SoapHeader : s'authentifier proprement a un WebService SOAP

Une solution est d'utiliser l'en-tete d'une requête SOAP. Pour cela on va tout d'abord créer un nouveau type qui hérite de SoapHeader, ce type contiendra les informations de connexion :

```
public class CredentialHeader : SoapHeader
{
    public CredentialHeader()
        : base()
    { }

    public CredentialHeader(int Pid)
        : base()
    {
        this._Pid = Pid;
    }

    private int _Pid;

    public int Pid
    {
        get { return _Pid; }
        set { _Pid = value; }
    }

}
```

Ensuite nous rajoutons l'attribute *SoapHeaderAttribute* sur la *WebMethod* qui prend en argument le nom d'une variable public du type que l'on vient de créer.

Ainsi lorsqu'un client enverra une requête SOAP contenant les informations de connexion dans l'en-tête, notre variable *Authentication* sera automatiquement renseigné, nous pouvons alors tester si le pid pour le suivit de notre correlation.

```
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class WebService : System.Web.Services.WebService
{

    public CredentialHeader Authentication;

    [WebMethod]
    [SoapHeader("Authentication")]
    public Product GetProduct(int productID)
    {
        if (Authentication.Pid == 5)
            return new Product(productID, "title", "description", 100);
        else
            throw new
System.Security.Authentication.InvalidCredentialException();
    }

}
```

Notre requête SOAP devra donc contenir les informations de connexions dans le header. Cela se traduit par une requête de ce type :

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
     <CredentialHeader xmlns="http://tempuri.org/">
       <Pid>int</Pid>
     </CredentialHeader>
   </soap:Header>
  <soap:Body>
    <GetProduct xmlns="http://tempuri.org/">
      <productID>int</productID>
    </GetProduct>
  </soap:Body>
</soap:Envelope>
```

Pour envoyer les paramétres dans le header d'un requete SOAP nous référençons classiquement notre paramétre en faisant un "*Add Web Reference*" dans Visual Studio. Puis lorsque nous instancions notre classe proxy, nous avons la possibilité de renseigner une propriété *CredentialHeaderValue* de type *CredentialHeader :*

```
WS.WebService ws = new WS.WebService();
ws.CredentialHeaderValue = new WS.CredentialHeader();
ws.CredentialHeaderValue.Pid = 5;
Console.WriteLine(ws.GetProduct(10).Price);
```

SOAP with Attachments API for Java

SOAP API for Java (SAAJ) is used mainly for the SOAP messaging that goes on behind the scenes in JAX-RPC and JAXR implementations. Secondarily, it is an API that developers can use when they choose to write SOAP messaging applications directly rather than use JAX-RPC. The SAAJ API allows you to do XML messaging from the Java platform: By simply making method calls using the SAAJ API, you can read and write SOAP-based XML messages, and you can optionally send and receive such messages over the Internet (some implementations may not support sending and receiving). This chapter will help you learn how to use the SAAJ API.

The SAAJ API conforms to the Simple Object Access Protocol (SOAP) 1.1 specification and the SOAP with Attachments specification. The SAAJ 1.2 specification defines the `javax.xml.soap` package, which contains the API for creating and populating a SOAP message. This package has all the API necessary for sending request-response messages. (Request-response messages are explained in SOAPConnection Objects.)

Le processus fonctionne en 5 étpes:

1. creation d'un connection SOAP
2. Creation d'un message SOAP
3. le remplir
4. l'envoyer
5. Recevoir la reponse

I/ Connection

```
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.SOAPConnection;

public class SOAPTip {

   public static void main(String args[]) {

      try {

         //First create the connection
         SOAPConnectionFactory soapConnFactory =
                             SOAPConnectionFactory.newInstance();
         SOAPConnection connection =
                              soapConnFactory.createConnection();



         //Close the connection
         connection.close();

        } catch(Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

II/creation du message

```
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.SOAPConnection;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPBody;

public class SOAPTip {

   public static void main(String args[]) {

      try {

         //First create the connection
         SOAPConnectionFactory soapConnFactory =
                             SOAPConnectionFactory.newInstance();
         SOAPConnection connection =
                              soapConnFactory.createConnection();

         //Next, create the actual message
```

```
         MessageFactory messageFactory = MessageFactory.newInstance();
         SOAPMessage message = messageFactory.createMessage();

         //Create objects for the message parts
         SOAPPart soapPart =     message.getSOAPPart();
         SOAPEnvelope envelope = soapPart.getEnvelope();
         SOAPBody body =         envelope.getBody();


         //Close the connection
         connection.close();

        } catch(Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

III/ Remplissage du message

```
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPElement;

public class SOAPTip {

   public static void main(String args[]) {

      try {
...
         //Create objects for the message parts
         SOAPPart soapPart =     message.getSOAPPart();
         SOAPEnvelope envelope = soapPart.getEnvelope();
         SOAPBody body =         envelope.getBody();

        //Populate the body
        //Create the main element and namespace
        SOAPElement bodyElement =
                  body.addChildElement(envelope.createName("getPrice" ,
                                                           "ns1",
                                        "urn:xmethods-BNPriceCheck"));
        //Add content
        bodyElement.addChildElement("isbn").addTextNode("0672324229");

        //Save the message
        message.saveChanges();


        //Check the input
        System.out.println("\nREQUEST:\n");
        message.writeTo(System.out);
        System.out.println();


         //Close the connection
         connection.close();

        } catch(Exception e) {
            System.out.println(e.getMessage());
```

```
        }
    }
}
```

IV/Envois du message

```
public class SOAPTip {

   public static void main(String args[]) {

...
         //Check the input
         System.out.println("\nREQUEST:\n");
         message.writeTo(System.out);
         System.out.println();

        //Send the message and get a reply

        //Set the destination
        String destination =
            "http://services.xmethods.net:80/soap/servlet/rpcrouter";
        //Send the message
        SOAPMessage reply = connection.call(message, destination);

         //Close the connection
         connection.close();
...
    }
}
```

V/ Reception de la reponse

```
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.Source;

import javax.xml.transform.stream.StreamResult;

public class SOAPTip {

   public static void main(String args[]) {

      try {

...
         //Send the message
         SOAPMessage reply = connection.call(message, destination);

        //Check the output
        System.out.println("\nRESPONSE:\n");
        //Create the transformer
        TransformerFactory transformerFactory =
                            TransformerFactory.newInstance();
        Transformer transformer =
                        transformerFactory.newTransformer();
```

```
        //Extract the content of the reply
        Source sourceContent = reply.getSOAPPart().getContent();
        //Set the output for the transformation
        StreamResult result = new StreamResult(System.out);
        transformer.transform(sourceContent, result);
        System.out.println();

         //Close the connection
         connection.close();
...
     }
}
```