Creating a SOAP Client Project

In this lesson, we will look at how to create a Java project with all the required dependencies for making SOAP web service calls.

We'll cover the following

Installing build tools

Please ensure JDK 8 or JRE 8 is installed on your machine as it is required by both Gradle and Maven.

`Gradle`

If you have a Macintosh/Linux machine and already have `brew` installed, you can just run the following command to install `gradle`.

```
brew install gradle
```

Else, you can manually install it by downloading the binary from Gradle Latest Release. To download version `6.3`, use this link.

After downloading the `.zip`, unzip the archive file.

`Macintosh/Linux`

Add the following lines in `~/.profile` file

```
export GRADLE_HOME=<unzipped gradle folder path>
export PATH=$PATH:$GRADLE_HOME/bin
export PATH=$PATH:$GRADLE_HOME/bin
```

Run the following command:

```
source ~/.profile
```

`Windows`

Run the following commands from the command prompt.

```
set GRADLE_HOME=<unzipped gradle folder path>
set PATH=%PATH%;%GRADLE_HOME%\bin
```

`Maven`

If you have a Macintosh/Linux machine and already have `brew` installed, you can just run the following command to install `maven`.

```
brew install maven
```

Else, you can manually install it by downloading the binary from https://maven.apache.org/download.cgi. To download version `3.6.3`, use this link.

After downloading the `.zip`, unzip the archive file.

`Macintosh/Linux`

Add the following lines in `~/.profile` file

```
export MAVEN_HOME=<unzipped maven folder path>
export PATH=$PATH:$MAVEN_HOME/bin
```

Run the following command:

```
source ~/.profile
```

`Windows`

Run the following commands from the command prompt.

```
set MAVEN_HOME=<unzipped maven folder path>
set PATH=%PATH%;%MAVEN_HOME%\bin
```

For more information on installation, please follow this link.

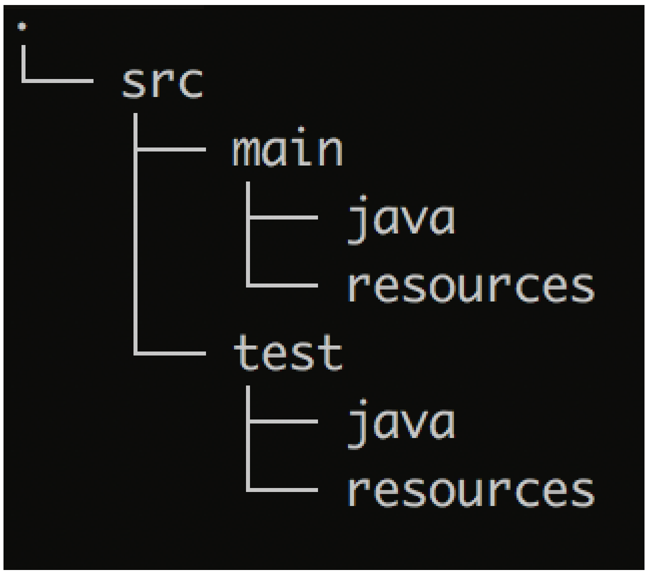
Creating the Java project

Macintosh/Linux users, please run the following commands to create the basic folder structure of the test project.

```
mkdir -p soap-client/src/{main,test}/{resources,java}
```

Windows

After running the above commands, the folder structure should look like this:



```
.
└── src
    ├── main
    │   ├── java
    │   └── resources
    └── test
        ├── java
        └── resources
```

Creating a build file

Inside the folder, we need to create build files containing the dependencies and plugin tasks. Here, we are going to use `spring-boot`, `spring-boot-starter-web-services` and `wsdl4j` libraries for creating a SOAP client.

`Gradle`

Create a file `build.gradle` and copy the content below into that:

```
plugins {
    id 'org.springframework.boot' version '2.2.2.RELEASE'
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'
    id 'java'
    id 'eclipse'
    id 'idea'
}

group = 'io.educative'
jar.baseName = 'soap-client'
jar.baseName = 'soap-client'
version = '1.0.0-SNAPSHOT'

bootJar {
    mainClassName = ''
}

sourceCompatibility = '1.8'
targetCompatibility = '1.8'

repositories {
    mavenCentral()
}

configurations {
    developmentOnly
    runtimeClasspath {
        extendsFrom developmentOnly
    }
    jaxb
}

task genJaxb {
    ext.sourcesDir = "${buildDir}/generated-sources/jaxb"
    ext.classesDir = "${buildDir}/classes/jaxb"
    ext.schema = "src/main/resources/students.wsdl"

    outputs.dir classesDir

    doLast() {
        project.ant {
            taskdef name: "xjc", classname: "com.sun.tools.xjc.XJCTask",
                    classpath: configurations.jaxb.asPath
            mkdir(dir: sourcesDir)
            mkdir(dir: classesDir)

            xjc(destdir: sourcesDir, schema: schema) {
                arg(value: "-wsdl")
                produces(dir: sourcesDir, includes: "**/*.java")
            }

            javac(destdir: classesDir, source: 1.8, target: 1.8, debug: true,
                    debugLevel: "lines,vars,source", includeantruntime: false,
                    classpath: configurations.jaxb.asPath) {
                src(path: sourcesDir)
                include(name: "**/*.java")
                include(name: "*.java")
            }

            copy(todir: classesDir) {
                fileset(dir: sourcesDir, erroronmissingdir: false) {
                    exclude(name: "**/*.java")
                }
            }
        }
    }
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web-services'
    implementation 'wsdl4j:wsdl4j:1.6.1'
    jaxb("org.glassfish.jaxb:jaxb-xjc:2.2.11")
    compile(files(genJaxb.classesDir).builtBy(genJaxb))
    compile 'org.testng:testng:7.1.0'
    compile 'org.apache.commons:commons-lang3:3.9'
}
```

`Maven`

Create a file called **pom.xml** and copy the content below into that:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.2.RELEASE</version>
        <relativePath />
    </parent>
    <groupId>io.educative</groupId>
    <artifactId>soap-client</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
```

```xml
        </dependency>
        <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.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.9</version>
        </dependency>
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>7.1.0</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>jaxb2-maven-plugin</artifactId>
                <version>2.5.0</version>
                <executions>
                    <execution>
                        <id>xjc</id>
                        <goals>
                            <goal>xjc</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <sourceType>wsdl</sourceType>
                    <sources>
                        <source>${project.basedir}/src/main/resources/student
s.wsdl</source>
                    </sources>
                </configuration>
            </plugin>
        </plugins>


    </build>
</project>
```

In the next lesson, we will learn how to use this web service client to make web service calls over HTTP.