Fight Night Bot Sample

### Overview

This article shows how to create an Adaptive Card bot for displaying fight card information for combat sports events.

### Prerequisites

git

Python 3.6+

Microsoft Bot Builder SDK - Python version

Bot Framework Emulator

Knowledge of Python data structures and JSON files

### Situtation

You want to create an aesthetically pleasing way to present information about the various bouts on a fight card of a combat sports event. You have access to statistics and information about the fighters and their respective bouts. The user of the bot can either type in the full bout name, the number of the bout, a fighter's name, or another keyword to show the relevant Adaptive Card for the bout.

### Getting Started

Follow Microsoft's instructions for building and running the Bot Builder SDK:: <https://github.com/Microsoft/botbuilder-python/wiki/building-the-sdk>

We will be using the Bot Framwork Emulator later to test the bot. Go ahead and install the most stable build now: <https://github.com/Microsoft/BotFramework-Emulator/releases>

Now go to <https://github.com/emgrol/fight-night-bot> and download requirements.txt, main-skeleton.py, and fight-night.bot (go ahead and download the whole project if you need a working example). Install the required packages in requirements.txt. Once we copy the three aforementioned files to a new folder you are ready to begin.

Our bot will be designed as a simple Adaptive card bot. The bot prompts the user with a menu of options and displays the Adaptive card content if the user inputs a valid string or the menu again if there is no response. Otherwise the bot displays an error message and re-prompts the user for input.

In this article we will be using an Adaptive Card made with the Adaptive Card Designer (<https://adaptivecards.io/designer>) specifically formatted for fight card bout information. For the remainder of this article I will be referencing this Adaptive Card, but feel free to add to this card if you want to display more/different information or create your own. If you want to use my Adaptive Card, download it from <https://github.com/emgrol/fight-night-bot> and add it to the folder you created, or if you want to create an Adaptive Card add that JSON file to the folder.

Next we need fighter and bout information for the fight card in a JSON file. In <https://github.com/emgrol/fight-night-bot> we’ll find a file called “ufc\_237\_fighters.json”, which contains information about the UFC 237 main card bouts. Again feel free to download this JSON file and add it to your bot folder, or create your own JSON file with fight card information.

Now that we have all of the JSON files we need to create the cards, we can begin working on the code in main-skeleton.py. This program contains the key functions for the asynchronous communication between the bot and the user, and contains “TODOs” where we will fill in the code that’s relevant to the Adaptive Card and fight card information JSON files.

Staring at the beginning, we add the names of our Adaptive Card and fight card information JSON files. In my program they are “fightercard.json” and “ufc\_237\_fighters.json”

**def** read\_in\_jsons():

**with** open (**"fightercard.json"**, **"rb"**) **as** file\_in:

fight\_card\_json = json.load(file\_in)

**with** open(**"ufc\_237\_fighters.json"**, **"rb"**) **as** file\_in:

fighter\_information = json.load(file\_in)

**return** fight\_card\_json, fighter\_information

We need to determine how to to assign the values of the fight card information JSON file to their appropriate place in the Adaptive Card. Essentially, we need to figure out how to “fill out” the Adaptive Card JSON file with the values from each bout from the fight card information JSON file. For example, here is how I assigned the necessary values for the red corner fighter, like nickname, full name, image url, record, country, height, and reach:

fight[**"body"**][0][**"items"**][0][**"columns"**][0][**'items'**][0][**'text'**] = v[**"red\_corner"**][**"nickname"**]

fight[**"body"**][0][**"items"**][0][**"columns"**][0][**'items'**][1][**'text'**] =v[**"red\_corner"**][**"fullname"**]

fight[**"body"**][0][**"items"**][0][**"columns"**][0][**'items'**][2][**'url'**] = v[**"red\_corner"**][**"img\_url"**]

fight[**"body"**][0][**"items"**][0][**"columns"**][0][**'items'**][3][**'facts'**][0][**'value'**] = v[**"red\_corner"**][**"record"**]

fight[**"body"**][0][**"items"**][0][**"columns"**][0][**'items'**][3][**'facts'**][1][**'value'**] = v[**"red\_corner"**][**"country"**]

fight[**"body"**][0][**"items"**][0][**"columns"**][0][**'items'**][3][**'facts'**][ ][**'value'**] = v[**"red\_corner"**][**"height"**]

fight[**"body"**][0][**"items"**][0][**"columns"**][0][**'items'**][3][**'facts'**][3][**'value'**] = v[**"red\_corner"**][**"reach"**]

We also need to create a list of keywords that will be used in the card\_response function. These keywords should be words or phrases that users would realistically use to indicate that they want information about a specific fight, like “<Red Corner name> vs. <Blue Corner name>” or “<Red/Blue Corner name>”, as seen below in my project.

