챕터 8 쿠버네티스 마이크로서비스아키텍처 프로젝트

Created	2022년 1월 6일 오후 9:07
Tags	비어 있음

VSCode 구축 환경 만들기

vscode 도커 컨테이너 개발 환경을 구성하는 방법을 모른다면 다음 영상을 참조해 vscode 환경을 먼저 구성하도록 하자.

https://www.youtube.com/watch?v=PgfH94rWsv8

Flask 실습 환경 구성

도커 및 도커 컴포즈 설치

```
apt update && apt install docker.io docker-compose -y
```

도커 컴포즈를 활용해 엘라스틱서치 및 flask 예제 설치

```
cat <<EOF > docker-compose.yaml version: '3.7' services: flask: image: gasbugs/flask-example container_name: flask command: sleep infinity ports: - 80:5000 # Elasticsearch Docker Images: https://www.docker.elastic.co/ elasticsearch: image: docker.elastic.co/elasticsearch/elasticsearch:7.16.2 container_name: elasticsearch environment: - xpack.security.enabled=false - discovery.type=single-node ulimits: memlock: soft: -1 hard: -1 nofile: soft: 65536 hard: 65536 cap_add: - IPC_LOCK volumes: - elasticsearch-data:/usr/share/elasticsearch/data ports: - 9200:9200 - 9300:9300 kibana: container_name: kibana image: docker.elastic.co/kibana/kibana:7.16.2 environment: - ELASTICSEARCH_HOSTS=http://elasticsearch:9200 ports: - 5601:5601 depends_on: - elasticsearch volumes: elasticsearch-data: driver: local EOF docker-compose -f docker-compose.yaml up -d
```

html 렌더링 예제

html 직접 전송 예제

app.py

```
from flask import Flask app = Flask(__name__) @app.route("/") def hello(): return "<h1>Hello World!</h1>" if __name__== "__main__": app.run(host='0.0.0.0', port=5000, debug=True)
```

uri를 활용한 변수 전달

app.py

```
from flask import Flask app = Flask(__name__) @app.route("/hello") def hello(): return "<h1>Hello World! 2</h1>" @app.route("/profile/<username>") def profile(username): return "<h1>Profile Page</h1>" + username if __name__== "__main__": app.run(host='0.0.0.0', port=5000, debug=True)
```

템플릿을 활용한 렌더링 예제

html form을 작성하고 이 form을 flask로 전달

templates/form.html

```
<form> {{var}} <p>이름: <input type="text" id="input"></p> <p>이름 입력 후 제출버튼을 누르세요. <input type="button" value="제출" onclick="alert('입력')"/> </p> </form>
```

app.py

```
from flask import Flask, render_template app = Flask(__name__) @app.route("/form") def form(): return render_template('form.html') # 추가 변수를 전달하는 경우 페이지 렌더링 @app.route("/form/<var>") def form_var(var): return render_template('form.html', var=var) if __name__ == "__main__": app.run(host='0.0.0.0', port=80, debug=True)
```

정적 파일 참조

source/static/img/pets-3715733_960_720.png

https://cdn.pixabay.com/photo/2018/10/01/09/21/pets-3715733_960_720.jpg

source/static/css/file.css

```
h1 { color-scheme: light; font-family: -apple-system,BlinkMacSystemFont,"Malgun Gothic","맑은 고딕",helvetica,"Apple SD Gothic Neo",sans-serif; list-style: none; font-size: 15px; line-height: 30px; font-weight: 700; letter-spacing: -.3px; display: block; text-decoration: none; color: #03c75a; }
```

source/templates/form.html

```
<html> <head> <link rel='stylesheet' href="{{url_for('static', filename='css/file.css')}}"/> </head> <body> <form> <h1> NAVER </h1> <img src="{{url_for('static', filename='img/pets-3715733_960_720.png')}}"/> {{var}} <p>이름: <input type="text" id="input"></p> <p>이름 입력 후 제출버튼을 누르세요. <input type="button" value="제출" onclick="alert('입력')"/> </p> </form> </body> </html>
```

