SOEN 6441 Advanced Programming Practices

Assignment #2

Due date (Moodle Submission): Tuesday, April 10
Due date (Lab Demo): April 11 & 12

This is the second (and final) part of our course project, building on the results of your first assignment.

Play Application: "Reactive TweetAnalytics". Your goal is to modify your Play application from Assignment #1 to make it *reactive*, i.e., stream live updates to the user interface. Thus, instead of having a static list of Tweets for a search keyword, your application will now dynamically show any new incoming tweets. You only have to make the search page reactive; the profile page can stay static (but you are free to make it reactive as well).

You **must** implement this reactive behavior as an asynchronous *server push* solution, using WebSockets and Akka Actors.[1] You can base your implementation on the Play 'Reactive Stocks' example application shown in the lecture (note: you can, but do not have to, use Akka Streams in your solution).

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.).

e) For testing your Twitter API, you must use dependency injection with Google Guice (if you are already using a mocking framework, like *Mockito*, this is also acceptable for A2).

f) You must create one actor for each user (WebSocket connection). Create separate actors for managing the Tweets and Twitter Users (profiles). Define appropriate message classes. You must have JUnit tests for your ActorSystem, using the Java Akka Testkit shown in the lectures.

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 during your lab session. Note that all group members must be present for the demo and ready to answer questions about the code.

[1] In particular, a solution where you simply refresh (parts of) a page using, e.g., AJAX on the client side will not be accepted.