Flask Tutorial

Rafael Castro G. Silva

Department of Computer Science
University of Copenhagen

20/05/2025

Learning Goals

Learning Goals

1. Learn what is Flask, and some basic Flask concepts
2. Learn the Model View Control (MVC) pattern
3. Learn basic Docker

Flask Introduction

What is Flask?

What is Flask?

- Flask is a micro framework for web applications

What is Flask?

- Flask is a micro framework for web applications
  - Framework: a set of modules and libraries that provide basic functionalities
    May or may nor enforce some standards (e.g. directory structures, MVC pattern, etc)
  - Micro: enforces very little
  - In/for Python

What is Flask?

- Flask is a micro framework for web applications
  - Framework: a set of modules and libraries that provide basic functionalities
    May or may nor enforce some standards (e.g. directory structures, MVC pattern, etc)
  - Micro: enforces very little
  - In/for Python

Other frameworks out there

- Ruby on Rails
- Django (Python)
- Java Spring
- Symfony (PHP)

Who uses Flask?

Who uses Flask?

Uber

What do I need know?

What do I need know?

- Very basic Python
  - Variables, functions, indentation, lists, maps (dictionaries), if-then-else, for loops, class (OO programming), import libraries
- Terminal commands: `ls`, `cd`, `pwd`...
- Basic HTML
- CSS (optional)
- SQL

Flask Concepts

The App Variable

- Our app is an instance of the `Flask` class
- `__name__` is a default configuration telling that the files of the project are in the current directory

```
1 from flask import Flask
2
3 app = Flask(__name__)
```

Routing

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/")
6 def hello_world():
7     return "<p>Hello, World!</p>"
```

- `@app.route("/")` is Python decorator: extends the behavior of a function
  - Tells Flask what URL should trigger our function
  - The `"/"` is the root of our web site domain
    `http://127.0.0.1:5000` takes us to this route
  - We can have other routes: `@app.route("/register")`
    `http://127.0.0.1:5000/register` takes us to this other route

Routing

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/")
6 def hello_world():
7     return "<p>Hello, World!</p>"
```

- `@app.route("/")` is Python decorator: extends the behavior of a function
  - Tells Flask what URL should trigger our function
  - The `"/"` is the root of our web site domain
    `http://127.0.0.1:5000` takes us to this route
  - We can have other routes: `@app.route("/register")`
    `http://127.0.0.1:5000/register` takes us to this other route

What else to learn:

- Variable rules:
  `https://flask.palletsprojects.com/en/stable/quickstart/#variable-rules`
- HTTP methods:
  `https://flask.palletsprojects.com/en/stable/quickstart/#http-methods`

Templates

File ./app.py:

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', person=name)
```

File ./templates/hello.html:

```
<!doctype html>
<title>Hello from Flask</title>
{% if person %}
  <h1>Hello {{ person }}!</h1>
{% else %}
  <h1>Hello, World!</h1>
{% endif %}
```

Templates

File ./app.py:

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', person=name)
```

File ./templates/hello.html:

```
<!doctype html>
<title>Hello from Flask</title>
{% if person %}
  <h1>Hello {{ person }}!</h1>
{% else %}
  <h1>Hello, World!</h1>
{% endif %}
```

- Import `render_template` (Jinja2 template engine)
- 2 routes for the same function
- The function has a named argument

Templates

File ./app.py:

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', person=name)
```

File ./templates/hello.html:

```
<!doctype html>
<title>Hello from Flask</title>
{% if person %}
  <h1>Hello {{ person }}!</h1>
{% else %}
  <h1>Hello, World!</h1>
{% endif %}
```

- Import `render_template` (Jinja2 template engine)
- 2 routes for the same function
- The function has a named argument
- `render_template` passes the named arguments
- The template has access to the Python variables.
  Inside of {% %} is Jinja2 code
  Inside of {{ }} is to print as HTML

What else to learn about templates?

- If statements: `https://jinja.palletsprojects.com/en/stable/templates/#if`
- For loops: `https://jinja.palletsprojects.com/en/stable/templates/#for`
- Template Inheritance
  `https://jinja.palletsprojects.com/en/stable/templates/#template-inheritance`

MVC (extra)

Model View Controller

Model View Controller

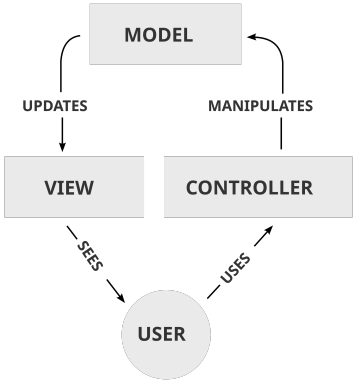
- Model View Controller (MVC) is a software architectural pattern

Model View Controller

- Model View Controller (MVC) is a software architectural pattern

We split the applications in:

- Model: The internal representation of the information (e.g., our domain objects)
  - Interact with the database
  - Busyness logic
  - Do complex computations
- View: The interface for the user
  - Builds the HTML/Javascript/JSON/Text
- Controller: Links model and view



Docker (extra)

Docker Introduction

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)
- I use Linux, you may use Windows, your next group colleague may use MacOS

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)
- I use Linux, you may use Windows, your next group colleague may use MacOS
- How do we make sure everybody has the exact same environment?

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)
- I use Linux, you may use Windows, your next group colleague may use MacOS
- How do we make sure everybody has the exact same environment?

Solution:

- We enforce the same environment in declarative manner:
  version of system libraries, databases, the environment variables...
- Docker does exactly that!

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)
- I use Linux, you may use Windows, your next group colleague may use MacOS
- How do we make sure everybody has the exact same environment?

Solution:

- We enforce the same environment in declarative manner:
  version of system libraries, databases, the environment variables...
- Docker does exactly that!

What is Docker?

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)
- I use Linux, you may use Windows, your next group colleague may use MacOS
- How do we make sure everybody has the exact same environment?

Solution:

- We enforce the same environment in declarative manner:
  version of system libraries, databases, the environment variables...
- Docker does exactly that!

What is Docker?

- What Docker is not: A virtual machine

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)
- I use Linux, you may use Windows, your next group colleague may use MacOS
- How do we make sure everybody has the exact same environment?

Solution:

- We enforce the same environment in declarative manner:
  version of system libraries, databases, the environment variables...
- Docker does exactly that!

What is Docker?

- What Docker is not: A virtual machine
- It is a container solution based on a Linux kernel feature called cgroups
- It is also runs on MacOS and Windows

Docker Introduction

The problem:

- Software depends on specific versions of system libraries, and in the computer environment in general (e.g. directory structures, databases, ambient variables, etc)
- I use Linux, you may use Windows, your next group colleague may use MacOS
- How do we make sure everybody has the exact same environment?

Solution:

- We enforce the same environment in declarative manner:
  version of system libraries, databases, the environment variables...
- Docker does exactly that!

What is Docker?

- What Docker is not: A virtual machine
- It is a container solution based on a Linux kernel feature called cgroups
- It is also runs on MacOS and Windows

Why use Docker?

- Like git, it is a standard in the industry
- It will make the life of the TAs way easier

An Usual Setup

The Bank Example Project

The Bank Example Project Overview

Let's write our web apps!

Hands-on

- Follow my step-by-step guide: `https://github.com/rafaelcgs10/dis2025`
- Follow the official Flask tutorial: `https://flask.palletsprojects.com/en/stable/tutorial/`