Flask RestAPI 예제

https://restfulapi.net/http-methods/

flask rest api 라이브러리 설치

```
pip install flask_restx
```

자동차 정보 관리 파이썬 rest API 예제

```
from flask import Flask, request, Response from flask_restx import Resource, A
pi, fields from flask import abort, jsonify app = Flask(__name__) api = Api(ap
p) ns_cars = api.namespace('ns_cars', description='Car APIs') car_data = api.m
odel( 'Car Data', { "name": fields.String(description="model name", required=T
rue), "price": fields.Integer(description="car price", required=True), "fuel_t
ype": fields.String(description="fuel type", required=True), "fuel_efficiency"
: fields.String(description="fuel efficiency", required=True), "engine_power":
fields.String(description="engine power", required=True), "engine_cylinder": f
ields.String(description="engine cylinder", required=True) } ) car_info = {} n
umber_of_vehicles = 0 @ns_cars.route('/cars') class cars(Resource): def get(se
lf): return { 'number_of_vehicles': number_of_vehicles, 'car_info': car_info }
@ns_cars.route('/cars/<string:brand>') class cars_brand(Resource): # 브랜드 정보
조회 def get(self, brand): if not brand in car_info.keys(): abort(404, descript
ion=f"Brand {brand} doesn't exist") data = car_info[brand] return { 'number_of
_vehicles': len(data.keys()), 'data': data } # 새로운 브랜드 생성 def post(self, b
rand): if brand in car_info.keys(): abort(409, description=f"Brand {brand} alr
eady exists") car_info[brand] = dict() return Response(status=201) # 브랜드 정보
삭제 def delete(self, brand): if not brand in car_info.keys(): abort(404, descr
iption=f"Brand {brand} doesn't exists") del car_info[brand] return Response(st
atus=200) # 브랜드 이름 변경 def put(self, brand): # todo return Response(status=2
00) @ns_cars.route('/cars/<string:brand>/<int:model_id>') class cars_brand_mod
el(Resource): def get(self, brand, model_id): if not brand in car_info.keys():
abort(404, description=f"Brand {brand} doesn't exists") if not model_id in car
_info[brand].keys(): abort(404, description=f"Car ID {brand}/{model_id} does
n't exists") return { 'model_id': model_id, 'data': car_info[brand][model_id]
} @api.expect(car_data) # body def post(self, brand, model_id): if not brand i
n car_info.keys(): abort(404, description=f"Brand {brand} doesn't exists") if
model_id in car_info[brand].keys(): abort(409, description=f"Car ID {brand}/{m
odel_id} already exists") params = request.get_json() # get body json car_info
[brand][model_id] = params global number_of_vehicles number_of_vehicles += 1 r
eturn Response(status=200) def delete(self, brand, model_id): if not brand in
car_info.keys(): abort(404, description=f"Brand {brand} doesn't exists") if no
t model_id in car_info[brand].keys(): abort(404, description=f"Car ID {brand}/
{model_id} doesn't exists") del car_info[brand][model_id] global number_of_veh
icles number_of_vehicles -= 1 return Response(status=200) @api.expect(car_data
) def put(self, brand, model_id): global car_info if not brand in car_info.key
s(): abort(404, description=f"Brand {brand} doesn't exists") if not model_id i
n car_info[brand].keys(): abort(404, description=f"Car ID {brand}/{model_id} d
oesn't exists") params = request.get_json() car_info[brand][model_id] = params
return Response(status=200) if __name__ == "__main__": app.run(debug=True, hos
t='0.0.0.0', port=5000)
```

샘플 restapi 서비스 애플리케이션 실행

```
python -i source/app.py
```

curl 명령으로 테스트 해보기

cars 전체 조회 → 비어있음

```
curl -X 'GET' \ 'http://192.168.100.132/ns_cars/cars' \ -H 'accept: application/json'
```

자동차 브랜드를 생성하고 조회

cars/<string:brand>

```
curl -X 'POST' \ 'http://192.168.100.132/ns_cars/cars/bentz' \ -H 'accept: application/json' \ -d '' curl -X 'GET' \ 'http://192.168.100.132/ns_cars/cars' \ -H 'accept: application/json'
```

자동차 브랜드에 특정 모델 정보 입력

cars/<string:brand>/<int:model_id>

```
curl -X 'POST' \ 'http://192.168.100.132/ns_cars/cars/bentz/0' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "name": "e-class", "price": 1000000, "fuel_type": "gasoline", "fuel_efficiency": "9.1~13.2km/l", "engine_power": "367hp", "engine_cylinder": "I6" }'
```

입력된 정보 조회

```
# 특정 모델 조회 curl -X 'GET' \ 'http://192.168.100.132/ns_cars/cars/bentz/0' \ -H 'accept: application/json' # 전체 모델 조회 curl -X 'GET' \ 'http://192.168.100.132/ns_cars/cars' \ -H 'accept: application/json'
```

브랜드 데이터 삭제

```
curl -X 'DELETE' \ 'http://192.168.100.132/ns_cars/cars/bentz' \ -H 'accept: application/json'
```

🚲 (참고 자료)파이썬으로 엘라스틱서치 다루기