Repertoire Client Java Guide

Ambience and Repertoire 2023+

Overview

Given a zip of the Repertoire Client, this will guide you through setting it up and running some examples to ensure everything is working.

There are two demo files that are equivalent that have been provided: JavaDemo and ScalaDemo. For this example you only need JavaDemo.

This assumes that you have:

- ElixirSamples installed on either an Ambience or Repertoire 2023 or later
- An IDE, such as IntelliJ (community edition) as that is the one used in the guide

Use the individual functions from the example source as a starting point for writing your own integrations.

Sample Functions

- testDS1(); – generating data from a datasource using the new API
- testOldDS(); – generating data from a datasource using the old API
- testRml1(); – rendering RML to Pdf using the new API
- testRml2(); – rendering RML to XlsX using the new API
- testOldRml(); – rendering RML to XlsX using the old API
- testDocX1(); – rendering DocX to DocX using the new API
- testOldDocX(); – rendering DocX to DocX using the old API
- testDocX2(); – rendering DocX to Pdf using the new API
    - This depends on Aspose functionality which is licenced separately
- testDSWithAPIToken(); – generating data from a datasource using API Token
    - This depends on adding an API Token on the server

Ambience / Repertoire Server Setup

For name/password authentication you need to enable login from non-web interfaces via "resource-owner-password". Add this line to your configuration file:

```
elixir.sso.server.resource-owner-password-enabled = true
```

You may already have this, if you have configured Repertoire GUI.

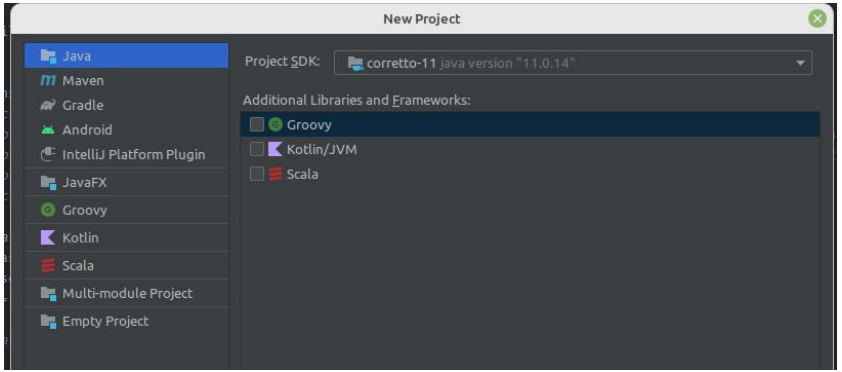
For API Token authentication, you will need to add an API Token via the web interface (described later)

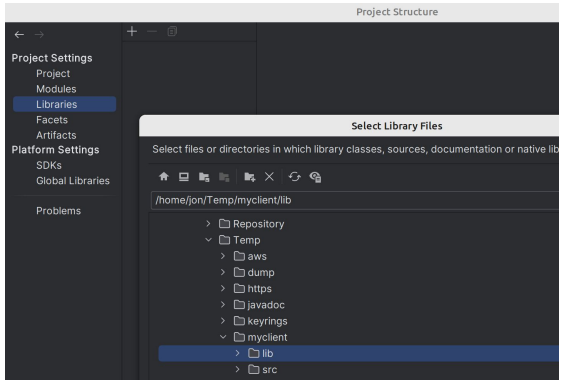
Create a Project

In IntelliJ create a new project from under file and set the project to use java 11+ before following the other steps of the wizard.

Within that new project under src create a new folder called demo. This will be where we place the demo file later.

