Books Wishlist API Project Description
Kevin Coryell
10/16/2018

Abstract
Project contains a RESTful API used for managing a user's books wishlist. It is written in Python using the Flask framework and associated extensions.

Assumptions

- Not intended to be run in a production environment
- Only one user at a time will be using API

Requirements

1. API shall provide way for users to manage a books wishlist
    a. Add books to wishlist
    b. View books in wishlist
    c. Update books in wishlist
    d. Delete books from wishlist
2. App shall be written in Python
3. API shall be RESTful

Technology
With overhead and portability in mind, this app utilizes the Flask microframework. Other frameworks for developing web apps in Python (e.g. Django) are more complex to setup and not as well suited for a non-enterprise application. Flask also offers a number of developer-friendly add-ons.

The API endpoints rely upon the Flask RESTful extension (flask-restful), which contains many built-in features for supporting REST APIs and reduces the amount of code. Endpoint access is managed via the Flask HTTP auth extension (flask-httpauth), implementing HTTP Basic authentication. Since this is not going to reach a production environment, many URLs can be reached anonymously and more advanced authentication methods were deemed unnecessary.

The database management system for this app is SQLite, which is natively supported in Python 3 and highly portable. The Flask SQLAlchemy extension provides an ORM interface to all database interactions.

Design
As provided at the onset, the two resources involved are users and books. Each user then must have their own books wishlist (i.e. nested resource), which could contain many books. The same book may also be included in multiple wishlists. Hence, the database is modeled as such:

Users

- id (auto-generated)
- first name
- last name
- email
- password (stored as a hash)

Books

- isbn (user provided)
- title
- author
- date of publication

Users books wishlist

- id
- isbn

Endpoints are mapped out as follows:

/api/users
/api/users/<id>
/api/users/<id>/books
/api/users/<id>/books/<isbn>
/api/books
/api/books/<isbn>
/api/books/<isbn>/users

All endpoints contain GET methods, while users can be added via POST to /api/users and books can be added via POST to /api/books. Books can be added to a user's wishlist via POST to /api/users/<id>/books or via PUT to /api/users/<id>/books/<isbn>. Books added to a user's wishlist will be added to the global books library, if not there already. Books in a user's wishlist can be updated via PUT to /api/users/<id>/books/<isbn>. The same endpoint can be used to delete a book from a user's wishlist via the DELETE method.

Users can only make changes to their own wishlists, supplying credentials in an HTTP Basic authorization header with their email as their username. This header is only required for making changes to a user's wishlist. Users cannot be updated or deleted, and books can only be updated or deleted via a user's wishlist. While book attribute updates (via /api/users/<id>/books/<isbn>) take effect globally, books cannot be deleted from the global library. All valid request and response structures use JSON.

All development completed using Python 3 in Cygwin with packages installed via pip.

Tests

Testing completed via a mix of endpoint integration testing (using Python's unittest module alongside Flask's test client) and functional testing with curl and Postman. To be certain, there are many flaws that could be easily exploited.