Routes

Web applications

- Client-Server model
- Stateless: server does not know the present state of the client
  - Must be ready to respond to whatever the client requests without assuming anything about the client
- Requests sent through HTTP protocol
  - Use variants of the GET, POST (Verbs) to convey **meaning**
  - Use URL (Uniform Resource Locator) structure to convey **context**

Routing: mapping URLs to actions

Python Decorators

- Add extra functionality on top of a function
- “@” - decorators before function name
- Effectively function of a function:
  - Take the inner function as an argument
  - Return a function that does something before calling the inner function

Basic routing in Flask

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def home():
    return "Hello World!"
```

HTTP verbs

```
@app.route('/', methods=['GET'])
def index():
    ...
@app.route('/create', methods=['POST'])
def store():
    ...
```

CRUD-like functionality

```
# Assume 'index', 'store' etc are functions


@app.route('/', methods=['GET'])(index)
@app.route('/create', methods=['POST'])(store)
@app.route('/<int:user_id>', methods=['GET'])(show)
@app.route('/<int:user_id>/edit', methods=['POST'])(update)
@app.route('/<int:user_id>', methods=['DELETE'])(destroy)
```

Summary

- Flask is not natively MVC
  - But MVC is more a way of thought than a framework
- Simple URL routing
- Helpers to do large scale routing of common functions

Structure the application with separation of concerns in mind

- MVC just one way to achieve clean design