Handling Authentications

In this lesson, we will learn how to make SOAP web service calls that are secured using plain text username and password.

We'll cover the following

We have already learned that SOAP can have its own implementation of security using `WS-Security`. The following is a sample SOAP header to pass plain text username and password:

Sample SOAP authorization header

```
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401
-wss-wssecurity-secext-1.0.xsd" wsse:mustUnderstand="1">
  <wsse:UsernameToken>
     <wsse:Username>testuser</wsse:Username>
     <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">testpass</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>
```

In the sample `SOAP` header above, we pass plain text username and password. Please take a look at the password type, as passwords can be of 3 different types – `PasswordDigest`, `PasswordText`, `UsernameToken`. To know more, please follow this link.

Sending authorization header in the request

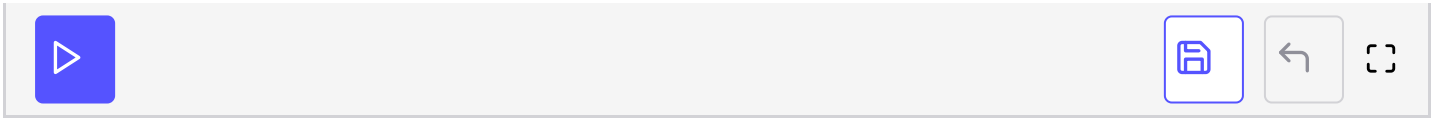
For the demonstration, we are considering a plain `PasswordText`. We have already seen how to use `WebServiceTemplate` to send the request and receive the response in the ***Sending requests using SOAP client*** lesson.

Here, we will see how to use the same to pass the additional authorization header in the request.

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

import java.io.IOException;

import javax.xml.soap.Name;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPException;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPHeaderElement;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.transform.TransformerException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.ws.WebServiceMessage;
import org.springframework.ws.client.core.WebServiceMessageCallback;
import org.springframework.ws.client.core.WebServiceTemplate;
import org.springframework.ws.soap.saaj.SaajSoapMessage;
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 testGetStudentsWithAuthHeader() {

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

		GetStudentsResponse response = (GetStudentsResponse) webServiceTemplate.marshalSen
				request, new AuthHeaderRequestCallback("testuser", "testpass"));

		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);
	}

}

class AuthHeaderRequestCallback implements WebServiceMessageCallback {
```

```
        private final String username;
        private final String password;


        public AuthHeaderRequestCallback(String username, String password) {
                this.username = username;
                this.password = password;
        }

        @Override
        public void doWithMessage(WebServiceMessage message) throws IOException, TransformerExcept

                try {

                        SaajSoapMessage saajSoapMessage = (SaajSoapMessage) message;

                        SOAPMessage soapMessage = saajSoapMessage.getSaajMessage();

                        SOAPPart soapPart = soapMessage.getSOAPPart();

                        SOAPEnvelope soapEnvelope = soapPart.getEnvelope();

                        SOAPHeader soapHeader = soapEnvelope.getHeader();

                        Name headerElementName = soapEnvelope.createName("Security", "wsse",
                                        "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
                        SOAPHeaderElement authHeader = soapHeader.addHeaderElement(headerElementNa

                        SOAPElement usernameToken = authHeader.addChildElement("UsernameToken", "w

                        SOAPElement usernamElement = usernameToken.addChildElement("Username", "ws

                        usernamElement.addTextNode(username);

                        SOAPElement passwordElement = usernameToken.addChildElement("Password", "w

                        passwordElement.addTextNode(password);

                        soapMessage.saveChanges();
                } catch (SOAPException e) {
                        throw new RuntimeException(e);
                }
        }
}

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();
                }
        }
}
```



In the above code sample, we pass an instance of `AuthHeaderRequestCallback` that is basically an implementation of `WebServiceMessageCallback`. `WebServiceMessageCallback` contains the method `doWithMessage(WebServiceMessage message)` that allows us to modify the request before making the web service call.

In our case, we need to add the authorization header in the same structure with appropriate namespaces as described in the ***Sample SOAP authorization header***. Once we are done with it, we are all set to send the request with the authorization header.

