Elixir Programmming Interfaces

Release 3.5.0

Elixir Technology Pte Ltd

Elixir Programmming Interfaces: Release 3.5.0

Elixir Technology Pte Ltd

Published 2014

Table of Contents

List of Figures

Chapter 1
Programming Interfaces

Introduction

You can use the Ambience APIs to extract, transform and load data between Ambience and your other applications.

The ERS client API was originally developed for Elixir Repertoire and has been supported since version 5.0. To ease the transition to Elixir Ambience, the same core functions have been made available so that old client codes can more easily work with the latest releases and do not need to be re-developed.

Users who are creating new clients should use HTTP/REST APIs directly.

All of these API libraries require at least Java 7 to run. Ensure that the version of Java on your system is at least 7.0 for these libraries to run.

Chapter 2
ERS Client API

The ERS Client API allows you to write your own JVM-based programs which can call into Ambience to render reports and generate datastores.

This API can be used to migrate data between older applications such as Elixir Report 4.0 and Ambience.

ERS Client Example

This example will render the standard Master-Detail RML sample into PDF as `tmp/output.pdf`. It will also output the number of pages rendered (showing the JobInfo API) and dumps the complete JobInfo so you can see all the information.

To run this example:

1. Make sure that Ambience is listening on its default port - 8080. If you have changed the Ambience listening port, then change it in the code also.

2. Create a new Java project.

3. Copy the following jars from the Ambience `lib` folder to the `lib` folder of your project.

   - commons-io-2.4.jar
   - commons-logging-1.1.1.jar
   - elx-arch-3.5.0.jar
   - elx-ers2-client-3.5.0.jar
   - elx-jdom-3.5.0.jar
   - elx-json-3.5.0.jar
   - httpclient-4.2.1.jar
   - httpcore-4.2.2.jar
   - httpmime-4.2.1.jar
   - logback-classic-1.0.13.jar
   - logback-core-1.0.13.jar
   - paranamer-2.1.jar
   - scala-library-2.10.4.jar
   - slf4j-api-1.7.5.jar

   The ERS Client is in `elx-ers2-client-3.5.0.jar`, the rest are supporting jars.

4. Create a file called `JavaRender.java` and enter the following code:

```
package demo;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;

import com.elixirtech.ers2.client.ERSClient;
import com.elixirtech.report2.runtime.IJobInfo;

public class JavaRender {

public final static void main(String[] args) throws Exception {
ERSClient c = new ERSClient("localhost", 8080, "eno",
 "admin", "sa");

try {
c.connect();

ByteArrayOutputStream baos = new ByteArrayOutputStream();

IJobInfo jobInfo = c.renderReport("/ElixirSamples/Report/RML/
 Master-Detail Report.rml", "application/pdf", baos,
  new java.util.Properties());

byte[] bytes = baos.toByteArray();

File output = new File("tmp/output.pdf");

output.getParentFile().mkdirs();

Files.copy(new ByteArrayInputStream(bytes),output.toPath(),
 StandardCopyOption.REPLACE_EXISTING);
System.out.println("Report with " +
 jobInfo.getLong(IJobInfo.PAGE_COUNT) +
  " pages rendered to tmp/output.pdf");

System.out.println(jobInfo.toString());
}

finally {
c.close();
    }
  }
}
```

5. Save and run the file. The output should be the following:

```
Report with 7 pages rendered to tmp/output.pdf
 [byte-size=98085, elapsed-time=418,
 log-file=/User/admin/logs/tmp-0000000009.json,
 mime-type=application/pdf,
 page-count=7, record-count=117,
 result=/Temp/admin/tmp-0000000007.pdf,
 status-code=1]
```

The output PDF is in `tmp/output.pdf`.

You can also use the ERS Client to generate data into a DataStore.

The API is almost exactly the same as the old Repertoire client jar. The only changes are as follows:

1. The constructor supports one more parameter - the domain name. For backwards compatibility, you can leave the domain name out (use the original API args) and "eno" will be used as the domain name by default.

2. The filesystem API is not supported:

   ```
       public synchronized IFileSystem getFileSystem(String fs)
   ```

   This requires duplicating the whole legacy `IFileSystem`, `IFileObject` tree which is quite different in DaCapo.

   The DaCapo REST API is more suitable for accessing this information, as it provides much more powerful functionality.

3. Secure mode API:

   ```
   public void setSecure(File keystore, String keystorePassword,
    File truststore, String truststorePassword)
   ```

   This is now handled internally if the server uses https. There is no support for client certificates in this release. Only server certificates are supported.

   As long as your code does not call these functions, it should run exactly the same as before.

