Coding Assignment

Betbull

September 2020

Abstract. In this assignment, you are expected to implement a single page iOS application whose content is retrieved from a URL over HTTP and is periodically updated in real-time using a web socket connection provided to you beforehand.

1 Introduction

In the starter project, you are given **two** applications:

- **Betbull**.xcworkspace
- **WebSocketServer**.xcodeproj

where **Betbull** is the application you must implement and **WebSocketServer** is just a read-only program which broadcasts live content which you'll receive and update the UI in Betbull app accordingly.

You do not have to touch any code in the WebSocketServer project.

1.1 Betbull.xcworkspace

Betbull has a feature of displaying sports tournaments called **Sportsbook** where bettors can view games (in other words *events*) and odds (in other words *outcomes*) between two opponents of the game.

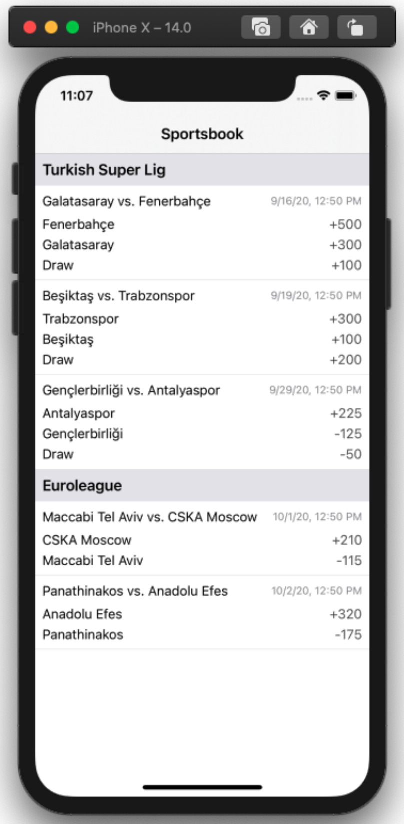
In order to understand Sportsbook, you should understand the following models that represent it:

- **Tournament**: Premier League
- **Event**: Manchester United vs. Arsenal
- **Market**: Match Result
- **Outcome**: Manchester United wins

You'll visualize the relation between them better in your mind when you look at the JSON content in the following URL:

`https://run.mocky.io/v3/38bc099e-1170-45ce-ab53-200d10e1522b`

At the end of project, the **home** screen of sports book feature should look like as follows using the URL mentioned above:



Sportsbook

Turkish Super Lig

Galatasaray vs. Fenerbahçe	9/16/20, 12:50 PM
Fenerbahçe	+500
Galatasaray	+300
Draw	+100
Beşiktaş vs. Trabzonspor	9/19/20, 12:50 PM
Trabzonspor	+300
Beşiktaş	+100
Draw	+200
Gençlerbirliği vs. Antalyaspor	9/29/20, 12:50 PM
Antalyaspor	+225
Gençlerbirliği	-125
Draw	-50

Euroleague

Maccabi Tel Aviv vs. CSKA Moscow	10/1/20, 12:50 PM
CSKA Moscow	+210
Maccabi Tel Aviv	-115
Panathinakos vs. Anadolu Efes	10/2/20, 12:50 PM
Anadolu Efes	+320
Panathinakos	-175

Figure 1: Sportsbook home screen

1.1.1 Design Pattern

Each feature *(ex: Sportsbook)* in the application is formed of **3** components which are used for different purposes:

- **Coordinator**: Handles navigation from one scene to another *(push, pop, present etc.)*.
- **Scene**: UI with **no** business logic involved. It calls the coordinator when a user action *(didSelectRowAt, tapping back button etc.)* is resulted in navigating another scene.
- **Interactor**: Handles business logic *(making HTTP requests, receiving data from web socket, generating data source for scene etc.)*

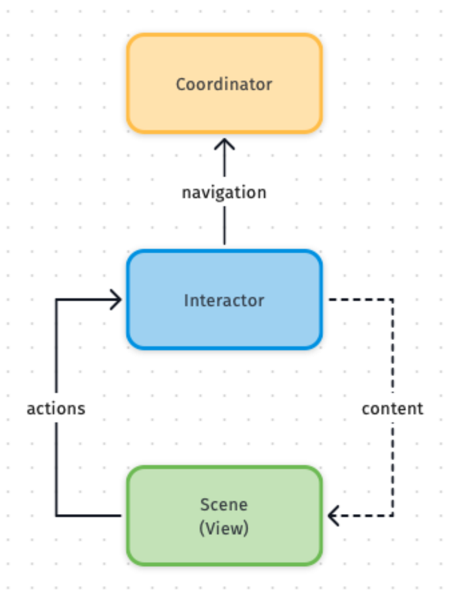


Figure 2: Relations between components of a feature

1.2 WebSocketServer.xcodeproj

WebSocketServer is basically responsible for broadcasting updates on events and odds in real-time.

