

HTTP 통신

컴퓨터네트워크 2주차

공과대학 5호관 633호
데이터 네트워크 연구실

조교 : 황동준

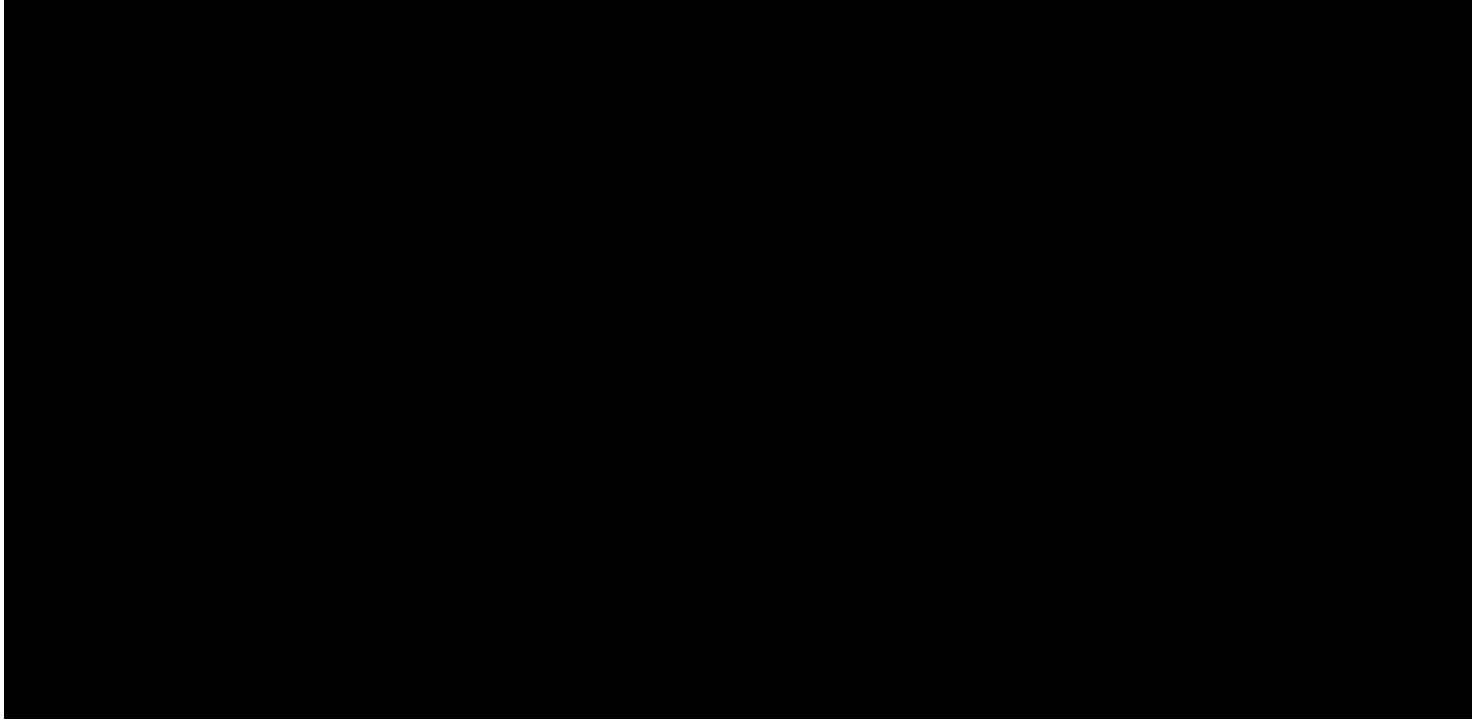
실습

1. Python을 통해 http.server 제작을 통해 HTTP 통신에 대한 학습
 - 네트워크 분석 프로그램인 Wireshark를 사용한 패킷 분석

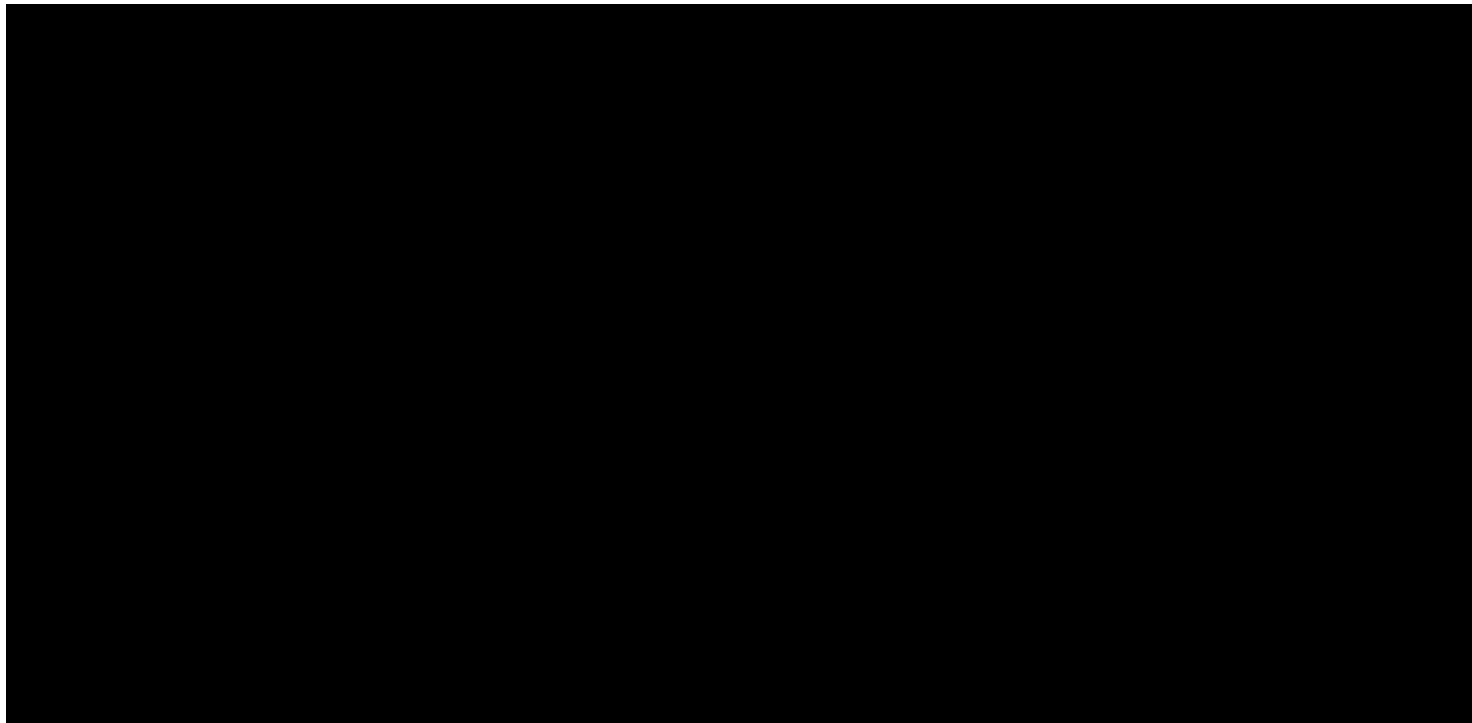
과제

1. Python의 웹 어플리케이션 제작 모듈인 Flask를 사용하여 GET / POST 요청 보내기 (10점)
 - GET 요청 보내기 (2점)
 - POST 요청 보내기 (6점)
 - Dockerizing (2점)

결과 영상



결과 영상 - Dockerizing



실습 : Python을 통해 http.server 제작

HTTP 통신

- Client의 요청이 있을 때만 Server가 응답하여 처리한 후에 연결을 끊는 방식
 - 단방향적 통신이 이루어짐



- 실시간 연결이 아니며, 필요한 경우에만 Server로 요청을 보내는 상황에 유용

Python → http.server

```
1 from http.server import BaseHTTPRequestHandler, HTTPServer
2
3 class HandleRequests(BaseHTTPRequestHandler):
4     def _set_headers(self):
5         self.send_response(200)
6         self.send_header('Content-type', 'text/html')
7         self.end_headers()
8
9     def do_GET(self):
10        self._set_headers()
11        self.wfile.write('received get request'.encode('utf-8'))
12
13    def do_POST(self):
14        self._set_headers()
15        content_len = int(self.headers['Content-Length'])
16        post_body = self.rfile.read(content_len)
17        self.wfile.write("received post request:<br>{}".format(post_body).encode('utf-8'))
18
19 host = ''
20 port = 80
21 HTTPServer((host, port), HandleRequests).serve_forever()
```

