Proyecto 1: Libros

Objetivos

- Ponte más cómodo con Python.
- Gana experiencia con Flask.
- Aprenda a usar SQL para interactuar con bases de datos.

Visión general

En este proyecto, creará un sitio web de reseñas de libros. Los usuarios podrán registrarse en su sitio web y luego iniciar sesión con su nombre de usuario y contraseña. Una vez que inicien sesión, podrán buscar libros, dejar reseñas de libros individuales y ver las reseñas realizadas por otras personas. También utilizará una API de terceros de Goodreads, otro sitio web de reseñas de libros, para obtener calificaciones de una audiencia más amplia. Finalmente, los usuarios podrán consultar los detalles del libro y las reseñas de libros mediante programación a través de la API de su sitio web.

Empezando

PostgreSQL

Para este proyecto, deberá configurar una base de datos PostgreSQL para usar con nuestra aplicación. Es posible configurar PostgreSQL localmente en su propia computadora, pero para este proyecto, utilizaremos una base de datos alojada por Heroku , un servicio de alojamiento web en línea.

1. Vaya a https://www.heroku.com/ y cree una cuenta si aún no tiene una.
2. En el Panel de Heroku, haga clic en "Nuevo" y elija "Crear nueva aplicación".
3. Dé un nombre a su aplicación y haga clic en "Crear aplicación".
4. En la página "Descripción general" de su aplicación, haga clic en el botón "Configurar complementos".
5. En la sección "Complementos" de la página, escriba y seleccione "Heroku Postgres".
6. Elija el plan "Hobby Dev - Free", que le dará acceso a una base de datos PostgreSQL gratuita que admitirá hasta 10,000 filas de datos. Haga clic en "Provisión".
7. Ahora, haga clic en el enlace "Heroku Postgres :: Base de datos".
8. Ahora debería estar en la página de resumen de su base de datos. Haga clic en "Configuración" y luego "Ver credenciales". Esta es la información que necesitará para iniciar sesión en su base de datos. Puede acceder a la base de datos a través de Adminer ,

completando el servidor (el "Host" en la lista de credenciales), su nombre de usuario (el "Usuario"), su contraseña y el nombre de la base de datos, todo lo cual puede encontrar en el Página de credenciales de Heroku.

Alternativamente, si instala PostgreSQL en su propia computadora, debería poder ejecutar `psql URI` en la línea de comando, donde `URI` está el enlace provisto en la lista de credenciales de Heroku.

Python y matraz

1. Primero, asegúrese de instalar una copia de Python . Para este curso, debe usar Python versión 3.6 o superior.
2. También deberás instalarlo `pip` . Si descargó Python del sitio web de Python, es probable que ya lo haya `pip` instalado (puede verificarlo ejecutando `pip` en una ventana de terminal). Si no lo tiene instalado, asegúrese de instalarlo antes de continuar.

Para intentar ejecutar su primera aplicación Flask:

1. Descargue el `project1` directorio de distribución de https://cdn.cs50.net/web/2020/x/projects/1/project1.zip y descomprímalo.
2. En una ventana de terminal, navegue a su `project1` directorio.
3. Ejecútelo `pip3 install -r requirements.txt` en la ventana de su terminal para asegurarse de que estén instalados todos los paquetes de Python necesarios (Flask y SQLAlchemy, por ejemplo).
4. Set the environment variable `FLASK_APP` to be `application.py` . On a Mac or on Linux, the command to do this is `export FLASK_APP=application.py` . On Windows, the command is instead `set FLASK_APP=application.py` . You may optionally want to set the environment variable `FLASK_DEBUG` to `1` , which will activate Flask's debugger and will automatically reload your web application whenever you save a change to a file.
5. Set the environment variable `DATABASE_URL` to be the URI of your database, which you should be able to see from the credentials page on Heroku.
6. Run `flask run` to start up your Flask application.
7. If you navigate to the URL provided by `flask` , you should see the text `"Project 1: TODO"` !

Goodreads API

Goodreads is a popular book review website, and we'll be using their API in this project to get access to their review data for individual books.

1. Go to https://www.goodreads.com/api and sign up for a Goodreads account if you don't already have one.
2. Navigate to https://www.goodreads.com/api/keys and apply for an API key. For "Application name" and "Company name" feel free to just write "project1," and no need to include an application URL, callback URL, or support URL.

3. You should then see your API key. (For this project, we'll care only about the "key", not the "secret".)
4. You can now use that API key to make requests to the Goodreads API, documented here. In particular, Python code like the below

```
import requests
res = requests.get("https://www.goodreads.com/book/review_counts.json", params={"key": "KE
print(res.json())
```

where `KEY` is your API key, will give you the review and rating data for the book with the provided ISBN number. In particular, you might see something like this dictionary:

```
{'books': [{
                'id': 29207858,
                'isbn': '1632168146',
                'isbn13': '9781632168146',
                'ratings_count': 0,
                'reviews_count': 1,
                'text_reviews_count': 0,
                'work_ratings_count': 26,
                'work_reviews_count': 113,
                'work_text_reviews_count': 10,
                'average_rating': '4.04'
            }]
}
```

Note that `work_ratings_count` here is the number of ratings that this particular book has received, and `average_rating` is the book's average score out of 5.

