

Flask 1: listing users / viewing user

Organizing your projects

- See 'note2-organization.md'

Hello world

- name it app.py
- flask run --debug

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

Rendering templates

- I'm using app.get to specify the GET method
- make sure that your html files are in a 'templates/' subdirectory
- navigation will go: home > accounts > users

```
@app.get("/")
def index():
    return render_template('index.html')

@app.get("/accounts/")
def accounts():
    return render_template('accounts.html')
```

Links

- use `url_for` to dynamically generate links.
- `url_for` refers to the function name, not the route.
- this way, things won't break if you change routes.

```
<a href="{{url_for('accounts')}}">Account management</a>
```

Dynamic content

- the user data will come from the API
- this can happen now, but it doesn't need to

```
@app.get("/users/")
def list_users():

    # TODO: get user data
    users = []

    return render_template('list_users.html', users=users)
```

Getting data from your API

- define your service / API url
- load your data just like you did in your api tests
- GET users returns a UserCollection as a JSON string
- when you use `json.loads`, it converts to a dict
- finally, we can get the user list

```
res = requests.get(f'{url}/users/')  
users = json.loads(res.text).get('users')
```

Mocking

- For development, it is sometimes helpful to mock data.
- This is especially helpful if you are developing the App and API simultaneously.
- As long as you agree on the API specification, the pieces should fit together later

Creating some data using Faker

- create one mock user

```
from user_models import *
from faker import Faker
fake = Faker()

def get_user():
    user = UserOut(
        _id=fake.uuid4(),
        username=fake.user_name(),
        password=fake.password()
    )
    return user
```

Creating many

- Note that this would have been helpful when we were testing our DB and API.

```
def get_users(num_users=10):  
    users = []  
    for _ in range(num_users):  
        users.append(get_user())  
    return users
```

Populating your user listing

- use a jinja for-loop to list the users
- note the syntax
 - {% %} for commands
 - {{ }} for evaluating variables

```
{% for user in users %}  
  <li> {{ user.username }} </a> </li>  
{% endfor %}
```