Christine Zhou

March 3, 2022

ISOM 356: Think.Code.Make

Midterm – The Guardian API

**1.  Document your API selection process. Why did you choose this API? What does the**

**API do? What did you hope to accomplish?**

As a college student, I discovered that there are two constants: limited time, and curiosity. Time is always of the essence, and with so many options to take, curiosity is fleeting. That is why I wanted to choose an API that would provide useful information quickly and easily.

When looking through API, the number of choices that could satisfy my requirements were overwhelming. Initially, I chose the Spotify API, which turned out to be more complicated than I imagined as a first-time API user.

A screenshot of a computer

Description automatically generated with medium confidence

Below, I provide screen captures of the different options for selecting the correct OAuth.

Table

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidence

The following diagram describes how the OAuth for “Client credentials” would work.

Diagram, timeline

Description automatically generated

After encountering these obstacles, I looked to a different category, and settled on The Guardian API.

Graphical user interface, text

Description automatically generated

It was a much smoother process. I entered my name, email, reason for use, and was sent an email almost immediately.

Graphical user interface, text, application

Description automatically generated

Finally, I could work with the data. In The Guardian API, I had a choice to look for content, tags, sections, editions, or single item. I chose content, because I felt that gave the most relevant information for someone looking for quick news as a starting point. The rest of the information was more additional; complementary (tags and sections), or longer (editions), and would more likely be information a user on The Guardian website would search for. The purpose of my API is more intended for additions to another site, or a quick bypass for news without needing to load all The Guardian site’s UI.

Graphical user interface, text

Description automatically generated with medium confidence

This API has the ability to search for the latest ten articles posted on The Guardian site, and allows one to keep up to date with the latest news. While the API itself does not give article details, the user can view information such as the article title and url, so they can seek more details if they are interested. I hope that this will allow someone to keep up to date with news if this code is added to another app or website as an additional information source.

**2.  Tell us how it went. What did you struggle with / how did it go?**

Using the API itself provided its own challenges. The output from the first 20 lines of printing the whole API was difficult to read because it was so condensed and did not print like the examples we learned in class. As a result, I had to copy and paste the text and do some simple searches of the text (Ctrl + F) for indicators such as the curly brackets ({}), in order to read the code more easily. The lack of indentations was an issue!

Text

Description automatically generated

After that, I was able to chose how to write my program. Based on the given information, I decided to give the user a choice to either see the details of one single article, or view “aggregate” data, for example the urls of all ten articles. Next, the user can choose to go back and make their selections again, if they want different information.

The next issue I encountered was writing the code to a file. I investigated printing the desired output in one long string as done in the transcription example in class. I settled on using a list because I had to assign the output to values one at a time after each function, so I could print these elements on one line each, which would then print on a file.

Text

Description automatically generated

The code above gives a glimpse into how to use this API. Despite the difficult-to-read file, I had a lot of data and code to work with. As mentioned, I was able to choose from a variety of articles and details of these articles, including title, date uploaded, and url. I chose from a series of five, type, section, date, title, or url, that the user could choose from to view details about. I also decided they could look at all their options in aggregate data. For example, they could see the ten article titles, or all 10 urls. Finally, they can choose to go back to the selection for more data on a specific article, if they want to see whether it is an article (factual) or editorial (opinion), or search for the url to search online for the full article.

**3.  Show us output from your program confirming the API usage (screenshots or**

**video).**

See the output in the terminal for how the program works, including giving (and selecting) different options for specific versus aggregate data.

**Viewing Specific Data**

Text

Description automatically generated

Choosing the number of the article you would like details on.

Text

Description automatically generated

Shown output of chosen article, followed by the option to return to original selection.

Text

Description automatically generated

**Output From the .txt File (Specific Data)**

The first line of code shows “specific” because that was the user’s first selection input. Next, the chosen article details are printed underneath.

Text

Description automatically generated

**Viewing Aggregate Data**

Text

Description automatically generated

Selection type of details. As shown, the user must select a number between 1-5, otherwise the question will be prompted again.

Text

Description automatically generated

The details are printed.

Text

Description automatically generated

**4.  What would you like to do next after this experience? How will you apply what you learned and/or what can you improve upon?**

I would like to build an API that pulls more articles or information from The Guardian, or other new sites, for a wider range, and more comprehensive information. This API is limited because it only searches for the most recent news articles. Many articles are posted very frequently, possibly even hundreds per day, but there is no way to search The Guardian API for older articles. This would be a better, complementary back-end code for a site that compiles news from many news sites, that also pulls information from The Guardian.

Another route to experiment is improving on my current application. This could mean making the output of the article details more readable. Currently, my output is still in the default, Python dictionary format. It would be a larger project to coerce a cleaner, more reader-friendly output by possibly removing the keys in the dictionary, and only printing the value. Currently, all inputs are printed if the user chooses to view more data. I could also allow the user to delete parts of the list that is printed, in case they only want to keep part of their search in the output.

For further projects, I can use this API knowledge to pull data from other sources, such as weather, stock prices, and more. As the current code would combine with more current events and information from the weather and so on, this could be a quick way for someone to become knowledgeable about current events, by a simple search input in their phone on their commute to work or school. As a college student, I would appreciate this all-in-one functionality. Additionally, this code would be good addition as the back-end for any app with a fancier UI, which could combine well to create an app.

Works Cited

*The Guardian*, Guardian News and Media, https://open-platform.theguardian.com/.

Public-Apis. “Public-Apis/Public-Apis: A Collective List of Free Apis.” *GitHub*, https://github.com/public-apis/public-apis.

“Web Api.” *Spotify for Developers*, https://developer.spotify.com/documentation/web-api/.