Sending Requests using SOAP Client

In this lesson, we will learn how to send requests and receive responses using the SOAP client that we created in the previous lesson. We will also learn how to validate the response.

We'll cover the following

Creating `BaseTest` class

Since we use the `Spring` framework and `Annotations` for defining beans, we need to load them before doing anything. Here, we will use the `TestNG` annotation `@BeforeSuite` to load the beans using `AnnotationConfigApplicationContext` which reads all the `Spring` annotated classes like `@Configuration`, `@Service`, etc.

In our case, we have annotated the `WebServiceClient` class with `@Configuration` for the bean that needs to be loaded.

We will create `BaseTest` which will be extended by all the test classes so that we need not duplicate the `@BeforeSuite` method that contains loading of beans. This will be executed once per test suite and initializing the `WebServiceTemplate` in `@BeforeClass` will be executed for every test class that is extending `BaseTest`. We will mark `BaseTest` as `abstract` to disallow the explicit initialization of the class.

It also has the `SERVICE_URL` that holds the location where the web service is hosted.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationConte
xt;
import org.springframework.ws.client.core.WebServiceTemplate;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeSuite;
import com.fasterxml.jackson.dataformat.xml.XmlMapper;
```

```
public abstract class BaseTest {


    protected static ApplicationContext CONTEXT;

    protected WebServiceTemplate webServiceTemplate;

    protected static final String SERVICE_URL = "http://ezifyautomationlabs.co
m:6566/educative-soap/ws";

    protected static final Logger LOG = LoggerFactory.getLogger(BaseTest.class
);

    @BeforeSuite
    public void init() {
        if (CONTEXT == null) {
            CONTEXT = new AnnotationConfigApplicationContext(io.educative.soap
.WebServiceClient.class);
        }
    }

    @BeforeClass
    public void initTemplate() {
        webServiceTemplate = CONTEXT.getBean(WebServiceTemplate.class);
    }

    protected void printResponse(Object response) {
        try {
            LOG.info("printing response '{}' => \n{}", response.getClass().get
Name(),
                    new XmlMapper().writerWithDefaultPrettyPrinter().writeValu
eAsString(response));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

To learn more about `TestNG` annotations, please follow this link.

Creating `TestClass`

Here, we are creating a test class to test the `GetStudents` API. All the initialization code is already contained in `BaseTest` and marked with `TestNG` configuration annotations. Now, we have the required things to make the web service call.

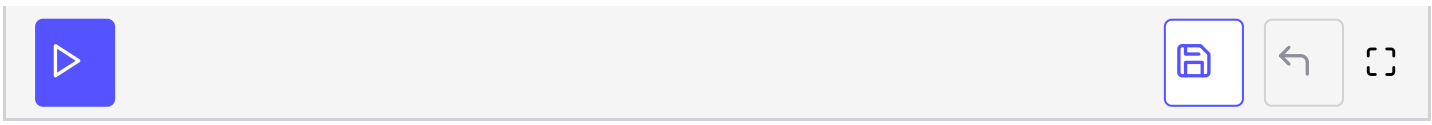
To make the web service call, we need to know the service URL (where the web service is hosted), the request format (or class), and the response format (or class).

All these information can be found in `students.wsdl`.

After building the test project, all the request and response formats (or classes) mentioned in `students.wsdl` will be generated and available for us to use.

```
import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;
import static org.testng.Assert.assertTrue;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.ws.client.core.WebServiceTemplate;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.Test;

import com.fasterxml.jackson.dataformat.xml.XmlMapper;

import io.educative.soap_automation.GetStudentsRequest;
import io.educative.soap_automation.GetStudentsResponse;

public class TestSOAP extends BaseTest {

	@Test
	public void testGetStudents() {

		GetStudentsRequest request = new GetStudentsRequest();
		request.setGender("male");

		GetStudentsResponse response = (GetStudentsResponse) webServiceTemplate.marshalSen
				request);

		assertNotNull(response, "GetStudentsResponse is null");

		assertTrue(!response.getStudents().isEmpty(), "students list is empty");

		assertTrue(response.getStudents().get(0).getGender().equalsIgnoreCase(request.getG
				"students must be having the same gender as sent in request - " +
					+ response.getStudents().get(0).getGender());

		printResponse(response);
	}

}

abstract class BaseTest {

	protected static ApplicationContext CONTEXT;

	protected WebServiceTemplate webServiceTemplate;

	protected static final String SERVICE_URL = "http://ezifyautomationlabs.com:6566/educative

	protected static final Logger LOG = LoggerFactory.getLogger(BaseTest.class);
```

```
        @BeforeSuite
        public void init() {

                if (CONTEXT == null) {
                        CONTEXT = new AnnotationConfigApplicationContext(io.educative.soap.WebServ
                }
        }

        @BeforeClass
        public void initTemplate() {
                webServiceTemplate = CONTEXT.getBean(WebServiceTemplate.class);
        }

        protected void printResponse(Object response) {
                try {
                        LOG.info("printing response '{}' => \n{}", response.getClass().getName(),
                                        new XmlMapper().writerWithDefaultPrettyPrinter().writeValu
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```



The web service is hosted at `http://ezifyautomationlabs.com:6566/educative-soap/ws` and saved as `SERVICE_URL` in `BaseTest` as shown in line 49.

In the code above:

- We have created a `TestNG` test method `testGetStudents` as shown in line 22. This method contains code for creating the request object `GetStudentsRequest` as shown in lines 24 & 25.
- Using `WebServiceTemplate`'s `marshalSendAndReceive` method that internally transforms the Java object to XML structure that the SOAP web service understands, we make the web service call and get the transformed xml to `GetStudentsResponse` object in response from the server as shown in line 27.
- We are asserting or validating whether the response is ***not null***, the students list in response is ***not empty*** and contains the same ***gender*** as sent in the request as shown in lines 30, 32 & 34 respectively.

In this lesson, we learned how to send a request using the client and validate the response. In the next lesson, we will learn how to handle authentication in SOAP.