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?

airbnb

trivago

reddit

Uber

Prerequisites

Prerequisites

- 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

```python
from flask import Flask

app = Flask(__name__)
```

Views (Routing)

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    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

Views (Routing)

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    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`:

```python
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`:

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

Templates

File `./app.py`:

```python
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`:

```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`:

```python
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`:

```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:

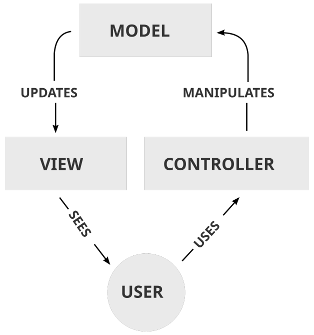
- Model: The internal representation of the information
  - Interacts with the database
  - Business logic (e.g. transfer money)
- View: The interface for the user
  - Builds the HTML/Javascript/JSON/Text
- Controller: Links model and view
  - Routes
  - Application logic: how to handle requests

Model View Controller

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

We split the applications in:

- Model: The internal representation of the information
  - Interacts with the database
  - Business logic (e.g. transfer money)
- View: The interface for the user
  - Builds the HTML/Javascript/JSON/Text
- Controller: Links model and view
  - Routes
  - Application logic: how to handle requests
  - Business logic?

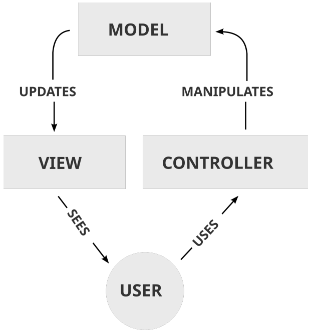


Model View Controller

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

We split the applications in:

- Model: The internal representation of the information
  - Interacts with the database
  - Business logic (e.g. transfer money)
- View: The interface for the user
  - Builds the HTML/Javascript/JSON/Text
- Controller: Links model and view
  - Routes
  - Application logic: how to handle requests
  - Business logic?



MVC in Flask

The Flask terminology does not quite fit the MVC model!

- Model: Python classes in a `models` folder
- View: Templates in the `templates` folder
- Controller:
  Flask views in a `controllers` folder, or a single `app.py` file

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

Docker basics

Docker basics

More concepts

- Docker container: isolated process with its own file system
- Docker image: is a package that includes all of the files, binaries, libraries, and configurations to run a Docker container
    - Built on layers: make your own image on top of existing ones
- Docker volume: persistent data stores for containers

Docker basics

More concepts

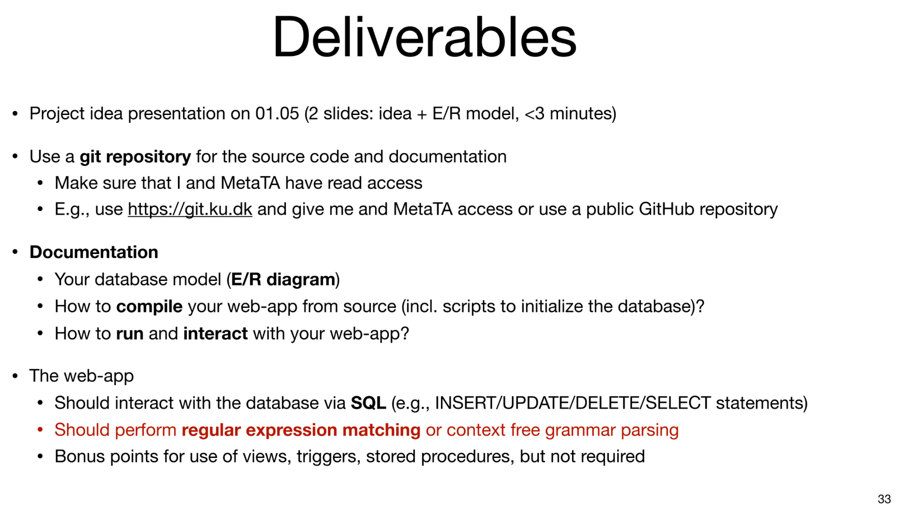
- Docker container: isolated process with its own file system
- Docker image: is a package that includes all of the files, binaries, libraries, and configurations to run a Docker container
    - Built on layers: make your own image on top of existing ones
- Docker volume: persistent data stores for containers

An usual setup:

- `Dockerfile`: The Docker image in which you main app will run on
- `entrypoint.sh`: The script that the Dockerfile calls when its container is starting up
- `docker-compose.yml`: Organizes all the different Docker container of your application

Let's write our web apps!

Requirements Reminder



Deliverables

- Project idea presentation on 01.05 (2 slides: idea + E/R model, <3 minutes)
- Use a **git repository** for the source code and documentation
  - Make sure that I and MetaTA have read access
  - E.g., use https://git.ku.dk and give me and MetaTA access or use a public GitHub repository
- **Documentation**
  - Your database model (**E/R diagram**)
  - How to **compile** your web-app from source (incl. scripts to initialize the database)?
  - How to **run** and **interact** with your web-app?
- The web-app
  - Should interact with the database via **SQL** (e.g., INSERT/UPDATE/DELETE/SELECT statements)
  - Should perform **regular expression matching** or context free grammar parsing
  - Bonus points for use of views, triggers, stored procedures, but not required

33

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/`
- Study the examples from previous years in Absalon (Bank, GreenGroceries, and nft-crypto-punk)