Python API Development with Flask and Heroku

Pertemuan 11

MK Algoritma Pemrograman II

M. N. Fakhruzzaman, S.Kom., M.Sc.

Ika Qutsiati Utami, S.Kom., M.Sc.

Program Studi S1 Teknologi Sains Data

Fakultas Teknologi Maju dan Multidisiplin

Universitas Airlangga Indonesia

Outline

- Flask Recap
- Heroku as a Platform (PaaS)
- API Recap
- Making RESTful API with Flask
- Deploying to Heroku!

- **FINAL PROJECT ANNOUNCEMENT**

Flask Recap

- Flask is a web framework, written in Python
- Flask can be used as a base for any web application with user interface, or even headless webapp/webservice

```python
from flask import Flask, redirect, url_for, request
app = Flask(__name__)

@app.route('/success/<name>')
def success(name):
   return 'welcome %s' % name

@app.route('/login',methods = ['POST', 'GET'])
def login():
   if request.method == 'POST':
      user = request.form['nm']
      return redirect(url_for('success',name = user))
   else:
      user = request.args.get('nm')
      return redirect(url_for('success',name = user))
```

```html
<html>
   <body>
      <form action = "http://localhost:5000/login" method = "post">

<p>Enter Name:</p>


<p><input type = "text" name = "nm" /></p>


<p><input type = "submit" value = "submit" /></p>

      </form>
   </body>
</html>
```

