INFO / CS 2300 HW3: Spot

Due - Tuesday April, 11th 2017 at 5:00pm on the 2300 server

Introduction

Homework 3 is designed to allow you to gain experience in working with web APIs (Application Programming Interface). So far, you have always had your data stored locally or in your own database, whether that be in an external text file, or stored in your database on the course server. The Spotify Web API, like all APIs, enables you to fetch data from an external source, for use in your own applications. Your task will be to fill in the skeleton code provided to create a mini Spotify application. The three major pieces you will have to complete to create the player are: Oauth (for login), populating the songs table + audio preview, and search across the API.

This assignment is optional. You can get up to 3 points added onto your final grade if you get a 100 on this assignment, 2 if you get between a 66 - 99, 1 between a 33 - 65. In other words, there is no way to lower your overall grade by attempting this assignment. Anything you succeed at here will improve your overall grade.

Making data from user accounts available through an API is susceptible to abuse so the steps to properly authenticate are significant.

Files

Instructions

Setup

You must create a free Spotify account to receive an API key. If you do not have an account, you can create a free account here. https://support.spotify.com/us/account_payment_help/account_basics/create-your-spotify-account/ Be sure to use the service for a little bit so that you can generate songs to populate your music library. In this assignment we will be looking at your saved tracks, so be sure to save some.

Besides a personal account, you must register a new application (your web program) with Spotify (https://developer.spotify.com/my-applications) to receive a CLIENT ID that will enable access to the Web API (don't worry about the CLIENT SECRET) this is also free. You will need to configure some settings with your application first before writing any code. We strongly advise that you work locally before trying to put it on the course server.

Get the starter files for this assignment (will be posted to piazza) and unzip them into a folder on your web server.
Inside of spotify-config.js, there are some variables that are left blank. Copy paste your unique clientID into this file, then open the site locally on a web server and look at the url, it might look something like: http://localhost/2300/INFO2300_hw3/

Yours might be different. Take this url and go back to the spotify application control panel (https://developer.spotify.com/my-applications) and paste this URL under the Redirect URIs section and append callback.html to the end of that base url you copied. For example, http://localhost/2300/INFO2300_hw3/callback.html

Make sure to save those changes by pressing save at the bottom of the control panel, it does not auto save.

Go back into spotify-config.js and where it says baseRedirect, assign that to the the base url for the local web app, for example: http://localhost/2300/INFO2300_hw3/

Once you've done this step, you should be done with the spotify application configuration panel for now. Make sure to save any changes before leaving.

Part 1: Oauth

In the first part of the assignment, you need to successfully make a request to the spotify API and log the user data. The Spotify API makes use of a type of authentication called Oauth. This is a standard used for many purposes on the internet.

Refer to comments in code for more in depth directions but, in general, the first goal is to get the accessToken back. Then call the predefined function getUserData with the token.

In order to do this, first you need to know the correct access scope you would like to request from the user, **reading** their **email** is the only scope you need for this part.

Make sure to add the right scope variable to the scopes array in spotify-config.js. You'll need to reference the spotify api documentation.

https://developer.spotify.com/web-api/using-scopes/

Afterwards, the buildLoginUrl function needs to be completed using the clientID, redirectURI, and scopes. To complete this, follow the given example in the url just mentioned above.

Next, you need to open a new window with the correctly built URL. The person using your application will use this new window to enter their Spotify username and password. This is the first parameter of the window.open function towards the end of the login function.

When the user submits a username and password, your login function should be able to successfully get back an accessToken. With this token, you can now use the spotify API to get data about that user. Call the getUserData function passing in the access token you get from the callback function of the login function.

This is similar to the way you use mysqli. Once you have set up a mysqli object with the proper host, username, password and database, you can use it to access lots of information in the database.

getUserData is an ajax call, which means you can not simply call the function directly and store the result as a variable. Use a .then() function in conjunction with the getUserData function call to properly call and wait for the ajax call to render. Inside of your .then() you need a function that gets back a response object. In other words: .then(function(response) { … });

To finish part one, simply hide the login button (you can use jQuery) and console.log(response); If everything worked out correctly you should see an object in the console with some basic user data about you!

Part 2: 50 tracks + audio playback

First, make sure you have at least **50** songs in your saved tracks. Each record (song name, artist, album) of the table will make up one of these 50 songs. You will need to make an API request to retrieve these 50 songs, and populate the table with each of these songs. When testing your app, the TA will see the their own songs not yours.

You may notice that as you interrupt audio from the media player, it will throw an error. This is ok, and is the only exception to JS errors for this assignment. Feel free to console.log anything you need.

Refer to the inline comments in the code for specific instructions on this section.

Part 3: Search

Your search button must implement an AJAX call that searches through your saved songs and dynamically adjusts the table with each character typed. Spotify does not explicitly specify a rate limit, but if you encounter a 429 error, you are making too many API requests.

It's ok to only display 20 results for each API call. Refer to inline code comments for specific instructions.

You may notice that as you interrupt audio from the media player, it will throw an error. This is ok, and is the only exception to JS errors for this assignment. Feel free to console.log anything you need.

What to submit

You must test your work on the server before submission. In lecture we spoke how the spotify API needs to know the specific redirectURI based on where the application is running. You will need to modify this on the server vs working locally. Work locally first and when you're ready to submit, change your spotify-config.js file on the server.

Upload the entire project to a new folder, HW3. Or, if the assignment has been distributed to all students on the server before hand, be sure to keep it the same name.

You will also submit a small text file on CMS that tells TAs you're ready to be graded. Simply put "Ready" in a ready.txt file.