SOEN 6441 Advanced Programming Practices

Assignment #1

Due date (Moodle Submission): Tuesday, March 13
Due date (Lab Demo): March 14 & 15

This is the first of two programming assignments in our course. Note that the assignments are not 'stand-alone', but rather build on top of each other to form a bigger project.

The assignments focus on applying the concepts covered in the lectures, such as functional programming, lambdas, streams, asynchronous programming with futures, and actor-based programming, in the context of a web application.

The assignments have to be developed based on the *Play Framework*, a full-stack reactive framework for the JVM, which is also used in large-scale commercial deployments by companies such as LinkedIn or Walmart.

Play Application: "TweetAnalytics". Your goal is to develop a simple web interface that takes a keyword (input text field at top of the page) and then displays the latest 10 tweets on the keyword. Adding another keyword will add a second set of 10 tweets, and so on. Each tweet must contain the name of the Twitter account it was sent from, hyperlinked to a page that shows the user's profile information and the last 10 tweets from that user.

You have to use the (free) Twitter API to access the tweet information, see https://developer.twitter.com.

Note: for this assignment, you do not need to stream updates to the user interface. In other words, the front-end page is static until a "refresh" is triggered.

Coding guidelines. Your submission must satisfy the following requirements:

a) Your application must be based on the Play framework and it must be possible to build and run it using the standard `sbt run` command. Any required third-party libraries must be automatically resolved through `sbt`.

b) Document all your classes and methods with *Javadoc* (including private methods).

c) Your controller actions must be *asynchronous*, using Java 8's `CompletionStage/CompletableFuture`. Do not use `.get()/.join()` to block for the future's results anywhere in your code (only exception are unit tests, see below).

d) Create JUnit tests for all your classes. You must have tests for every method in your controller and every controller action, as well as any additional classes you wrote. Compute your test coverage with JaCoCo. Notes: *(i)* within a test, you can use `.get()/.join()` to wait for a result; *(ii)* do **not** call the life Twitter API from a unit test, use a mock class for testing instead.

d) You must include a `README.txt` file with: *(i)* Group member information (names, IDs); *(ii)* Contributions of each group member (classes, methods – also include the names in the relevant Javadoc `@author` tags); *(iii)* any technical notes about your app (compiling, running, etc.).

Submission. You must submit your code electronically on Moodle by the due date (late submission will incur a penalty, see Moodle for details). You must also demo your code to the marker in one of the lab sessions (time slots for the demo will be reserved through Moodle). Note that all group members must be present for the demo and ready to answer questions about the code.