Project Darknet – Developer's manual

Johann Beleites, Ibtehaj Nadeem, Farah Kiran Patel,
Josh Treon, Augustin Zidek

2014-03-02

1 Purpose of this text

This text is short manual explaining how to extend the Project Darknet system. It documents only the parts of the system that are exposed to the developers extending the system. The rest of the system is thoroughly documented in Javadoc, which is provided together with the source code.

2 Resources used in the system

The system depends on various external libraries:

- **RestFB** – Java wrapper on top of Facebook RESTful API. The library requires Facebook access tokens. The easiest way to obtain a token (valid for a couple of hours) from Facebook is to go to `https://developers.facebook.com/tools/explorer` and copy and paste the string from the 'access token' field into the input prompt displayed by this method.
- **Twitter4J** – Twitter Java access library.
- **HSQLDB** – Java SQL database.
- **OpenCSV** – Used for parsing the CSV files obtained from Spektrix.
- **iTextPDF** – Used for `pdf` report generation.
- **Reflections** – Used for dynamic class loading from a package. Reflections needs **Googe Guava**, **javassist** and **slf4j** to run.

The system also uses couple of folders (must be located in the same directory as the `jar` files):

- `doc/` – Location of the Javadoc.
- `documents/` – Location of the documentation.
- `lib/` – Location of libraries' `jar` files.
- `log/` – Location of the main program log. You can use the log by using class `LoggerFactory`.
- `res/` – Location of the database configuration file.
- `src/` – Location of the source code.
- `storage/` – Location of the image storage. This is where all the harvested images should be stored. Use `ImageStorage` class to use it.
- `twitter4j/` – Location of Twitter4J libraries.

3 Adding new `SecondaryDataCollectors`

Adding new `SecondaryDataCollector` is rather simple. The only thing that is necessary to create is a new class `MyCollector` that extends the `SecondaryDataCollector`. Moreover, it has to implement interface `Runnable` and provide `run()` method hence.

The new collector has to implement method `setup(List<Individuals>)` which takes a list of individuals as an argument. This tells the collector what set of individuals it should harvest data for when the `run()` method is invoked.

The collector has to be place in the package `uk.ac.cam.darknet.backend`. This is because internally, Java reflection is used to get all collector classes from this package.

4 Adding new `Effects`

Adding new `Effect`s is similar to adding new collectors. Again, only a single class per effect needs to be added. The class must extend the `Effect` and hence it must provide three methods:

1. `setup(String[])` which sets up the effect with custom arguments,
2. `getSetupArgDescriptions()` which is used by the other parts of the system to tell the user what the expected custom arguments of `setup()` are,
3. `execute(Show)` method that executes the effect on the given show (that determines uniquely a set of individuals).

5 Entry points

The entry points into the system are `backend/DataCollector` for the data collection phase and `gui/EffectExecutionGUI` for the effect DJ system.

6 Expected input `csv` file format

The file should have these (11) columns: Customer ID, First Name, Last Name, Email Address, Event Name, Event Date/Time, Seat, Price, Ticket Type, Date Confirmed, Sales Channel.

All attributes are strings, hence they have surrounded by quotation marks. The format of the dates should be `dd/MM/yyyy HH:mm:ss`.

An example line:
`"I-TS00-0022","John","Smith","john.smith@test.com","Macbeth","11/07/2011 19:30:00","A24","15.00","Adult","02/06/2011 14:51:28","Counter"`

7 Database

The relational database library used in for this project is HyperSQL (version 2.3.1), licensed under the BSD license. Whilst it would be technically possible to use a different database system, we do not recommend this as we have not tested the system with other SQL implementations.

HyperSQL supports two connection modes: in-process and server mode. Our system requires server mode connections. This means that a database server has to be hosted, possibly on the same machine which executes the system[1]. This allows multiple instances of the system (be it front-end or back-end) to execute concurrently. Chapter 14 of the HyperSQL documentation (see below) explains the process of setting up such a database server. We recommend that an encrypted connection is used if the data is to be held on an external server (see chapter 13 in the user guide). After setting up the database, ensure that:

- The user that will be used for establishing connections has the DBA role (i.e. it can create and drop tables and has the usual administrator rights). The default user created when the database is first created is such a user.
- The user's default schema is set to the one intended to be used with the system. If no schema is created specifically for the system, then the default schema will be used.
- The user has a strong password.

The following SQL commands, to be submitted when the database is first created, present one possible way of achieving the above:

```
1  -- Create a schema for the system.
2  -- Can skip this and use default schema instead.
3  CREATE SCHEMA <schema name>;
4
5  -- Create the user to be used to connect to the database.
6  -- Can skip this and use default user instead.
7  CREATE USER <user name> PASSWORD <password> ADMIN;
8
9  -- If a schema was created, must set the user to use
10 -- the created schema by default.
11 -- The name of the default user is usually 'SA'.
12 ALTER USER <user name> SET INITIAL SCHEMA <schema name>;
13
14 -- Must do this once when the database is created!
15 SET DATABASE SQL UNIQUE NULLS FALSE;
```

Once the database is created, a configuration file (`res/dbconfig.txt`) must be edited. The file's contents should look something like the following:

```
# Connection to the database.
# Edit this file as described in the database documentation.
prefix=jdbc:hsqldb:hsql:
host=127.0.0.1
port=5000
alias=DARKNET
username=theusername
password=thepassword
```

[1]Notice however that some data (such as pictures from the Facebook data collector) may be stored locally on the machine that executed the collector. In such a case the database can still be hosted externally, but any effects relying on the local data must be executed on the same machine which produced the data.

The prefix should not be changed (without a very good reason of doing so). The `host`, `port`, `username` and `password` fields should be self-explanatory (`host` is the IP address of the server hosting the database, `port` is the port number on which the server listens for connections, `username` and `password` are the authentication details as set up when the database was created). The `alias` is the name by which the server identifies the database (as a server can host multiple databases at the same time). This name should have been specified when the server was set up as explained in the HyperSQL user guide.

When the configuration file has been set up correctly, and the server is running, the system will be able to connect to the database. For a detailed documentation of HyperSQL, we refer to the online user manual. [2]

[2] At the time of writing, it can be found at: `http://hsqldb.org/doc/2.0/guide/index.html`