⑂ **audreywelch** / **ios-itunes-search**
   forked from LambdaSchool/ios-itunes-search

No description, website, or topics provided.   [Edit]

Manage topics

⑪ **1** commit	⑂ **1** branch	◌ **0** releases	👥 **1** contributor

| Branch: master ▾ | New pull request | | Create new file | Upload files | Find file | Clone or download ▾ |

This branch is even with LambdaSchool:master.   ⑂ Pull request   ⎘ Compare

👤 **SpencerCurtis** Create README.md with project instructions   Latest commit 33b918f on Aug 7

| 📄 README.md | Create README.md with project instructions | 4 months ago |

📖 **README.md**   ✏️

iTunes Search

A student that completes this project shows that they can:

- understand and explain common use cases for enums
- define custom enum types
- use documentation to understand how to use a REST API
- understand and explain the purpose of JSON
- use URLSession to make a GET request to a URL
- implement a URLSessionDataTask completion closure
- use JSONDecoder() to convert JSON data returned by an API into model objects

Introduction

iTunes Search allows you to search for apps, music and movies using the iTunes Search API. You will practice using enums, GET requests using `URLSessionDataTask` , using API documentation and parsing JSON.

Please look at the screen recording below to know what the finished project should look like:

Buena Vista Home Entertainment, Inc.
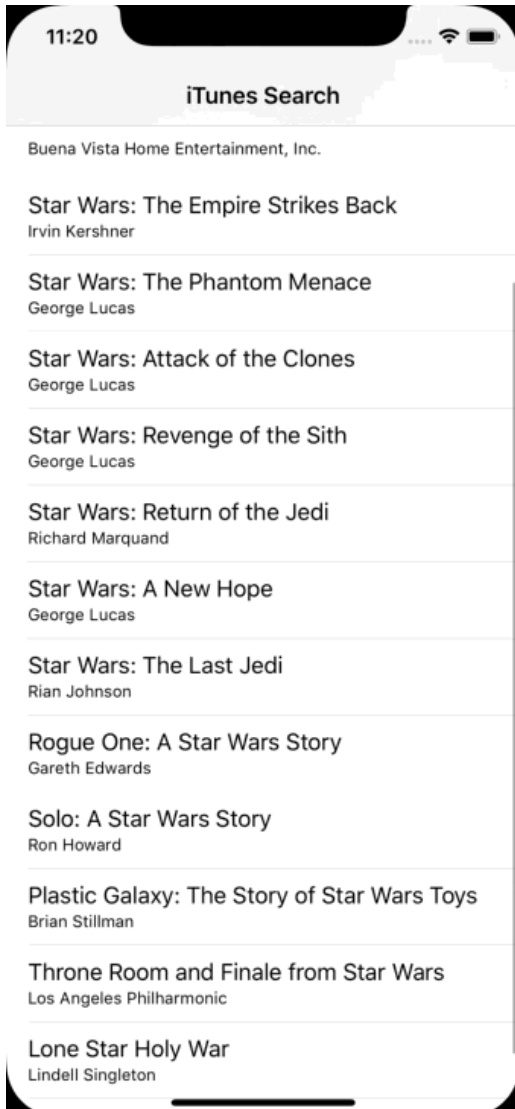
Instructions

Please fork and clone this repository. This repository does not have a starter project, so create one inside of the cloned repository folder.

🔗 Part 1 - SearchResult and SearchResultController

SearchResult

This application uses the iTunes Search API. Please familiarize yourself with the API's documentation here. Read the overview, and look at the "Searching" section. There are some search examples that are worth looking at as well.

We're going to search for either an app, music, or movie. Use this URL to see some example JSON. **NOTE:** the URL will download a file to your desktop which contains the JSON. You will need to open the file in a text editor to see the JSON. It's recommended that you then copy and paste the JSON in a JSON formatter.

1. Create a new Swift file "SearchResult.swift", and create a struct called `SearchResult`. Adopt the `Codable` protocol.
2. Add the following properties:
   - a `title` string

- an `creator` string.

Note: The JSON returned from the iTunes Search API refers to the developers of an application as the `artistName`, and the title of the search result as the `trackName`. The same goes for the `trackName` for the title of the search result. No matter whether it's music, movies, apps, etc. the JSON uses these same key-value pairs to keep the JSON consistent.

You may have noticed that the `title` and `creator` properties in this struct don't match any keys in the JSON. As a recap, a class or struct that adopts `Codable` will by default use the names of its properties as the keys it should look for in the JSON that it is trying to decode. However there are instances where the keys in the JSON perhaps aren't what you'd like to call your properties throughout your app. We can implement another part of `Codable` that lets us override this behavior called `CodingKeys`. `CodingKeys` allow us to and map the keys from the JSON to the properties we want their values to be stored in. Still in the `SearchResult` struct, add the following:

```
enum CodingKeys: String, CodingKey {

}
```

3. This enum has `String` raw values, and it conforms to the `CodingKey` protocol. Now, add a `case` for `title` in the enum. Give it a raw value of `"trackName"`. When the decoder is going through the JSON, it will now look for a key called `trackName`, but instead of trying to put that value in a property called `trackName` (which in this struct doesn't exist, thus making the decoding fail), it knows to put the value in a property called `title` because that is the name of the raw value's case. Make a case for `artist`. Its raw value should be `"artistName"`.

