spring-WS
Maven plugin `maven-jaxb2-plugin` for generating Java sources from WSDL

- `schemaDirectory`: Directory where WSDL file can be found.
- `schemaIncludes` : Here we can specify which file should be used. In our case a WSDL file
- generateDirectory - Target directory for the generated code, In our case is unspecified, so the default generateDirectory will be used `target/generated-sources/xjc`
- generatePackage is not set, consequently the package name will be derived from the wsdl. Exactly from targetNamespace

Let's generate code

```
mvn clean compile
```



Implementing the endpoint

```
package com.wstutorial.ws;

import java.util.ArrayList;
import java.util.List;

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.wstutorial.ws.tutorialservice.DeleteTutorialRequest;
import com.wstutorial.ws.tutorialservice.DeleteTutorialResponse;
import com.wstutorial.ws.tutorialservice.GetTutorialsRequest;
import com.wstutorial.ws.tutorialservice.GetTutorialsResponse;
import com.wstutorial.ws.tutorialservice.ObjectFactory;
import com.wstutorial.ws.tutorialservice.StatusCode;
import com.wstutorial.ws.tutorialservice.TutorialType;
import com.wstutorial.ws.tutorialservice.UpdateTutorialRequest;
import com.wstutorial.ws.tutorialservice.UpdateTutorialResponse;

@Endpoint
public class TutorialServiceEndpoint {
    private static final String NAMESPACE_URI = "http://www.wstutorial.com/ws/TutorialService";


    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "updateTutorialRequest" )
    @ResponsePayload
    public UpdateTutorialResponse updateTutorial(@RequestPayload UpdateTutorialRequest request)throw
        ObjectFactory factory = new ObjectFactory();
        StatusCode code = factory.createStatusCode();
        UpdateTutorialResponse response = factory.createUpdateTutorialResponse();
        code.setCode(200);
        response.setStatusCode(code);
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "deleteTutorialRequest" )
    @ResponsePayload
    public DeleteTutorialResponse deleteTutorial(@RequestPayload DeleteTutorialRequest request)throw
        System.out.println("-->deleteTutorial<--");
        ObjectFactory factory = new ObjectFactory();
        DeleteTutorialResponse response = factory.createDeleteTutorialResponse();
        StatusCode code = factory.createStatusCode();
        code.setCode(204);
        response.setStatusCode(code);
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "getTutorialsRequest" )
    @ResponsePayload
    public GetTutorialsResponse getTutorials(@RequestPayload GetTutorialsRequest request)throws Exce
        ObjectFactory factory = new ObjectFactory();
        GetTutorialsResponse response = factory.createGetTutorialsResponse();
```

```
        List<TutorialType> tutorials = getTutorials();

        response.getTutorials().addAll(tutorials);
        return response;
    }

    private List<TutorialType> getTutorials() {
        List<TutorialType> tutorials= new ArrayList<TutorialType>();
        TutorialType tut1 = new TutorialType();
        tut1.setAuthor("John Doe");
        tut1.setId(151);
        tut1.setName("Web Service with spring boot");

        TutorialType tut2 = new TutorialType();
        tut2.setAuthor("John Doe");
        tut2.setId(152);
        tut2.setName("Web Service with spring boot");

        tutorials.add(tut1);
        tutorials.add(tut2);
        return tutorials;
    }

}
```

A simple implemenation with dummy data

Configuration

```
package com.wstutorial.ws;

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.SimpleWsdl11Definition;
import org.springframework.ws.wsdl.wsdl11.Wsdl11Definition;

@EnableWs
@Configuration
public class WebServiceConfig extends WsConfigurerAdapter {

    @Bean
    public ServletRegistrationBean messageDispatcherServlet(ApplicationContext applicationContext) {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(applicationContext);
        return new ServletRegistrationBean(servlet, "/wsdlfirst/*");
    }

    @Bean(name="tutorialService")
    public Wsdl11Definition defaultWsdl11Definition() {
        SimpleWsdl11Definition wsdl11Definition = new SimpleWsdl11Definition();
        wsdl11Definition.setWsdl(new ClassPathResource("/wsdl/TutorialService.wsdl"));
        return wsdl11Definition;
    }
}
```

Notes:

- `@EnableWs`: Provides spring web service configuration
- We define DefaultWsdl11Definition with `@Bean(name = "tutorialService")`. tutorialService will be the name of the WSDL in the URL
  - `wsdl11Definition.setWsdl`, spicify the localtion of the wsdl
- MessageDispatcherServlet will be used to handle the http requests
  - Setting *ApplicationContext* is required
  - The `ServletRegistrationBean` maps all the incoming requests with URI `/wsdlfirst/*` to the servlet
- The url of the wsdl will be: http://localhost:8080/wsdlfirst/tutorialService.wsdl

Run the SOAP Web Service

```
package com.wstutorial.ws;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class WsdlFirstApp {
    public static void main(String[] args) {
        SpringApplication.run(WsdlFirstApp.class, args);
    }
}
```

Test with soapUI

References

- maven-jaxb2-plugin

spring boot