Requirements

Alright, it's time to actually build your web application! Here are the requirements:

- **Registration**: Users should be able to register for your website, providing (at minimum) a username and password.
- **Login**: Users, once registered, should be able to log in to your website with their username and password.
- **Logout**: Logged in users should be able to log out of the site.
- **Import**: Provided for you in this project is a file called `books.csv`, which is a spreadsheet in CSV format of 5000 different books. Each one has an ISBN number, a title, an author, and a publication year. In a Python file called `import.py` separate from your web application, write a program that will take the books and import them into your PostgreSQL database. You will first need to decide what table(s) to create, what columns those tables should have, and how they should relate to one another. Run this program by running `python3 import.py` to

import the books into your database, and submit this program with the rest of your project code.

- **Search**: Once a user has logged in, they should be taken to a page where they can search for a book. Users should be able to type in the ISBN number of a book, the title of a book, or the author of a book. After performing the search, your website should display a list of possible matching results, or some sort of message if there were no matches. If the user typed in only part of a title, ISBN, or author name, your search page should find matches for those as well!
- **Book Page**: When users click on a book from the results of the search page, they should be taken to a book page, with details about the book: its title, author, publication year, ISBN number, and any reviews that users have left for the book on your website.
- **Review Submission**: On the book page, users should be able to submit a review: consisting of a rating on a scale of 1 to 5, as well as a text component to the review where the user can write their opinion about a book. Users should not be able to submit multiple reviews for the same book.
- **Goodreads Review Data**: On your book page, you should also display (if available) the average rating and number of ratings the work has received from Goodreads.
- **API Access**: If users make a GET request to your website's `/api/<isbn>` route, where `<isbn>` is an ISBN number, your website should return a JSON response containing the book's title, author, publication date, ISBN number, review count, and average score. The resulting JSON should follow the format:

```
{
    "title": "Memory",
    "author": "Doug Lloyd",
    "year": 2015,
    "isbn": "1632168146",
    "review_count": 28,
    "average_score": 5.0
}
```

If the requested ISBN number isn't in your database, your website should return a 404 error.

- You should be using raw SQL commands (as via SQLAlchemy's `execute` method) in order to make database queries. You should not use the SQLAlchemy ORM (if familiar with it) for this project.
- In `README.md`, include a short writeup describing your project, what's contained in each file, and (optionally) any other additional information the staff should know about your project.
- If you've added any Python packages that need to be installed in order to run your web application, be sure to add them to `requirements.txt`!

Beyond these requirements, the design, look, and feel of the website are up to you! You're also welcome to add additional features to your website, so long as you meet the requirements laid out in the above specification!

Hints

- At minimum, you'll probably want at least one table to keep track of users, one table to keep track of books, and one table to keep track of reviews. But you're not limited to just these tables, if you think others would be helpful!
- In terms of how to "log a user in," recall that you can store information inside of the `session`, which can store different values for different users. In particular, if each user has an `id`, then you could store that `id` in the session (e.g., in `session["user_id"]`) to keep track of which user is currently logged in.

FAQs

For the API, do the JSON keys need to be in order?

Any order is fine!

`AttributeError: 'NoneType' object has no attribute '_instantiate_plugins'`

Make sure that you've set your `DATABASE_URL` environment variable before running `flask run`!

How to Submit

IMPORTANT: *On 1 July 2020, CS50W will update to a new version and this version of this project will no longer be accepted for credit. If you haven't completed the entire course by that date, that's okay! If you ultimately receive a passing grade on this project, we will consider it equivalent in scope and content to Project 2 in the new version of the course, and within two weeks after 1 July 2020 your gradebook at https://cs50.me/cs50w will update to reflect that you have received credit for having completed Project 2.*

1. If you haven't already done so, visit this link, log in with your GitHub account, and click **Authorize cs50**. Then, check the box indicating that you'd like to grant course staff access to your submissions, and click **Join course**.
2. If you've installed `submit50`, execute

   ```
   submit50 web50/projects/2020/x/1
   ```

   Otherwise, using Git, push your work to `https://github.com/me50/USERNAME.git`, where `USERNAME` is your GitHub username, on a branch called `web50/projects/2020/x/1`.
3. Record a 1- to 5-minute screencast in which you demonstrate your app's functionality and/or walk viewers through your code. Upload that video to YouTube (as unlisted or public, but not private) or somewhere else. To aid in the staff's review, in your video's description on

YouTube, you should timestamp where, minimally, each of the following occurs. This is **not optional**; failure to do this will result in your submission being rejected:

- Search (showing multiple results for a query)
- Individual book page showing leaving a review
- API page

4. Submit this form.

You can then go to https://cs50.me/cs50w to view your progress!