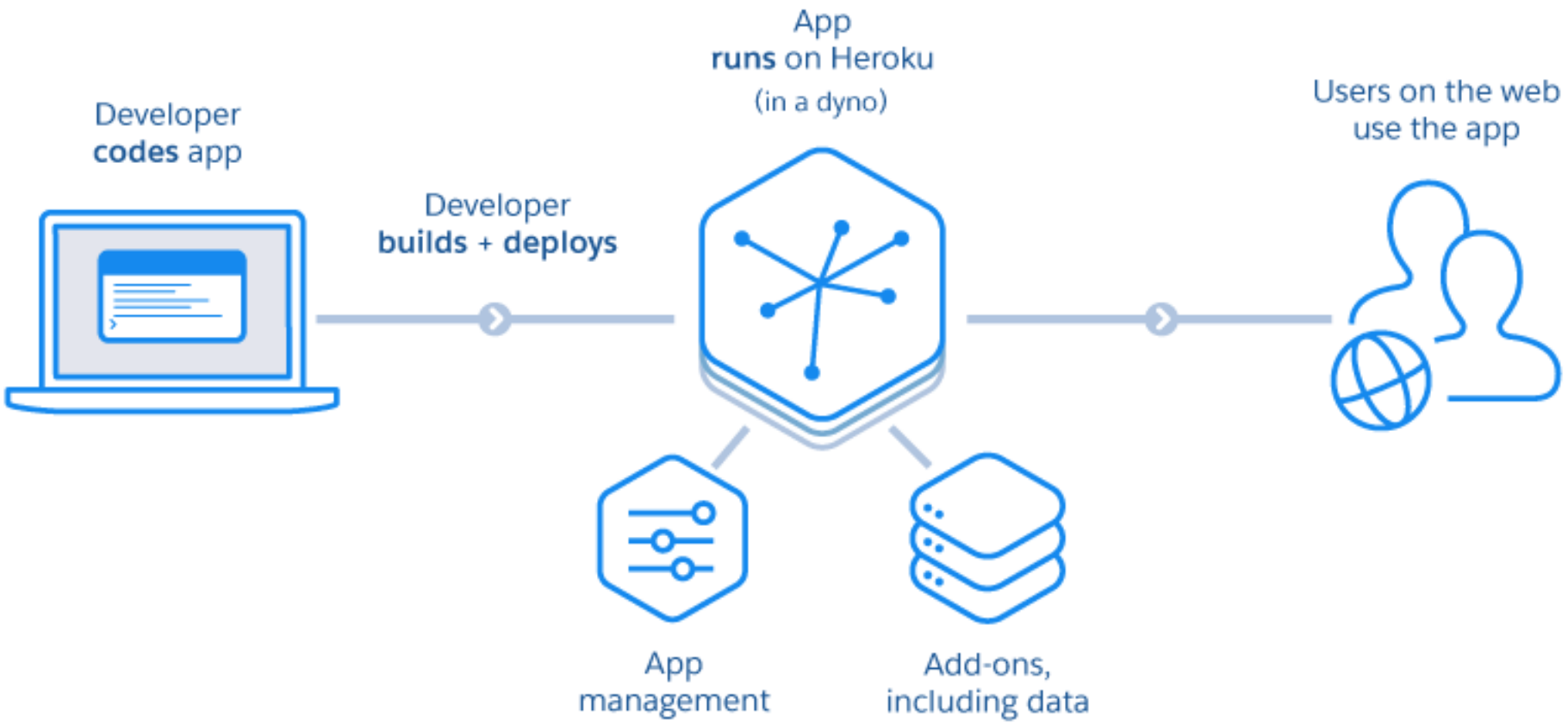
Another trials

- Left for controller
- Right for view / UI

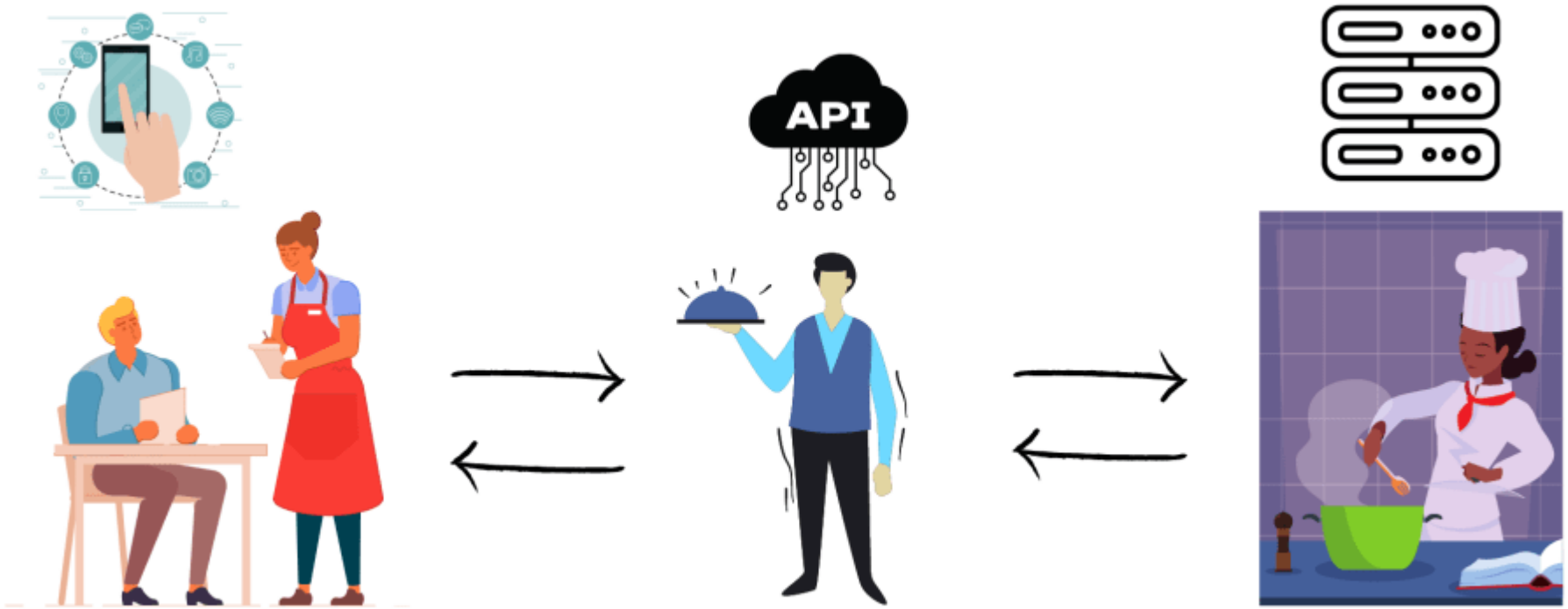
Heroku

- Heroku is a container-based cloud Platform as a Service (PaaS)
- Developers use Heroku to deploy, manage, and scale modern apps
- Free option, with limits of dynos (compute hours) per month
- Good alternative to learn cloud computing and MLOps