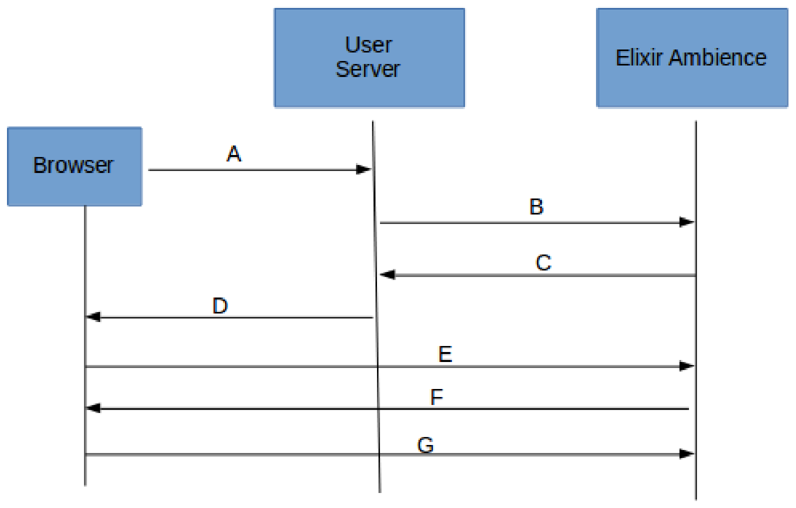
Chapter 3
Single Sign On Sample

Using Ambience APIs for Single Sign On

The following example demonstrates the use of Ambience APIs to perform Single Sign On (SSO) to Ambience from a third party application.

The diagram in Figure 3.1, “Single Sign On Workflow” depicts the workflow:

Figure 3.1. Single Sign On Workflow



- A. Browser connects to user server.
- B.User server POSTS to `/elx/sso/[domain]/[user]` with the authenticators - username and password.
- C. Ambience checks that authenticator has `SSOAuthentication` permissions and that [user] is not admin. If these conditions are true, Ambience then returns an `[sso-token]`.
- D. User server sends a REDIRECT message back to the browser, sending it to `/elx/sso/[sso-token]`. This can also include a `?ret=XXX` target URL.
- E. Browser GETs `/elx/sso/[sso-token]?ret=XXX`.
- F. Ambience checks valid `[sso-token]` and adds a session cookie to the reply. It then sends a reply (with the cookie) as part of a REDIRECT to XXX (the return value), or to the default landing page if the return value is not defined.
- G. Browser visits the regular Ambience page (XXX) and is already authenticated by the cookie.

The following Java code sample demonstrates this functionality.

To run the sample, you need to compile it as a Java servlet and put it in a Java web server.

You can also re-implement the same logic in any other modern language, such as C#, Ruby or Python.

⚠

As a security consideration, do not use `admin` as the SSO Authenticator. It is only done for simplicity in this demo.

```
package com.elixirtech.sample.sso;

import java.io.DataInputStream;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.StatusLine;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

@SuppressWarnings("serial")
public class SSODemo extends HttpServlet {

  final static String url = "http://localhost:8080/elx/sso";

  @Override
  public void doGet(HttpServletRequest request,
     HttpServletResponse response)
     throws ServletException, java.io.IOException {

    String domain = "eno";

    // Ambience user with SSOAuthentication permission
    String ssoAuthName = "admin";
    String ssoAuthPassword = "sa";

    // user we want to sign-on
    String username = "test";

    DefaultHttpClient client = new DefaultHttpClient();

    HttpPost sso = new HttpPost(url + "/" + domain + "/" +
      username);

    List<NameValuePair> nameValuePairs =
      new ArrayList<NameValuePair>(2);

    nameValuePairs.add(new BasicNameValuePair("username",
        ssoAuthName));
    nameValuePairs.add(new BasicNameValuePair("password",
      ssoAuthPassword));

    sso.setEntity(new UrlEncodedFormEntity(nameValuePairs));

    sso.setHeader("ContentType",
      "application/x-www-form-urlencoded");

    HttpResponse reply = client.execute(sso);

    StatusLine status = reply.getStatusLine();
    int code = status.getStatusCode();

    if (code < 400) {
      int length = (int)reply.getEntity().getContentLength();

      byte[] bytes = new byte[length];

      DataInputStream is =
       new DataInputStream(reply.getEntity().getContent());

      is.readFully(bytes);
      is.close();

      String token = new String(bytes,"UTF-8");

      System.out.println("token = " + token);

      response.sendRedirect(url + "/" + token +
         ret(request));

      return;
    }

    else {
      response.sendError(code,status.getReasonPhrase());
    }
  }

  private String ret(HttpServletRequest request) {
    String ret = request.getParameter("ret");

    if (ret==null || ret.trim().length()==0) return "";

    else {

      // prevent header splitting

      System.out.println("set ret to " + ret);

      return "?ret=" + ret.trim().replace("\r", "").
         replace("\n", "");
    }
  }
}
```