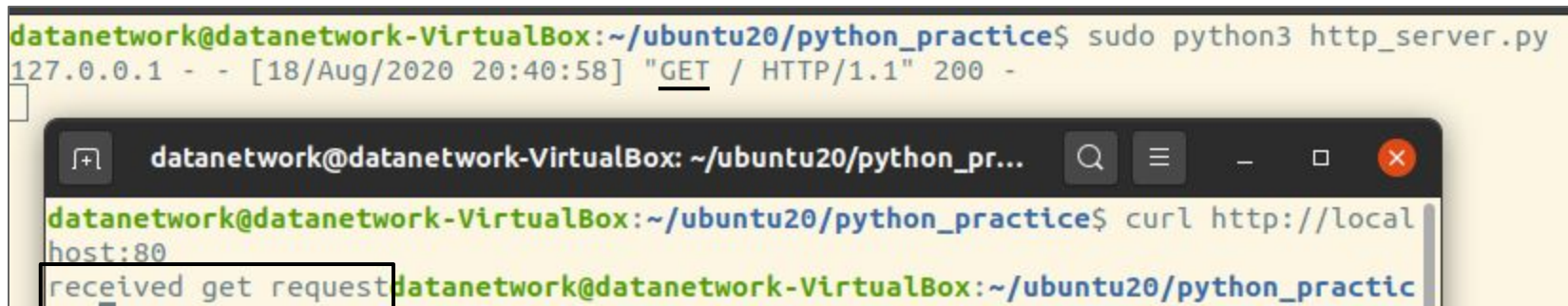
클라이언트와 서버가 요청 또는
응답으로 부가적인 정보를
전송할 수 있도록 도와줌

GET Message

- 지정된 리소스(URI)를 요청

curl : 리눅스에서 http 메시지를 shell상에서 요청하여 결과를 확인하는 명령어

- 8 슬라이드에서 제작한 http.server파일을 [python3 파일이름.py] 으로 실행
 - curl <http://localhost:80> 명령어를 통해 GET을 통해 메시지 획득



The screenshot shows a terminal window with the following content:

```
datanetwork@datanetwork-VirtualBox:~/ubuntu20/python_practice$ sudo python3 http_server.py
127.0.0.1 - - [18/Aug/2020 20:40:58] "GET / HTTP/1.1" 200 -
```

Below the terminal window is a smaller window showing the command prompt and the output of the curl command:

```
datanetwork@datanetwork-VirtualBox: ~/ubuntu20/python_pr...
datanetwork@datanetwork-VirtualBox:~/ubuntu20/python_practice$ curl http://localhost:80
received get request
```

- curl이 설치되어있지 않을 경우 ⇒ `sudo apt install -y curl` 로 설치

GET Message → Wireshark

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	35244 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 S...
2	0.000000061	127.0.0.1	127.0.0.1	TCP	76	80 → 35244 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 ...
3	0.000016237	127.0.0.1	127.0.0.1	TCP	68	35244 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval...
4	0.000050666	127.0.0.1	127.0.0.1	HTTP	141	GET / HTTP/1.1
5	0.000053410	127.0.0.1	127.0.0.1	TCP	68	80 → 35244 [ACK] Seq=1 Ack=74 Win=65536 Len=0 TSva...
6	0.0000496775	127.0.0.1	127.0.0.1	TCP	184	80 → 35244 [PSH, ACK] Seq=1 Ack=74 Win=65536 Len=1...
7	0.0000501240	127.0.0.1	127.0.0.1	TCP	68	35244 → 80 [ACK] Seq=74 Ack=117 Win=65536 Len=0 TS...
8	0.0000521720	127.0.0.1	127.0.0.1	TCP	88	80 → 35244 [PSH, ACK] Seq=117 Ack=74 Win=65536 Len...
9	0.0000524506	127.0.0.1	127.0.0.1	TCP	68	35244 → 80 [ACK] Seq=74 Ack=137 Win=65536 Len=0 TS...
10	0.0000546363	127.0.0.1	127.0.0.1	HTTP	68	HTTP/1.0 200 OK (text/html)
11	0.0000701445	127.0.0.1	127.0.0.1	TCP	68	35244 → 80 [FIN, ACK] Seq=74 Ack=138 Win=65536 Len...
12	0.0000706663	127.0.0.1	127.0.0.1	TCP	68	80 → 35244 [ACK] Seq=138 Ack=75 Win=65536 Len=0 TS...
13	2.148129089	10.0.2.15	210.94.0.73	DNS	102	Standard query 0x507c AAAA connectivity-check.ubun...

