General overview of L0 API

Use cases

Layer0 API is a relatively simple way to:

1) Replay your own file format with Bookmap.
2) Connect Bookmap to your own datasource in real time (both for receiving data and trading).

Doing that requires basic Java knowledge, however (1) is possible even without any Java knowledge at all using any programming language if you don't mind converting files before playing instead of integrating format support into Bookmap platform.

Code examples

Layer0ApiDemo eclipse project provides 4 demo classes with detailed comments that illustrate most typical scenarios.

1) **DemoTextDataReplayProvider** - load simple text format into Bookmap. As an option you can just generate a file in this format and feed it to bookmap using this module. This enables using any language for conversion to Bookmap format
2) **DemoGeneratorReplayProvider** - mimics a replay provider, but instead of loading data this provider generates it. Example of setting order queue position and displaying simple indicators using legacy API. Intended for simple transition from Recorder API, at some point in the future will be replaced by Layer2-based solution.
3) **DemoExternalRealtimeProvider** - example of custom realtime data provider. This one only generates random data for illustrational purposes, but you can use it to build your own connectivity.
4) **DemoExternalRealtimeTradingProvider** - extends **DemoExternalRealtimeProvider** providing trading capability. Simulates limit orders over the generated data. Shows how to build your own provider for connecting to platform with trading support.

Javadoc

Javadoc is available inside bm-l1api-javadoc.jar typically located in "C:\Program Files\Bookmap\lib" (location might differ depending on path selected during installation). It contains documentation for all levels of API, but for L0 API it's mostly sufficient to only look inside velox.api.layer0 package.

It’s highly advised to read javadoc for **ExternalLiveBaseProvider** and **ExternalReaderBaseProvider** - extending those classes will provide some parts of the logic already implemented for you.

Configuration files

In order for Bookmap to load your module it should be added to a configuration file inside C:\Program Files\Bookmap\lib\UserModules\L0
There are 2 files, **external-live-modules.txt** and **external-reader-modules.txt**
First one contains list of live connectivity module (currently only 1 module at a time is supported), and second one contains a list of replay modules (unlimited number).
Files can contain comment lines (start with #) and lines in following format:
<full-class-name> <path-without-spaces>
Path can point to either folder or .jar file with classes, whatever is more convenient.
For example, line in **external-live-modules.txt** can look like this:
velox.api.layer0.live.DemoExternalRealtimeTradingProvider D:\Layer0ApiDemo\bin

Using modules

After modules are configured those are used similarly to Bookmap internal functionality

Replay

Select a file the same way you would open Bookmap file. Note, that since extension of your file will likely be different from .bmf, you should first type “*” into the file name field and press enter - this will reset file type filter and will allow you to load any type of file.

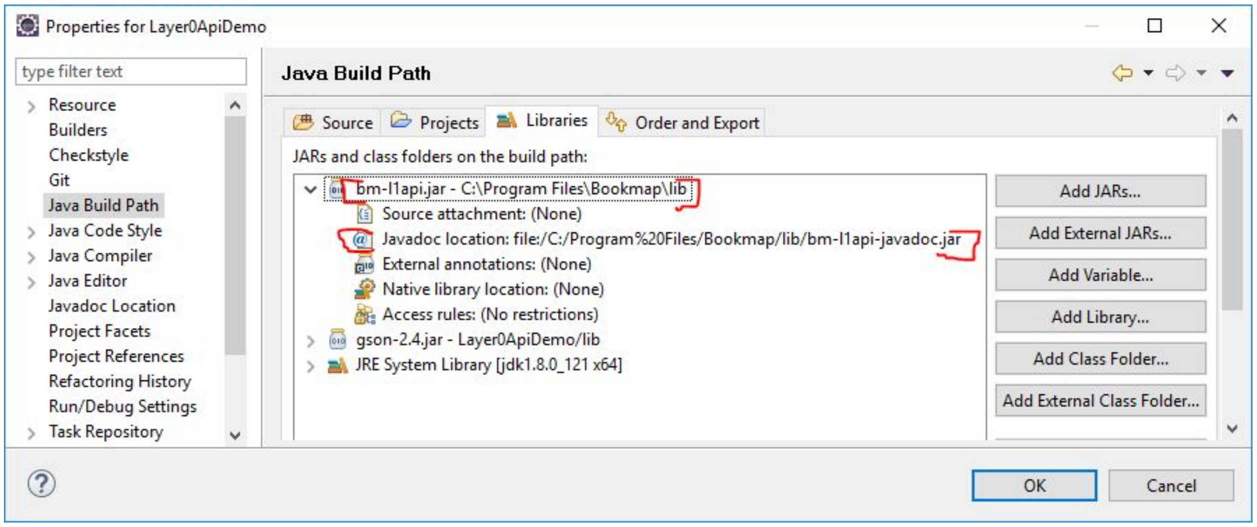
Live

Configure a connection in “Connections” menu selecting “External” as platform. Fill username, password fields, connect the same way you would do with any other connection.
If you don’t need username or password just leave those empty - actual use of that data will be defined by your code.
If you need to place some additional data like server address, you can place it into one of those fields. E.g. user “u1” at server “127.0.0.1” could be set as u1@127.0.0.1 inside “username” field.

