Programming - Part I

Ricard Solé Casas

February 10, 2018

Foreword

The source code for this report and app can be found on Github[1]. A production version of the application can be installed from Google's Play Store at http://bit.ly/ada-zooify. The minimum Android version required is 5.0 (Lollipop).

Some of the decisions taken in building this app do not follow the original suggested guidelines. TODO.

Declaration

I confirm that the submitted coursework is my own work and that all material attributed to others (whether published or unpublished) has been clearly identified and fully acknowledged and referred to original sources. I agree that the College has the right to submit my work to the plagiarism detection service. TurnitinUK for originality checks.

[1] https://github.com/rcsole/zoo-coursework

Contents

Chapter 1

Design Choices

I've made some choices for this solution that require some justification. The project guidelines suggested using the Java Swing[1] framework, which was meant to be replaced by JavaFX[2]. However, both technologies are quite outdated. The former is partially deprecated and the latter doesn't appear to be well maintained, documentation is scarce, and Oracle has discontinued support for the editor tools. After some research I came to the conclusion that, as of June 2017, the community has pivoted towards using Java for the business logic and persistence through a web server, while leaving the templating and UI sections to a much more mature and seasoned technology: the HTML, CSS, and JavaScript triad.

In this particular project, the server is built on the shoulders of the Java bindings to the Play[3] framework (see footnote for more information). The persistence layer is built on Java's Ebean[4] targeting a PostgreSQL[5] backend. For deployment, I used Heroku[6] and integrated it with Github[7] for automated deployments.

I've also decided against displaying which questions were answered incorrectly. The reason for that is that it would serve as *cheating*. One would just have to retake the entire Quiz and know which ones they have to actually change. Hiding which answers were incorrectly answered makes, in my opinion, for a more interesting game overall.

Likewise I've also decided against prompting the user when skipping a question. I think that prompting the user **every time** they want to skip ahead involves too many clicks.

In the following chapter I will elaborate on the core pieces of the application.

[1] https://www.wikiwand.com/en/Swing_(Java)
[2] https://www.wikiwand.com/en/JavaFX
[3] https://playframework.com
[4] http://ebean-orm.github.io/
[5] https://www.postgresql.org
[6] https://heroku.com
[7] https://devcenter.heroku.com/articles/github-integration