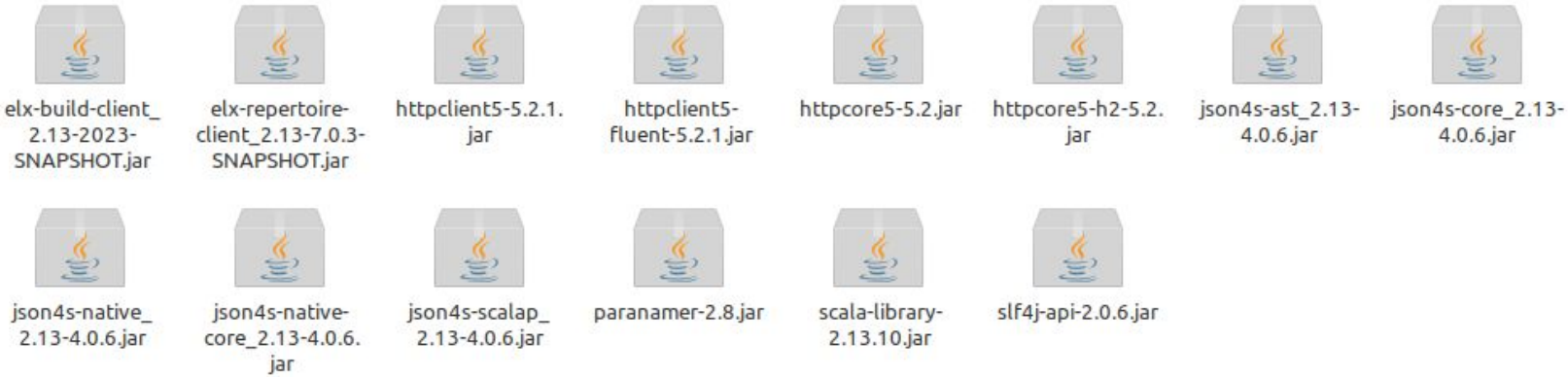
In this new project, under project settings, select a folder to be used as the lib folder in the next step.





Contents of lib folder

All the jars from the zip lib folder should be placed into your project lib folder. Note that the version numbers are baseline and may be later versions than shown.



These jars are kept separate to make independent update easier should there be a security vulnerability in future.

Demolib folder

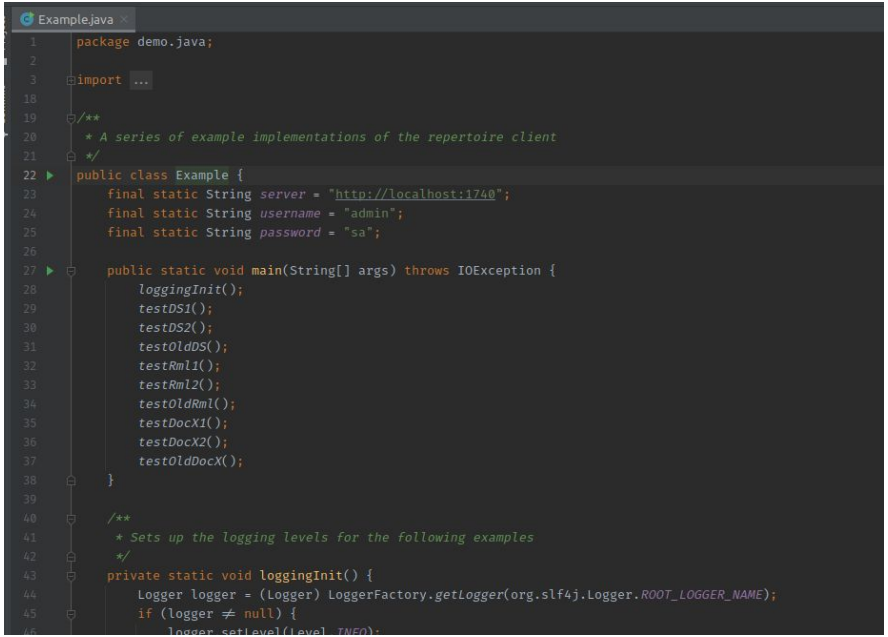
In addition to the jars from the lib folder, there is a separate demolib folder. The jars inside are kept here and separate from the others because these are only used by the demo files and not by the RepertoireClient itself.

These should also be placed in the same location as the other jars to run the demos but are not necessary for further usage of the API.