Frame 4: 141 bytes on wire (1128 bits), 141 bytes captured (1128 bits) on interface any, id 0

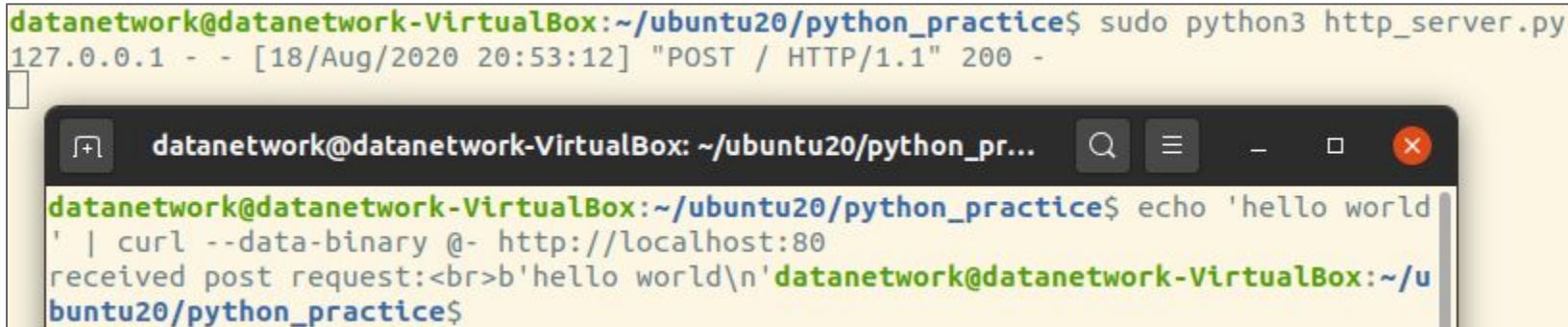
- Linux cooked capture
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 35244, Dst Port: 80, Seq: 1, Ack: 1, Len: 73
- Hypertext Transfer Protocol
 - GET / HTTP/1.1\r\n
 - Host: localhost\r\n
 - User-Agent: curl/7.68.0\r\n
 - Accept: */*\r\n
 - \r\n
 - [Full request URI: <http://localhost/>]
 - [HTTP request 1/1]
 - [Response in frame: 10]

0000 00 00 03 04 00 06 00 00 00 00 00 00 00 08 00
0010 45 00 00 7d b5 7c 40 00 40 06 86 fc 7f 00 00 01 E...|@. @.....
0020 7f 00 00 01 89 ac 00 50 a6 90 43 33 fb 0a 6d edP...C3..m.
0030 80 18 02 00 fe 71 00 00 01 01 08 0a b5 a1 b3 d3q.....
0040 b5 a1 b3 d3 47 45 54 20 2f 20 48 54 54 50 2f 31GET / HTTP/1
0050 2e 31 0d 0a 48 6f 73 74 3a 20 6c 6f 63 61 6c 68 .1..Host : localh
0060 6f 73 74 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a ost..Use r-Agent:
0070 20 63 75 72 6c 2f 37 2e 36 38 2e 30 0d 0a 41 63 curl/7. 68.0..Ac
0080 63 65 70 74 3a 20 2a 2f 2a 0d 0a 0d 0a cept: */ *....