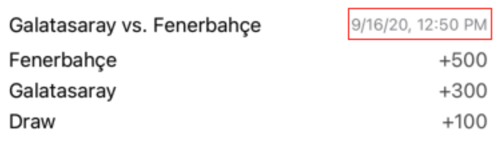
When you build and run WebSocketServer.xcodeproj, you'll see the web server starts running at URL **http://0.0.0.0:8080** and sending clients some rudimentary payloads in JSON format like below:

```
Started websocket server at 0.0.0.0:8080

Client connection received: /websocket
Sent: {"event": "2365260", "startTime": 1600260600}
Sent: {"outcome": "9818322", "price": "+300"}
```

Those payloads describe which part of an event or outcome will be updated. For instance;

```
{"event": "2365260", "startTime": 1600260600}
```

indicates the *starting time* of an event whose *id* is 2365260 has been updated so you should redraw the content of corresponding cell in the Betbull app:

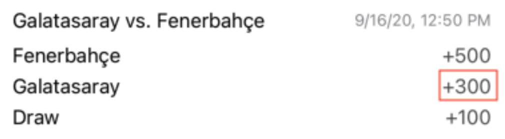


Galatasaray vs. Fenerbahçe	9/16/20, 12:50 PM
Fenerbahçe	+500
Galatasaray	+300
Draw	+100

Similarly;

```
{"outcome": "9818322", "price": "+300"}
```

indicates the *price* of an outcome whose *id* is 9818322 has been updated so you should redraw the content of corresponding cell in the Betbull app:



Galatasaray vs. Fenerbahçe	9/16/20, 12:50 PM
Fenerbahçe	+500
Galatasaray	+300
Draw	+100

1.2.1 Receiving real-time updates (Testing)

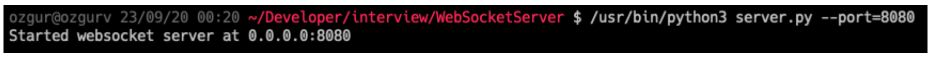
In order to receive such payloads from **WebSocketServer.xcodeproj** for testing purposes, you should **first** run it, then add the following code snippet into AppDelegate.swift in **Betbull.xcworkspace**:

```
public class AppDelegate: ... {

  private var token: WebSocketClient.Token!

  public func application(... didFinishLaunchingWithOptions ... {
    WebSocketClient.shared.connect()

    token = WebSocketClient.shared.register { payload in
      print("Received: \(payload)")
    }
    return true
  }
}
```

1.2.2 Troubleshooting

If you managed to build and run **WebSocketServer**.xcodeproj, you can skip this section. Otherwise, please read on.

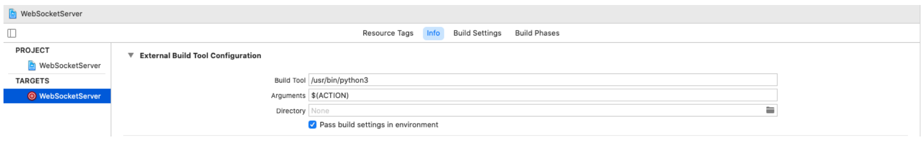
You should make sure that you have **python3** installed on your system because WebSocketServer.xcodeproj does nothing but executes the following command on your terminal:



```
ozgur@ozgurv 23/09/20 00:20 ~/Developer/interview/WebSocketServer $ /usr/bin/python3 server.py --port=8080
Started websocket server at 0.0.0.0:8080
```

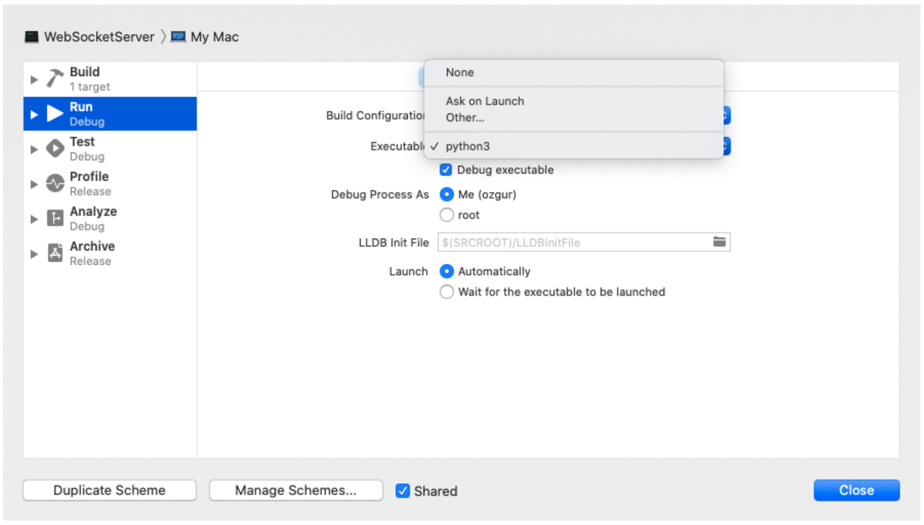
Figure 3: Running web socket server

It is very unlikely however if **python3** is located somewhere else than /**usr**/**bin**, then you have to update the path by editing scheme and target configurations:



Build Tool /usr/bin/python3
Arguments $(ACTION)
Directory None
✓ Pass build settings in environment

Figure 4: Updating build tool of WebSocketServer target



Build Configuratio
Executabl ✓ python3
✓ Debug executable
Debug Process As [✓ Me (ozgur)] root
LLDB Init File $(SRCROOT)/LLDBInitFile
Launch [✓ Automatically] Wait for the executable to be launched
✓ Shared

Figure 5: Editing scheme and selecting Other... executable

2 Conclusion

Once you manage to fetch and parse the JSON in the URL and then update the content in real-time using the web socket connection, you are expected to have the following table in the sports book home screen like below:

View the final application

Good luck.