Setting up examples

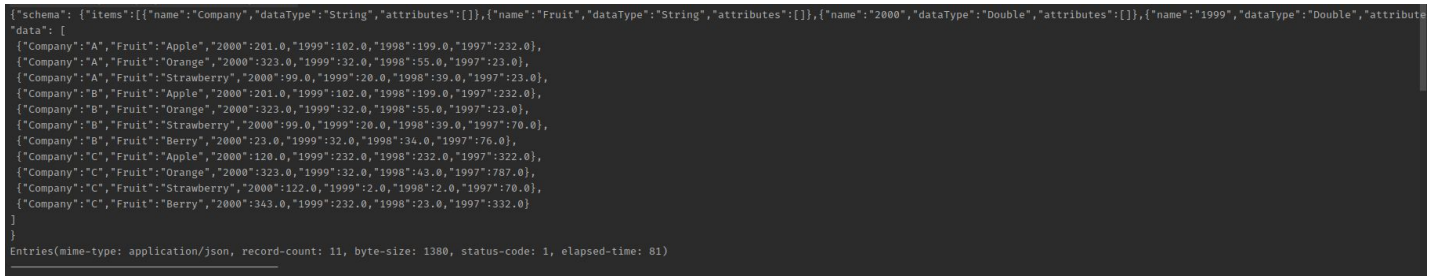
On the path sample/demo you will find a file named JavaDemo. Place this file in your project source tree under the folder "demo" and open it in your IDE.

Replace the server, username and password details with your own.

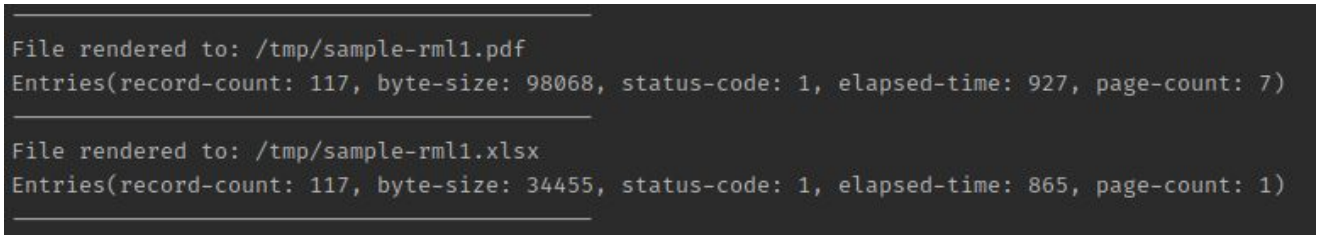


Running examples

With the server running, run the main in Example. The output for the first function should look like this.



The functions that render a payload to a folder will print the path to the rendered file in the console.



Examples Using API Token

The new RepertoireClient allows for the use of two different forms of authentication. Basic, which uses a username and password as demonstrated in the earlier examples and token which uses an API Token

One of the functions in the example is commented out - this function depends on an API Token which you need to set up in the Ambience/Repertoire server and then include within the sample source.

The function is initially commented out so the basic test can be run without having to configure anything on the server. The following section deals with setting up the token and function to test that this functionality works.

Setup an API Token in the Server

On your server, go to the API Tokens under administration and add a token with Add Cookie enabled. Save the token for use in the next step.

API Token	User	Description	Last Access	Add Cookie	Enabled	Expire At
c532b968-fd1e-4b02-b065-48b122def059	admin		2023-02-02 15:46:57	✓	✓	

Testing API Tokens

Add the API token into the source code here:

```
/*private static void testDSWithAPIToken() {
    String apiToken = "c532b968-fd1e-4b02-b065-48b122def059"; //Insert your token here
```

Uncomment the function testDSWithAPIToken at the end of the file and in the main function at the top. You can then run the test as before from the main.

You will see extra output now at the end of the console that should be identical to the output of the first function.

Response Codes

The server will return one of our internal status codes. Provided is a table of the codes and their meaning.

Status Code	Message
1	Status Ok - Bytes have been returned
4	Exception Logged - A logged exception on the server, check server logs for further details
8	Unable to Complete - An unexpected exception occurred. Check server logs of further details