keywords = [**"%s"** % (v[**"red\_corner"**][**"fullname"**]),

**"%s"** % (v[**"blue\_corner"**][**"fullname"**]),

**"%s vs. %s"** % (v[**"red\_corner"**][**"fullname"**],

v[**"blue\_corner"**][**"fullname"**])]

Now that we have a dictionary filled with Adaptive Cards we can create a menu string that will be displayed in a card as a list of fights. Each string should consist of a number (to reference the bout by number) as well as a string the user would recognize for the fight, most probable “<red corner fighter> vs. <blue corner fighter>”. For my project the string is formatted as so:

s = **"(%d) %s\n"** % (i[2], i[1][2])

Lastly we need to finish the code in card\_response function so we can create a dictionary of the possible valid inputs from the user and the associated card to that input. In my project the logic is as follows, where both the number associated with the fight and all keywords are keys and the dictionary that displays the Adaptive Card is the value:

**for** i in fight\_night:

choice\_dict[str(i[2])] = i[0]

**for** j in i[1]:

choice\_dict[j] = i[0]

After filling out all of the “TODOs” in main-skeleton.py we should be ready to run the bot. Feel free to change the name of main-skeleton.py to main.py or another name now. To run the bot open a command prompt and cd into the directory with your bot. Type in the following command:

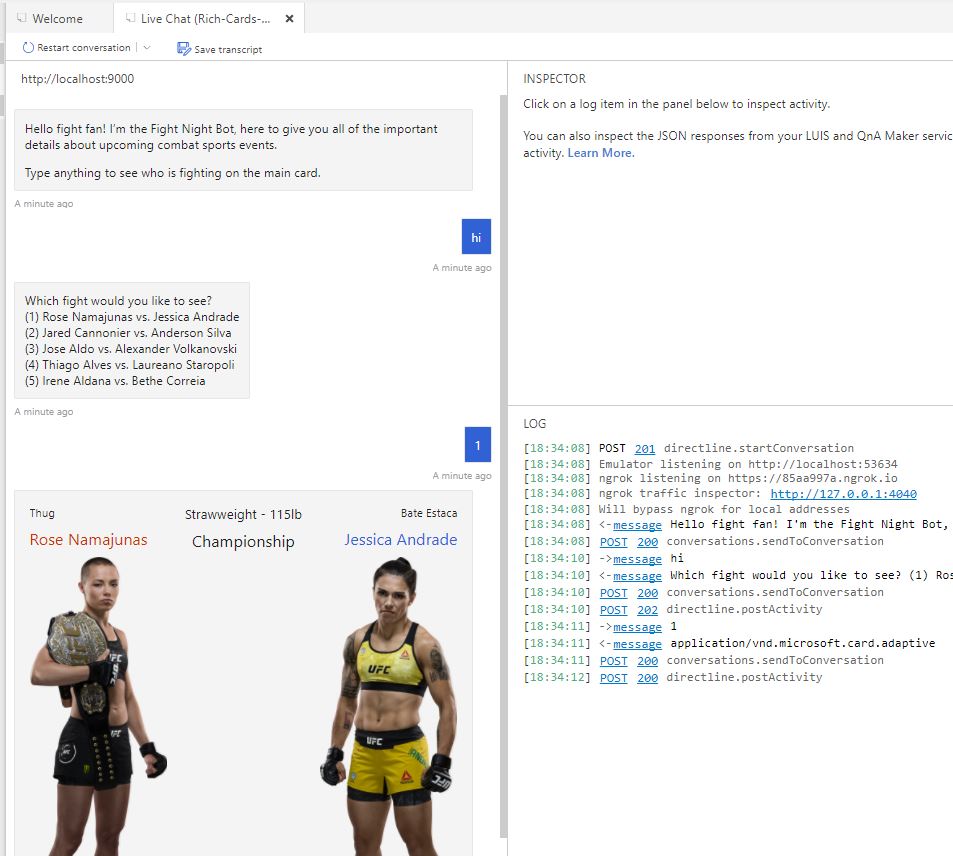
python <name of main Python file, usually main.py>

When we run the command we should see the following:

======== Running on http://localhost:9000 ========

(Press CTRL+C to quit)

Open up the Bot Framework Emulator. You can either search for the .bot file in your bot’s folder, or you can type “<http://localhost:9000>” into the address bar. You should see a screen like this:



Go ahead and try a variety of inputs to see which Adaptive Cards are displayed. If you are getting 400 errors or 500 errors make sure that you have the required packages installed.

### Conclusion

This tutorial detailed the creation of a fight card information bot and use of the Bot Framework Emulator.