- Wireshark에서 GET Message를 확인 (요청, 응답)

POST Message

- 서버가 클라이언트의 폼 입력 필드 데이터의 수락을 요청. 클라이언트는 서버로 HTTP Body에 Data를 전송
- 8 슬라이드에서 제작한 http.server파일을 [python3 파일이름.py] 으로 실행
 - echo 'hello world' | curl --data-binary @- <http://localhost:80> 명령어를 통해 body에 있는 data를 전송



```
datanetwork@datanetwork-VirtualBox:~/ubuntu20/python_practice$ sudo python3 http_server.py
127.0.0.1 - - [18/Aug/2020 20:53:12] "POST / HTTP/1.1" 200 -

datanetwork@datanetwork-VirtualBox:~/ubuntu20/python_practice$ echo 'hello world' | curl --data-binary @- http://localhost:80
received post request:<br>b'hello world\n'datanetwork@datanetwork-VirtualBox:~/ubuntu20/python_practice$
```

POST Message → Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
68	553.605358975	127.0.0.1	127.0.0.1	TCP	76	80 → 35250 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 ...
69	553.605386397	127.0.0.1	127.0.0.1	TCP	68	35250 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=...
70	553.605708913	127.0.0.1	127.0.0.1	HTTP	223	POST / HTTP/1.1 (application/x-www-form-urlencoded...)
71	553.605716241	127.0.0.1	127.0.0.1	TCP	68	80 → 35250 [ACK] Seq=1 Ack=156 Win=65408 Len=0 TSv=...
72	553.607320544	127.0.0.1	127.0.0.1	TCP	184	80 → 35250 [PSH, ACK] Seq=1 Ack=156 Win=65536 Len=...
73	553.607349569	127.0.0.1	127.0.0.1	TCP	68	35250 → 80 [ACK] Seq=156 Ack=117 Win=65536 Len=0 T=...
74	553.607456795	127.0.0.1	127.0.0.1	TCP	110	80 → 35250 [PSH, ACK] Seq=117 Ack=156 Win=65536 Le=...
75	553.607466489	127.0.0.1	127.0.0.1	TCP	68	35250 → 80 [ACK] Seq=156 Ack=159 Win=65536 Len=0 T=...
76	553.607655613	127.0.0.1	127.0.0.1	HTTP	68	HTTP/1.0 200 OK (text/html)
77	553.607779088	127.0.0.1	127.0.0.1	TCP	68	35250 → 80 [FIN, ACK] Seq=156 Ack=160 Win=65536 Le=...
78	553.607795352	127.0.0.1	127.0.0.1	TCP	68	80 → 35250 [ACK] Seq=160 Ack=157 Win=65536 Len=0 T=...
79	602.126611385	10.0.2.15	210.94.0.73	DNS	102	Standard query 0x9f47 AAAA connectivity-check.ubun...
80	602.134670951	210.94.0.73	10.0.2.15	DNS	163	Standard query response 0x9f47 AAAA connectivity-c...
Host: localhost\r\n						
User-Agent: curl/7.68.0\r\n						
Accept: */*\r\n						
Content-Length: 12\r\n						
Content-Type: application/x-www-form-urlencoded\r\n						
\r\n						
[Full request URI: http://localhost/]						
[HTTP request 1/1]						
[Response in frame: 76]						
File Data: 12 bytes						
HTML Form URL Encoded: application/x-www-form-urlencoded						
Form item: "hello world"						
" = ""						
Key: hello world\r\n						
Value:						
0050	31 2e 31 0d 0a 48 6f 73	74 3a 20 6c 6f 63 61 6c	1.1..Host: local			
0060	68 6f 73 74 0d 0a 55 73	65 72 2d 41 67 65 6e 74	host..User-Agent			
0070	3a 20 63 75 72 6c 2f 37	2e 36 38 2e 30 0d 0a 41	: curl/7.68.0..A			
0080	63 63 65 70 74 3a 20 2a	2f 2a 0d 0a 43 6f 6e 74	ccept: */*..Cont			
0090	65 6e 74 2d 4c 65 6e 67	74 68 3a 20 31 32 0d 0a	ent-Leng th: 12..			
00a0	43 6f 6e 74 65 6e 74 2d	54 79 70 65 3a 20 61 70	Content- Type: ap			
00b0	70 6c 69 63 61 74 69 6f	6e 2f 78 2d 77 77 77 2d	plicatio n/x-www-			
00c0	66 6f 72 6d 2d 75 72 6c	65 6e 63 6f 64 65 64 0d	form- url encoded			
00d0	0a 0d 0a 68 65 6c 6c 6f	20 77 6f 72 6c 64 0a	..hello world..			

