Unbabel Java Challenge

Tomasz Szypuła

June 5, 2019

1 Project Overview and Objectives

The goal of the challenge was to write a Java Application meeting the requirements specified in the java-coding-challenge. The main resources I have chosen to use in this task are

- JavaFX GUI framework which provides tools for building graphical user interfaces. Unfortunately as of Java 8 JavaFX is no longer prepackaged in the Java standard edition SDK and needs to be imported as a separate dependecy(library).
- The JUnit5 testing framework
- and of course the Unbabel API

The git repository with the project can be found here

2 The Application

The application is written in a MVC (model-view-controller) design pattern. Which in combination with `JavaFX` allows for easy and scalable data flow between the user and the back-end data model.

2.1 How to run the Application

As mentioned before the application is made using JavaFX. The application can be deployed as a

- standalone application
- standalone self-contained application package
- web application
- embedded in a web page

For the purpose of this challenge I have deployed my application as a runnable `.jar` file. In order to execute the `.jar` file(assuming you are in the directory with the `.jar` file) one can use the following command:

```
  java --module-path $PATH_TO_JAVAFX/javafx-sdk-11.0.2/lib
--add-modules=javafx.controls,javafx.fxml
-jar java-coding-challenge.jar $USERNAME $API_KEY
```

Where

- `$PATH_TO_JAVAFX` is the directory containing the JavaFX SDK which can be downloaded for example here
- `$USERNAME` username to the Unbabel Sandbox API
- `$API_KEY` api key to the Unbabel Sandbox API

The runnable `JAR` file can be found here

2.2 View

An example of the application as seen by the user is shown in fig.1
The user interface consists of

- A `TextField` where the user inputs the text to be translated.
- A `ChoiceBox` with the available languages.
- Two `Buttons` to submit the translation request and refresh the status of the request
- A `TableView` to store all the user queries.
- A `MenuBar` for future options such as *save,file,help,etc.*

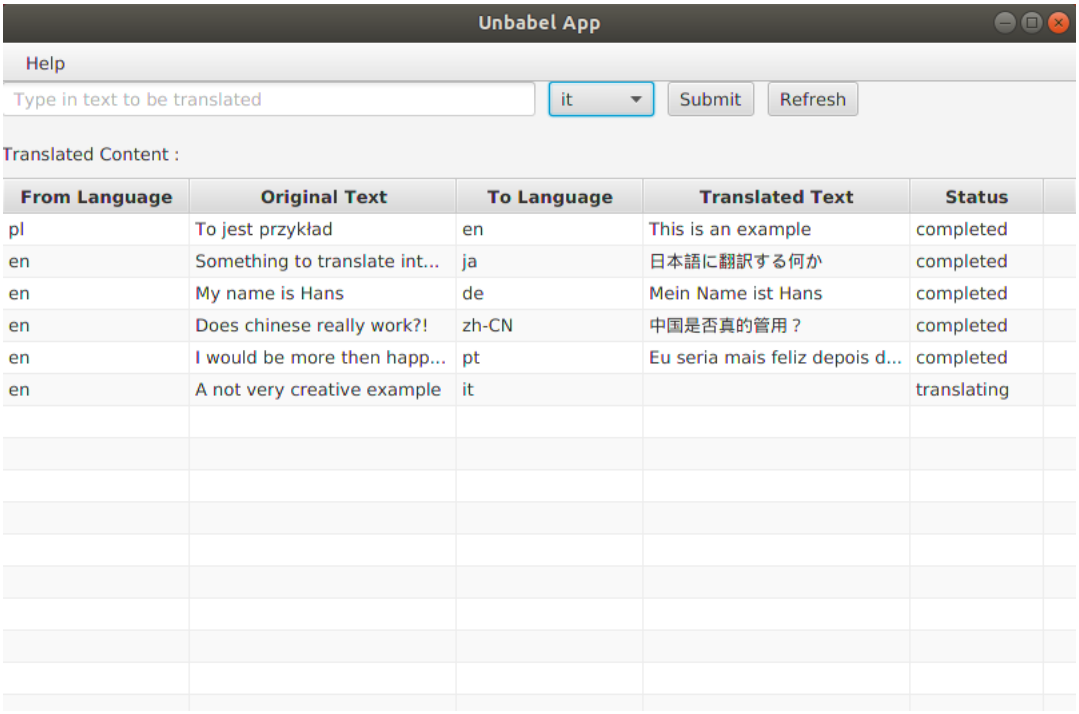

Figure 1: An example of the application user interface.

2.3 Model

The model takes care of all the user translation requests through the `HttpURLConnectionHandler` class. The class sends `POST` and `GET` requests using a `HttpsURLConnection` object from the `javax.net.ssl.HttpsURLConnection` package. When the user clicks on the *Submit* `Button` a `GET` request is sent to the Unbabel API. All data sent and received to and from the Unbabel API is stored and transferred between objects through the `Translation` class. Then when using the *Refresh* `Button` the `Model` according to the *Status* of the query either sends a `GET` request to check whether the translation is completed or resends the request (`POST`) in case of errors or failures. This simple approach proves to be surprisingly error resistant which is necessary when working with web applications.

2.4 Tests

For the sole purpose of this application I have decided to write my own `JsonParser` class to parse all `JSON` objects.In the future this should be replaced with a more reliable tool such as `Gson`. But for my simple needs my implementation was more then enough. The `JsonParser` class was also a great opportunity to write *Unit tests*. For this purpose I have used `JUnit5`. My tests can be found in the `JsonParserTest` class, which tests all the use cases in my application.

3 Scalability and Future Improvements

- The `Translation` class allows for easy adding of new features and translation parameters.
- Possible target languges used in the `ChoiceBox` should be replaced for example with an `Enum` which would provide greater flexibilty in adding new languages and getting the code languages in different forms (e.g. for user and model purposes).
- The `JsonParser` class should be replaced with a more reliable tool such as `Gson`.
- More test cases should be considered. Not only for `JSON` parsing but also fro `HTTP` connections and other features.