Python: REST-API mit Flask und Swagger

Repository

github.com/meisterk/flask-swagger

Anwendung

1. Installation

```
pip3 install flask flask_restx flask_cors
```

2. App starten

```
python3 app.py
```

3. Im Browser auf App zugreifen

```
http://127.0.0.1:5000/
```

Quellcode: app.py

1. Imports, App, Dokumentation, Namespace

```
# Imports
from flask import Flask
from flask_cors import CORS
from flask_restx import Api, Resource, fields

# Setup von Application und Dokumentation
app = Flask(__name__)
CORS(app)  # Cross-Origin Resource Sharing erlauben
api = Api(app,
          version='1.0',
          title='Schueler API mit Swagger',
          description='Eine einfache Schueler API')

# Definition des Namespace
ns = api.namespace('schueler', description='Schueler CRUD-Operationen')
```

2. Schema des Models

```
# Definition des Models (wird automatisch als Teil der API-Dokumentation angezeigt)
schueler = api.model('schueler', {
    'id': fields.Integer(readonly=True, description="Eindeutige Id der Schüler:in"),
    'vorname': fields.String(required=True, description="Vorname der Schüler:in"),
    'nachname': fields.String(required=True, description="Nachname der Schüler:in")
})
```

3. DAO-Klasse mit CRUD-Methoden

```
class SchuelerDAO(object):
    def __init__(self):
        self.counter = 0
        self.schueler = []

    def get(self, id):
        for s in self.schueler:
            if s['id'] == id:
                return s
        api.abort(404, "die Schüler:in {} gibt es nicht.".format(id))

    def create(self, data):
        s = data
        s['id'] = self.counter = self.counter + 1
        self.schueler.append(s)
        return s

    def update(self, id, data):
        s = self.get(id)
        s.update(data)
        return s

    def delete(self, id):
        s = self.get(id)
        self.schueler.remove(s)
```

4. DAO-Objekt mit Beispieldaten

```
DAO = SchuelerDAO()
DAO.create({'vorname': 'Anna', 'nachname': 'Arm'})
DAO.create({'vorname': 'Berta', 'nachname': 'Bein'})
DAO.create({'vorname': 'Carla', 'nachname': 'Copf'})
```

5. API-Endpoints

```
# API Endpoints: /schueler/ (GET, POST)
@ns.route('/')
class SchuelerList(Resource):
    @ns.doc('list_schueler')
    @ns.marshal_list_with(schueler)
    def get(self):
        '''Alle Schüler:innen auflisten'''
        return DAO.schueler

    @ns.doc('create_schueler')
    @ns.expect(schueler)
    @ns.marshal_with(schueler, code=201)
    def post(self):
        '''Neue Schüler:in erstellen'''
        return DAO.create(api.payload), 201


# API Endpoints: /schueler/{id} (GET, DELETE, PUT)
@ns.route('/<int:id>')
@ns.response(404, 'Schueler not found')
@ns.param('id', 'The schueler identifier')
class Schueler(Resource):
    @ns.doc('get_schueler')
    @ns.marshal_with(schueler)
    def get(self, id):
        '''Hole eine Schüler:in anhand ihrer Id'''
        return DAO.get(id)

    @ns.doc('delete_schueler')
    @ns.response(204, 'Schueler deleted')
    def delete(self, id):
        '''Lösche eine Schüler:in anhand ihrer Id'''
        DAO.delete(id)
        return '', 204

    @ns.expect(schueler)
    @ns.marshal_with(schueler)
    def put(self, id):
        '''Verändere eine vorhandene Schüler:in anhand ihrer Id'''
        return DAO.update(id, api.payload)
```

6. App starten

```
if __name__ == '__main__':
app.run(debug=True)
```