- Wireshark에서 POST Message를 확인 (Body의 Data 확인 가능)

과제1 : Python Flask를 사용하여 GET / POST 요청

Flask



Flask

web development,
one drop at a time

- 파이썬에서 웹 어플리케이션을 만드는 마이크로 웹 프레임워크
- python, HTML + CSS + JavaScript를 통해 제작이 가능하며, 매우 간단하게 배우기가 가능
- 우리는 웹 프로그래밍보다는 웹 서버를 만들어 보는 것에 초점을 둠

Flask 서버 제작



Flask

web development,
one drop at a time

- Flask 서버 제작

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def index():
6     return '<h1>Hello World!</h1>'
7
8 if __name__ == "__main__":
9     app.run(debug=True, port=5000)
```

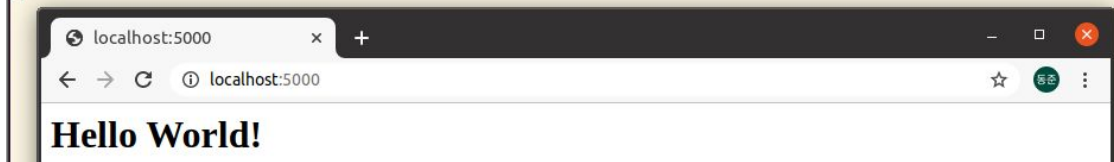
Flask 인스턴스 생성

- 웹 표현 : route() 함수 사용
- 맨 앞에 @가 붙는다면 장식자(decorator)를 나타냄
 - 이러한 장식자가 URL 연결에 사용됨
 - GET, POST method를 설정 가능
 - 현 상태가 GET
 - POST 힌트 : 장식자의 methods 옵션

Flask 서버 실행 결과

- 제작한 flask 파일을 실행
 - python3 파일이름.py
- 브라우저를 접속하여 “localhost:5000”으로 접속
 - Hello World! 문구를 확인

```
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/3week$ python3 test.py
* Serving Flask app "test" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 641-418-293
127.0.0.1 - - [20/Aug/2020 00:04:38] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Aug/2020 00:04:38] "GET /favicon.ico HTTP/1.1" 404 -
```



Flask template - 1

- html파일로 template 파일을 제작하여 렌더링하는 과정 진행

```
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/basic_flask$ ls
app.py  templates  test.py
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/basic_flask$ cd templates/
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/basic_flask/templates$ ls
get.html  post.html
```

- /basic_flask/ 디렉토리의 templates 디렉토리에 집중할 것! (이름 동일하게 지킬것!)
- 다음 슬라이드의 코드는 실제 과제 코드인데, GET 방식은 공개하며, POST 방식을 구현해야 함

Flask template - 2

- Flask의 render_template에 의해서 template 디렉토리의 존재하는 html파일을 인식
 - render_template함수를 통해 렌더링을 진행할 수 있음

```
1 from flask import Flask, render_template, request
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello_world():
7     return 'Hello World!'
8
9 @app.route('/test_get')
10 def test_get():
11     return render_template('get.html')
12
13 @app.route('/test_post')
14     POST method
15
16
17 @app.route('/get', methods=['GET'])
18 def get():
19     value = request.args.get('test')
20     return value
21
22 @app.route('/post', methods=['POST'])
23     POST method
24
25
26
27 if __name__ == '__main__':
28     app.run(debug=True, host='0.0.0.0')
```

Flask template - 3

- get.html과 post.html을 미리 작성해두고, 렌더링을 진행하는 방식
 - get.html, post.html파일은 그대로 사용해도 무방

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Get</title>
6 </head>
7 <body>
8     <form action="/get" method="get">
9         <p>테스트 : <input type="text" name="test"></p>
10        <input type="submit" value="Send Info">
11    </form>
12 </body>
13 </html>
```


get.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Post</title>
6 </head>
7 <body>
8     <form action="/post" method="post">
9         <p>테스트 : <input type="text" name="test"></p>
10        <input type="submit" value="Send Info">
11    </form>
12 </body>
13 </html>
```

post.html

결과 - GET methods

