CODING NOMADS

18) Going RESTful: Build an API Lesson

Lab: Create an API Blueprint for Your Flask App

5 min to complete · By Brandon Gigous

Contents

Note: This lesson builds on a project that's been gradually worked on throughout this Python + Flask course.

By this point, you and your users will have the ability to create lots of content on your website! Users, compositions, and comments galore.

Content means information, which may be valuable to people who use or interact with your website. However, right now the only way for someone to access all the information is by navigating to the site, logging in, and manually clicking around to get the information the user seeks. You and I are tech-minded people; we live in the 21st century, so there's gotta be a better way, a *programmatic* way, to get information from your site... right?

Revisiting APIs

Of course there is! Most modern, well-known webapps have something called an **Application Programming Interface (API)** which allow developers, or *clients*, to query for information about the site's content, and even add or edit content as allowed by the webapp, or *server*.

From the server side (that means you), there's a lot to consider in carefully providing this content from your own API, like permissions, error handling, and serializing the data. The good news is you've already done much of this in the course already, but there is a bit of extra work that must be done to integrate permissions, error handling, and the like with an API. You'll learn more about it in this upcoming course section.

API Blueprint For Your Webapp

The next few lessons will show you how to implement your Flask app's **REST** API. REST is an architecture for accessing and providing *resources* in a stateless manner.

To get you started with developing your API in Flask, you'll first need to create a blueprint for your API.

Learn by Doing 

Create and Register

The API blueprint will live in `app/api`. Create this directory and create a blueprint `api`.

Register your blueprint just like you did for your other ones. **Important:** set your new blueprint's `url_prefix` as `/api/v1`.

Commonly, an API has different versions as the API goes through updates and enhancements, with the old API version still in place so that other peoples' apps that interface with the API don't immediately break. The `v1` just means this is version one of your API. If you come out with a new one, you can keep the old version in place and also have a new, fancier version.

Add Other Modules

While you're at it, create these other files as part of your API blueprint:

```
api/
├── __init__.py (your blueprint creation)
├── authentication.py
├── comments.py
├── compositions.py
├── decorators.py
├── errors.py
└── users.py
```

Then, import all but `decorators.py` within your blueprint.

Summary: API Blueprint

- An **Application Programming Interface (API)** allows developers, or *clients*, to query for information about the site's content, and even add or edit content as allowed by the webapp, or *server*.
- The API blueprint will live in `app/api`. Create this directory and create a blueprint `api`. Don't forget to register
- Set your new blueprint's `url_prefix` as `/api/v1`.
- create these other files as part of your API blueprint:

```
api/
├── __init__.py (your blueprint creation)
├── authentication.py
├── comments.py
├── compositions.py
├── decorators.py
├── errors.py
└── users.py
```

- Then, import all but `decorators.py` within your blueprint.