Environment setup

Example project

You can use any IDE with Java support or even use javac compiler directly, but for this guide we will use Eclipse.
Make sure you have Bookmap installed (using default path will make things a bit more simple), then open example **Layer0ApiDemo** project.
Make sure that in Project->Properties -> Java Build Path on Libraries tab you have bm-l1api.jar and javadoc paths set accordingly to what was selected when installing Bookmap.



Java Build Path

JARs and class folders on the build path:

```
bm-l1api.jar - C:\Program Files\Bookmap\lib
    Source attachment: (None)
    Javadoc location: file:/C:/Program%20Files/Bookmap/lib/bm-l1api-javadoc.jar
    External annotations: (None)
    Native library location: (None)
    Access rules: (No restrictions)
gson-2.4.jar - Layer0ApiDemo/lib
JRE System Library [jdk1.8.0_121 x64]
```

This should be enough for the code to get compiled. You can either include “bin” folder using config file or pack it to jar file and connect that file.

Creating your own project

For your own project you just need to add bm-l1api.jar to it. It’s also recommended but not required to configure Javadoc location.

Running from IDE

While you can run Bookmap using the shortcut, it will be easier to run it from IDE and this will also provide additional debugging capabilities.

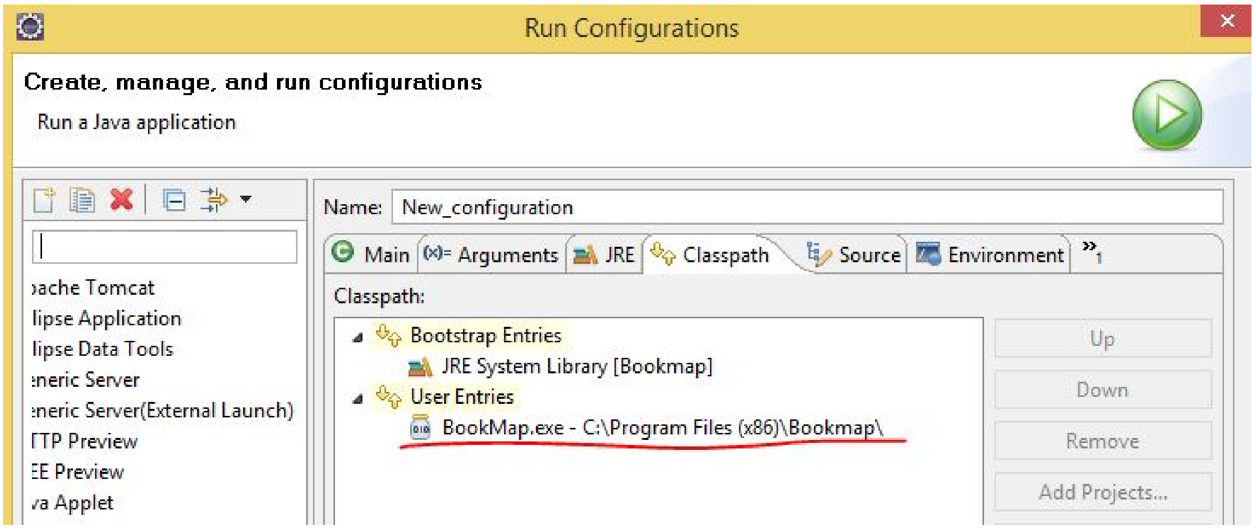
1) Configure your IDE to run bookmap. For this configure ide to
- add C:\Program Files\Bookmap\Bookmap.exe to classpath

- start velox.ib.Main
- Add -Xmx1000M to VM options
- Use C:\Bookmap\Config as working directory

2) Adjust configuration files in C:\Program Files\Bookmap\lib\UserModules\L0 to load the class you are developing from the folder where it's compiled to (in eclipse it's "bin" folder)

3) Now you should be able to use this configuration to start bookmap with your strategy being loaded after it starts.

Note that Bookmap.exe should be added as User entry (in Eclipse run configuration):



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: New_configuration

Classpath:

- Bootstrap Entries
  - JRE System Library [Bookmap]
- User Entries
  - BookMap.exe - C:\Program Files (x86)\Bookmap\

When you start Bookmap this way you will be able to debug your code from IDE, including capability to edit code in debug mode without restarting the application.