Take a minute to look at the example json at the URL at the start of part 1. The JSON objects that represent each `SearchResult` are nested in the JSON. If we were to try to use a `JSONDecoder()` and try to decode it into `[SearchResult]` it wouldn't work. We need to create an object that represents the JSON at the highest level (the object with `resultCount` and `results` keys).

4. Under the `SearchResult` struct, add a `SearchResults` struct. Create a `results: [SearchResult]` constant. This will allow us to decode the JSON data into this object, then access the actual search results through its `results` property.

ResultType

1. Create a new Swift file called "ResultType.swift".
2. Create an enum called `ResultType`. Set its raw values to be `String`. This will represent the kind of item you want to search for in the API.
3. Add the following cases:
   - `software` - (for apps)
   - `musicTrack`
   - `movie`

SearchResultController

1. Create a new Swift file called "SearchResultController.swift", and make a class called `SearchResultController`.
2. Add a `baseURL` constant. This should be the base URL for the iTunes Search API.
3. Add a `searchResults: [SearchResult] = []` variable. This will be the data source for the table view.
4. Create a `performSearch` function with a `searchTerm: String`, a `resultType: ResultType` parameter, and a `completion` closure. The completion closure should return an `NSError?`. As a first measure of help for closure syntax, look at the "As a parameter to another function" section of this page. You're obviously free to ask a PM for help as well.

5. Create your full request url by taking the `baseURL` , and adding the necessary query parameters (in the form of `URLQueryItem` s.) to it using `URLComponents` .

6. This function should use `URLSession` 's `dataTask(with: URL, completion: ...)` method to create a data task. Remember to call `.resume()` .

7. In the completion closure of the data task:
   - Give names to the return types.
   - Check for errors. If there is an error, call completion with the error.
   - Unwrap the data. If there is no data, call completion, and return `NSError()` in it.
   - If you do get data back, use a do-try-catch block and `JSONDecoder` to decode `SearchResults` from the data returned from the data task. Create a constant for this decoded `SearchResults` object.
   - Set the value of the `searchResults` variable in this model controller to the `SearchResults` ' `results` array.
   - Still in the `do` statement, call completion with `nil` .
   - In the `catch` statement, call completion with `nil` for the array of recipes, and the error thrown in the catch block.

Part 2 - Storyboard Layout

1. In the Main.storyboard delete the pre-added `UIViewController` scene.
2. Add a `UITableViewController` scene. Embed it in a navigation controller, and set it as the initial view controller.
3. Set the navigation item's title to "iTunes Search"
4. Change the cell's style to "Subtitle"
5. Add a `UIView` as the table view's header view. **HINT:** To do this, you drag the `UIView` right above the table view, and it should expand to the width of the table view, then you can drop it there. Select the `UIView` and expand it to about 100 points.
6. Add a `UISegmentedControl` to the header view. This segmented control will let the user control which type of item they search for in the API. Select the segmented control, and in the Attributes Inspector:
   - Change the number of segments to three.
   - Set the first segment's title to "Apps", the second's to "Music" and the third's to "Movies".
7. Add a `UISearchBar` to the header view under the segmented control.
8. Constrain the segmented control and the search bar.
9. Create a Cocoa Touch Subclass of `UITableViewController` . Call it `SearchResultsTableViewController` . Set the table view controller scene's class to this new subclass.
10. Create an outlet from the segmented control and the search bar.

Part 3 - View Controller Implementation

In the `SearchResultsTableViewController`

1. Create a constant called `searchResultsController` whose value is a new instance of `SearchResultController` .
2. Using the instance of `SearchResultController` , fill out the `numberOfRowsInSection` and `cellForRowAt` methods. Each cell should display the `title` and `artist` of a `SearchResultObject` .

When using a `UISearchBar` , you use a method in the `UISearchBarDelegate` to trigger searches when the user taps the search button on their keyboard. This method is called `searchBarSearchButtonClicked` .

3. In order to use the `searchBarSearchButtonClicked` method, adopt the `UISearchBarDelegate` protocol in your table view controller.
4. In the `viewDidLoad` , set the search bar's `delegate` property to the table view controller.
5. Call the `searchBarSearchButtonClicked` method. This method should take the search term that the user enters in,

and trigger a search for the specific result type they specify using the segmented control. Look below for step-by-step instructions on how to do this if needed.

▶ Step-by-step implementation of searchBarSearchButtonClicked

Go Further

- Perform a search every time the user selects a new segment on the segmented control.
- Allow for limiting searches by country and/or by a number of desired search results.
- Refer back to the JSON and make the `searchResult` object have additional properties. Create a custom cell and/or a detail view controller to display these extra properties to the user.