TP 2 - Client–Server communication

Adrien Humilière 25/03/2020

Part I
HTTP Requests

The objective of this lab is to use `URLSession` to pull data down from the iTunes API and display it in the console. You'll use a search query dictionary to configure a URL, which the URL session will use to fetch and print the correct data.

With the current situation, we will use repl.it to code this lab. Create an account on the platform to easily save your work, as we will probably keep using it in the following labs. Please use the **Share** button to send me the final project's URL. Also, link it if you have any question for me.

Start by adding the following lines of code to `main.swift` :

```
1 import Foundation
2 RunLoop.main.run()
```

Step 1: Review the iTunes Search API

- Take a few minutes to review the documentation for the **iTunes Search API**. Find the base URL for search requests, and pay particular attention to the types of queries you can make.
- Look at the different parameter keys listed in the documentation and their corresponding values. The different keys (term, media, limit, etc.) will be used when you create your query dictionary. Think of some parameters that you might want to include in your query.
- Create a query `[String: String]` dictionary that looks for your favorite movie or for songs by your favorite music artist. Make sure to use the exact keys and expected values listed in the API documentation. At a minimum, you'll want to use the `term` and `media` keys.
- Now that you have your query dictionary, create a variable to hold the base URL for the iTunes Search API: `https://itunes.apple.com/search?`.
- To make it easier to configure your URL properly, add an extension to the `URL` type with a function that returns a `URL?` based on a `[String: String]` query dictionary. You can use the same extension displayed in the slides of this lesson.

```
extension URL {
    func withQueries(_ queries: [String: String]) -> URL? {
        var components = URLComponents(url: self,
        resolvingAgainstBaseURL: true)
        components?.queryItems = queries.map
        { URLQueryItem(name: $0.0, value: $0.1) }
        return components?.url
    }
}
```

- With your base URL and query properties, create the search URL that you'll use to request data from the API.

Step 2: Pulling Data from the Web

- Now that you have your URL configured correctly, you'll use it to fetch your data from the web.
- Use the shared URLSession to create a dataTask for the specified URL. The completionHandler will give you three properties to work with: Data?, URLResponse?, and Error?. Provide appropriate names for the placeholder values.
- Don't forget to resume your dataTask.
- You're now ready to check if the dataTask has completed with valid data. Unwrap the data you received. If the data exists, create a string from the data that will display the data's contents, then print that string to the console.