Corndel DevOps Engineering Programme

in association with Softwire

Module 2: Project Exercise

Corndel Digital.

DA6G

Module 2

Integrating with an API

Introduction

We'll use this exercise to build on the to-do app we made in module 1.

So far, we've been storing the list of to-do items in a browser cookie (accessed in the app using the `session` object provided by Flask). This has some drawbacks, the main one being that if you visit the app in another browser (or from another machine), all your carefully created tasks will no longer be available, because they're only saved in a cookie within the browser you used to create them. Can you think of any other drawbacks or limitations to using cookies to save the to-do list?

In this exercise, we'll use a web service called **Trello** to store and manage the to-do items for our app. Trello is a web-based list-making app popular among many dev teams because it provides a lightweight way to track work. Fortunately for us, it also provides a nice REST API that we can use to create, read and update to-do items. In Trello, items or tasks are called *cards* and they are grouped together as *lists* ('To Do', 'Done', etc.) under *boards*. You can read about the basic concepts used by Trello in this overview.

You'll need to create a Trello account and a dedicated board to store the to-do items for your app. Then you can replace the existing app logic to get and create items with equivalent calls to Trello's REST API instead. This means that you'll be able to access your to-do list from any computer (or even directly within Trello's web app)!

This exercise should give you plenty of practice using REST APIs. It's also a good opportunity to look for ways to refactor your code to improve its structure and readability.

Exercise structure

Each project exercise will follow the same structure:

1. **Introduction:** what we'll be adding to the course project in this exercise.
2. **Setup:** any steps you need to follow first, in order to be able to complete the exercise.
3. **Exercise:** the core goals that you'll need to complete before submitting your work.
4. **Stretch goals:** optional additions you can make to the project that go beyond the core concepts for this module.
5. **Hints:** some pointers to helpful resources if you find yourself stuck.

Setup

Step 1: Prepare your starter project

Start off with your project code from the Module 1 exercise. Your pull request from the last module should be merged into master by now, allowing you to create a fresh new branch off master (`module-2` or `exercise-2` are good names - maintain whichever convention you established in the first module).

Alternatively, if you'd prefer not to work with your existing code, you can checkout a branch in your fork of the project that contains starter code for this module here: https://github.com/CorndelWithSoftwire/DevOps-Course-Exercise/commits/Exercise-1-Solution.

Step 2: Create a Trello account and API key

We're going to be using Trello's API to fetch and save to-do tasks. In order to call their API, you need to first create an account, then generate an API key and token by following the instructions here.

Submitting your work

We'll be using Pull Requests on GitHub in the same fashion as the last module to review this exercise submission.

If you experience issues, ask a tutor for help!

Exercise

Part 1: Explore the Trello API

You can read the documentation for Trello's REST API here. Use Postman to explore the API. It might be helpful to create a new board and a few example cards on the Trello web app, then see if you can fetch them and update them via the API using Postman. Can you work out how to move a card from 'To Do' to

'Done'? (Hint: this will involve changing the list that the card belongs to.)

Part 2: Keep your secrets safe!

All of these API calls require an API key and token to authenticate the request. These credentials are tied to your account and need to be kept secret! For this reason, they shouldn't be committed to your project's Git repository. There are a number of ways of specifying these credentials separately and then having the app read them in, such as using a separate configuration file or specifying environment variables.
Whichever way you choose to store and read those secrets, make sure that any files containing the secret values themselves are added to the .gitignore file in your project repository, to make sure they cannot be committed by accident!

Part 3: Replace existing app logic to call the Trello API instead

Now that you're familiar with the API and how to fetch, create and modify cards, let's update the to-do app to call the Trello API.
Python has built-in modules for making web requests, but there are libraries available that make things easier. We'll be using an external library called **Requests** to make the API requests and interpret the JSON response.
Install the library using `pip` and update the `requirements.txt` file in the project:

```
$ pip install requests
$ pip freeze > requirements.txt
```

Requests should now be installed and ready for use in the app. Now is a good point to spend some time learning how to use it. In general, when looking for libraries that contain the functionality

Don't forget to commit!

Make sure you use **`git commit`** to save your work regularly.

It's good practice, and lets you revert back to a previous revision if you find you've broken everything horribly!

Try to make your commit messages concise and descriptive. Using source control tools effectively is just as important as the code you submit.

you want, you should choose a library that is *well-documented* (so that learning how to use it is as quick as possible) and *popular* (so you are more likely to get search results containing solutions to your problems). Accordingly, Requests is very popular, and its documentation is good too. A quickstart guide is available here.

Have a look at the methods available on the `requests` object and work out how to call the Trello API from your code. You will need to read the result, convert it into some sort of object, and replace the calls to `session_items` methods with calls to your new methods instead. The `json()` method provided by Requests should help you to convert the JSON response into a Python dictionary.

Don't forget to commit your work often, making it easier to back-track if you run into problems.

The actions you will need to achieve by calling the Trello API are:

- Fetch all to-do items (cards) for the specified board
- Create a new card on the board's 'To Do' list
- Move a card from 'To Do' to 'Done' (or 'Doing' if you want to allow in-progress to-do items)

(**Note:** some of these actions may require *more than one* API call, depending on how you implement them.)

Part 4: Change the status of to-do items

Let's extend the functionality of the app by using more of the Trello API calls we looked at earlier.

We need a way to mark items as completed. Add a `complete_item` route that accepts the ID of an item as a parameter and then calls a method to change its status from 'To Do' to 'Done' using the appropriate Trello API call(s). You'll then need to expose that functionality through the UI: modify the `index.html` template to display a button for each item with a link to the route you just created. The `url_for()` method accepts keyword arguments that can be used to specify the item ID in each link.

If you wanted to go further, you could add another route and logic to change status from 'Done' back to 'To Do'.

Part 5: Create a class for to-do items

So far, you have probably been representing our items just as objects. Something like this:

```
{'id': 1, 'status': 'Not Started',
'title': 'List saved todo items'}
```

That's been fine, but we can probably make the code a bit cleaner and more readable now by adding an `Item` class. Having a class will allow us to:

- define what properties an item has (e.g. `id`, `status` and `title`)
- provide a common place to add methods for creating a new `Item` (e.g. how to create one using a response from the Trello API)

Stretch goals

- How clean is your code? Is there anything you could do to improve readability?
- Sort items by status.
- Add item descriptions.
- Add due dates to items.
  - You may want to use a date-picker library, such as bootstrap-datepicker (documentation available here). Note that Bootstrap and jQuery are already included in the project, so there's no need to add them.

Hints

- If you're having trouble getting an API call to work, try adding some logging to your app to see what data it is sending/ receiving.
- Trello boards are created with the following lists by default: 'To Do', 'Doing', 'Done'.