Heroku: Platform as a Service

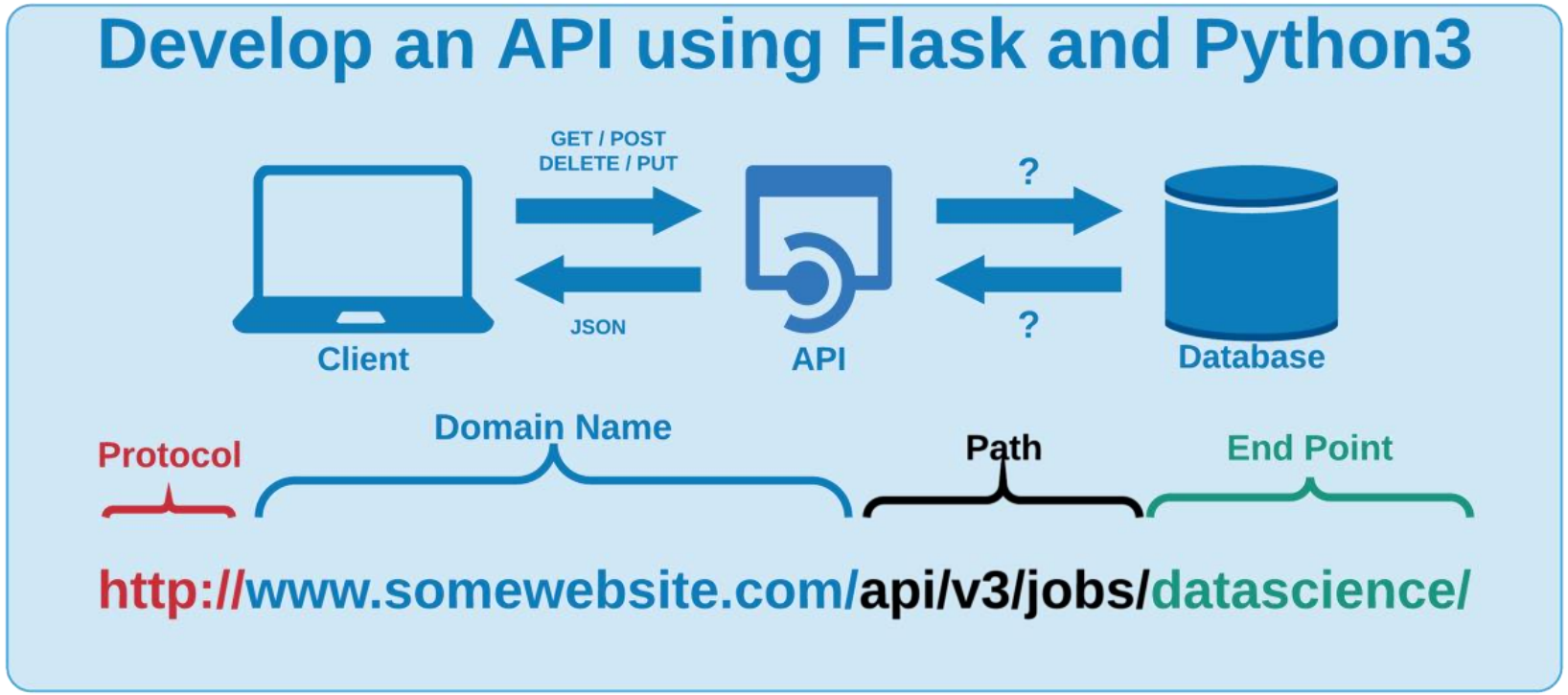


API Recap



RESTful API with Flask

- Flask offers easy to use framework for routing
- Flask can understand HTTP requests
- Flask can be made as an API



RESTful Flask

- Activate / create a new virtual environment for API (covered in previous lecture)
- Install Flask, FlaskRestful, and gunicorn (WSGI)

```
$ pip install Flask
$ pip install flask-restful
```

```
$ pip install gunicorn
```

- Main.py is the boilerplate for flask API
- Wsgi.py is required to run flaskapi through wsgi (serving locally)

FlaskAPI

- Note that the function name is get, which is a specific HTTP Request for GET
- Try to modify the API by adding a new class

```
class Sum(Resource):
    def get(self, a, b):
        return jsonify({'data': a+b})
```

- Class sum, has a GET function requiring a & b as a GET Query / parameters to be returned by the API as the sum of a & b
- Don’t forget to assign resource / route

```
api.add_resource(Sum,'/add/<int:a>,<int:b>')
```

CORS

- CORS stands for Cross-Origin Resource Sharing
- Needed to be stated to allow API created with python to be accessible by other language, such as Javascript

```
pip install Flask-Cors
```

- Don’t forget to import flask cors, and “corsify” the app

```
from flask import Flask, jsonify
from flask_restful import Resource, Api
from flask_cors import CORS

app = Flask(__name__)
api = Api(app)
CORS(app)
```

FlaskAPI without Flask-restful (alternative)

