

Generating Data Using a HTTPS Call

The following example demonstrates how an authenticated user is able to log onto the Repertoire Server from the application through the REST API to generate data using a datasource.

The full set of code can be downloaded [here](#).

The parameters required to be parsed in are:

1. The url of the data source on the server, e.g.
"*https://localhost:8443/data/Elixir/ER_Exercises/Application-Integration/samples/datasources/FruitSales.ds*"
2. The output path and filename for the rendered file
3. User name and password

Within the Java class environment, import the following Java references:

```
import java.io.BufferedInputStream;  
import java.io.FileOutputStream;  
import java.io.InputStream;
```

To talk to the server via the REST API add the following .jar files from the "/lib" directory of the Repertoire Server installation:

```
commons-codec-1.3.jar  
commons-httpclient-3.0.1.jar  
commons-logging.jar
```

and import them into the class file:

```
import org.apache.commons.httpclient.Credentials;  
import org.apache.commons.httpclient.Header;  
import org.apache.commons.httpclient.HttpClient;  
import org.apache.commons.httpclient.UsernamePasswordCredentials;  
import org.apache.commons.httpclient.auth.AuthScope;  
import org.apache.commons.httpclient.methods.GetMethod;  
import org.apache.commons.httpclient.params.HttpClientParams;
```

For logging purposes, the `getResponseHeaders()` method is used which is similar to the `IJobInfo` found in the `ERSClient`:

```
Header[] headers = method.getResponseHeaders();
for (Header header : headers) {
    System.out.println( header.getName());
    System.out.println( header.getValue());
}
```

The following are explanations and walkthroughs of the code. The full Java class can be downloaded [here](#).

```
//Opens a connection and sends the credentials
//to the Repertoire Server
try {
    HttpClient client = new HttpClient();
    Credentials defaultCreds = new UsernamePasswordCredentials(username,password);
    client.getState().setCredentials(new AuthScope("localhost",8443,AuthScope.ANY_REALM),
defaultCreds);
    HttpClientParams params = client.getParams();
    params.setCredentialCharset(CredentialCharset);
    params.setAuthenticationPreemptive(true);

    //Instantiate a new GET method and redirect to
    //to the target URL
    GetMethod method = new GetMethod( url );
    method.setFollowRedirects( true );

    // Execute the GET method
    int statusCode = client.executeMethod( method );
    if( statusCode != -1 ) {
        System.out.println( "Reading file" );
        InputStream is = method.getResponseBodyAsStream();
        BufferedInputStream bis = new BufferedInputStream( is );
        FileOutputStream fos = new FileOutputStream( filename );
        byte[] bytes = new byte[ 8192 ];
        int count = bis.read( bytes );
        while( count != -1 && count <= 8192 ) {
            System.out.print( "-" );
            fos.write( bytes, 0, count );
            count = bis.read( bytes );
        }
    }
}
```

```
}  
if( count != -1 ) {  
    fos.write( bytes, 0, count );  
}  
  
//Information regarding the request is return as the response header  
Header[] headers = method.getResponseHeaders();  
for (Header header : headers) {  
    System.out.println( header.getName());  
    System.out.println( header.getValue());  
}  
  
//Once completed the File and Buffered  
//output streams are closed  
fos.close();  
bis.close();  
method.releaseConnection();  
System.out.println( "\nDone" );  
}  
}  
catch( Exception e ) {  
    e.printStackTrace();  
}  
}
```