```
GetStudentsResponse response = (GetStudentsResponse) webServiceTemplate.marsha
lSendAndReceive(SERVICE_URL, request, new AuthHeaderRequestCallback("testuser"
, "testpass"));
```

In the code snippet above, we pass an object of `AuthHeaderRequestCallback` initialized with the authorization username and password along with the other parameters to call the web service and receive the response.

On a successful validation of authorization header at the server-side, we should get a valid response.

On passing invalid credentials, we should see an exception with the following message:

```
org.springframework.ws.soap.client.SoapFaultClientException: com.sun.xml.wss.i
mpl.WssSoapFaultException: Authentication of Username Password Token Failed; n
ested exception is com.sun.xml.wss.XWSSecurityException: com.sun.xml.wss.impl.
WssSoapFaultException: Authentication of Username Password Token Failed
```

That basically imitates the following SOAP response:

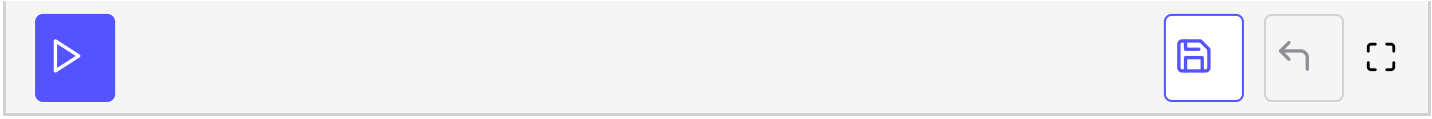
```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Client</faultcode>
         <faultstring xml:lang="en">com.sun.xml.wss.impl.WssSoapFaultExceptio
n: Authentication of Username Password Token Failed; nested exception is com.s
un.xml.wss.XWSSecurityException: com.sun.xml.wss.impl.WssSoapFaultException: A
uthentication of Username Password Token Failed</faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Have a look at this by running the code below to understand what happens when we pass invalid credentials in the authorization header.

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

import java.io.IOException;

import javax.xml.soap.Name;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPException;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPHeaderElement;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.transform.TransformerException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.ws.WebServiceMessage;
import org.springframework.ws.client.core.WebServiceMessageCallback;
import org.springframework.ws.client.core.WebServiceTemplate;
import org.springframework.ws.soap.saaj.SaajSoapMessage;
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 testGetStudentsWithInvalidAuthHeader() {
```

```
        public void testGetStudentsWithInvalidAuthHeader() {

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

                try {
                        GetStudentsResponse response = (GetStudentsResponse) webServiceTemplate.ma
                                request, new AuthHeaderRequestCallback("testuser", "testpassinvali
                } catch(Exception e) {
                        e.printStackTrace();
                }
        }

}

class AuthHeaderRequestCallback implements WebServiceMessageCallback {

        private final String username;
        private final String password;

        public AuthHeaderRequestCallback(String username, String password) {
                this.username = username;
                this.password = password;
        }

        @Override
        public void doWithMessage(WebServiceMessage message) throws IOException, TransformerExcept

                try {

                        SaajSoapMessage saajSoapMessage = (SaajSoapMessage) message;

                        SOAPMessage soapMessage = saajSoapMessage.getSaajMessage();

                        SOAPPart soapPart = soapMessage.getSOAPPart();

                        SOAPEnvelope soapEnvelope = soapPart.getEnvelope();

                        SOAPHeader soapHeader = soapEnvelope.getHeader();

                        Name headerElementName = soapEnvelope.createName("Security", "wsse",
                                        "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
                        SOAPHeaderElement authHeader = soapHeader.addHeaderElement(headerElementNa

                        SOAPElement usernameToken = authHeader.addChildElement("UsernameToken", "w

                        SOAPElement usernamElement = usernameToken.addChildElement("Username", "ws

                        usernamElement.addTextNode(username);

                        SOAPElement passwordElement = usernameToken.addChildElement("Password", "w

                        passwordElement.addTextNode(password);

                        soapMessage.saveChanges();
                } catch (SOAPException e) {
                        throw new RuntimeException(e);
                }
        }
}

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();
                }
        }
}
```



In this lesson, we learned how to pass the authorization header in the SOAP header. In the next lesson, we will learn how to set a proxy in the SOAP client.