- Flask can still create API without Flask-restful
- Using traditional flask routing, we can utilize into a FlaskAPI (also RESTful)

```python
import flask
from flask import request, jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True

matkul = [
    {'id': 0,
     'course': 'Algoritma Pemrograman 1',
     'lect': 'MN Fakhruzzaman',
     'code': 'SIA107'},
    {'id': 1,
     'course': 'Algoritma Pemrograman 2',
     'lect': 'MN Fakhruzzaman',
     'code': 'SIA206'},
    {'id': 2,
     'course': 'Metodologi Penelitian dan Statistik',
     'lect': 'MN Fakhruzzaman',
     'code': 'PNG687'}
]


@app.route('/', methods=['GET'])
def home():
    return '''<h1>THIS API WORKS!!!!</h1>
<p>selamat ya..</p>'''


# A route to return all of the available entries in our catalog.
@app.route('/api/v1/resources/matkul/all', methods=['GET'])
def api_all():
    return jsonify(matkul)

app.run()
```

GET API with Query?

```python
@app.route('/api/v1/resources/matkul', methods=['GET'])
def api_id():
    # Check if an ID was provided as part of the URL.
    # If ID is provided, assign it to a variable.
    # If no ID is provided, display an error in the browser.
    if 'id' in request.args:
        id = int(request.args['id'])
    else:
        return "Error: No id field provided. Please specify an id."

    # Create an empty list for our results
    results = []

    # Loop through the data and match results that fit the requested ID.
    # IDs are unique, but other fields might return many results
    for mk in matkul:
        if matkul['id'] == id:
            results.append(matkul)

    # Use the jsonify function from Flask to convert our list of
    # Python dictionaries to the JSON format.
    return jsonify(results)
```

To test, serve it first, then navigate to the API
http://127.0.0.1:5000/api/v1/resources/matkul?id=2

HTTP POST

- What if you want to send a ‘secret’ query (can’t be seen on URLs)
- You send a POST Request
- Traditionally used to create new item in database via HTTP, also can be used to deliver ‘invisible’ payload

- To configure your API to process POST requests, simply change the methods to POST. Or name your function with post (flask-restful)

```python
@app.route('/api/v1/resources/matkul', methods=['POST'])
def req_matkul():
    if not request.json or not 'id' in request.json:
        abort(400)
    else:
        results = []
        data = request.get_json()
        for mk in matkul:
            if matkul['id'] == data['id']:
                results.append(matkul)
        return jsonify(results), 201
```

How to invoke POST?

- Use HTML Forms (for button)

```
<form action="http://127.0.0.1:5000/api/v1/resources/matkulsecret" method="post">
  <label for="id">First name:</label>
  <input type="text" id="id" name="id"><br><br>
  <input type="submit" value="Submit">
</form>
```

- Or Javascript (AJAX / Axios)

Using Javascript and Axios

```
<p>ID Matkul:</p>
<input type="text" id="id"></br>
<button id="submit" onclick="reqmatkul()">Submit</button>

<!-- Tempat untuk menerima hasil request axios/ajax -->
<p>Result: <span id="result">No request yet</span></p>
```

```
<!-- axios -->
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <script type="text/javascript">
        // base url
        var base_url = "<your API baseurl>";

        function reqmatkul() {

            // get ID
            var idmatkul = document.getElementById('id').value;

            // predicting msg
            document.getElementById('result').innerHTML = "Searching from Matkuls ..."

            // send post request
            axios({
                method: 'post',
                url: base_url + "/matkulsecret",
                data: {
                    id: idmatkul
                }
            })
            .then(function (response){
                var uuid = response.data.id;
                var course = response.data.course;


                document.getElementById('result').innerHTML = course;


        })
        .catch(function (error) {
            try {
                if (error.response.status == 429){
                document.getElementById('result').innerHTML = "Error!";
                } else {
                    document.getElementById('result').innerHTML = "Error. Try again later";
                }
            } catch {
                document.getElementById('result').innerHTML = "Error. Timeout";
            }
        });}
```

Deploying to HEROKU

- Create Heroku account
- Set up a new app, name it as you wish
- To deploy, we have to create Procfile

```
web: gunicorn  main:app
```

- Create runtime.txt

```
python-3.7.3
```

- Then on your current terminal (with active virtual env)

```
pip freeze > requirements.txt
```

Deploying to Heroku

- Install Heroku CLI
- Open terminal and invoke `heroku login`
- Create your app `heroku create flask-heroku1 --buildpack heroku/python`
- Navigate to your flask project directory, and git init, git add .
- Make your first commit
- `heroku git:remote -a flask-heroku1`
- And Deploy! `git push heroku master`
- Your app will be accessible through https://<appname>.herokuapp.com/
- This will be your API's Base URL

Headless app is Live

- Your API is live on Heroku
- You can access through the browser
- You can add user interface to different python file with different Flask routes
- You can even send post request through your github page!

Refer

https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask

https://medium.com/analytics-vidhya/flask-restful-api-with-heroku-da1ecf3e04bv