```
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/basic_flask$ python3 app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [25/Aug/2020 14:08:51] "GET /test_get HTTP/1.1" 200 -
127.0.0.1 - - [25/Aug/2020 14:08:56] "GET /get?test=hello HTTP/1.1" 200 -
```

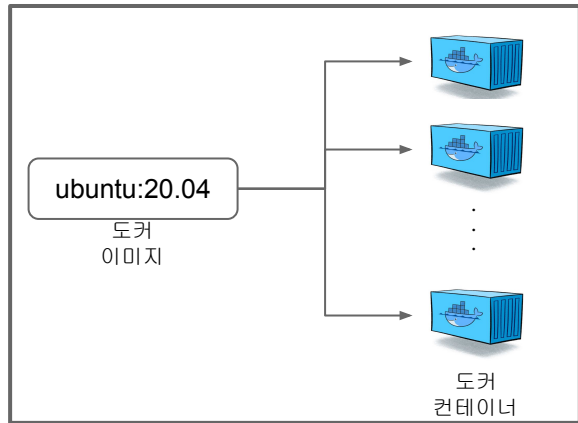
A screenshot of a web browser window. The address bar shows the URL 'localhost:5000/get?test=hello' with a red box highlighting the entire address bar area. Below the address bar, the page content displays the word 'hello'.

- flask파일을 실행(python3 이름.py) → localhost:5000/test_get 으로 접속 → 문자열 입력후 Send Info 버튼 클릭 → URL의 변화 확인

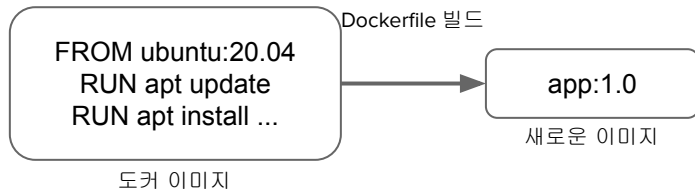
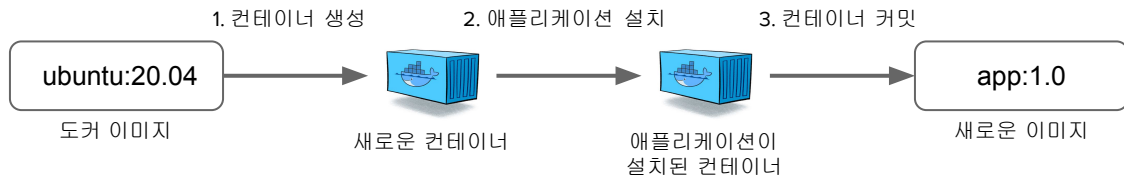
과제2 : Dockerizing

Docker Container를 다뤄보자.

- Docker Hub에서 이미지를 다운받을 수 있음
 - a. 이러한 이미지로 컨테이너를 생성하면 해당 이미지의 목적에 맞는 파일이 들어있는 격리된 시스템 자원 및 네트워크를 사용할 수 있음
 - i. 이것이 컨테이너!
 - b. 컨테이너는 이미지를 읽기 전용으로 사용하되 이미지에서 변경된 사항만 컨테이너 계층에 저장하므로 컨테이너에서 무엇을 하든 원래의 이미지에는 영향을 주지 않음
- Container를 다뤄보자.
 - a. `docker run -it ubuntu:20.04`
 - b. `docker pull ubuntu:20.04` → `docker create -it --name hello ubuntu:20.04` → `docker start [CONTAINER ID]` → `docker attach [CONTAINER ID]`
 - c. `docker rename [PREV_CONTAINER_NAME] [NEW_CONTAINER_NAME]`
 - d. `docker rm` / `docker -f rm`
 - e. `docker start` / `docker stop`
 - f. `docker container prune`
 - g. `docker stop $(docker ps -a -q)` / `docker rm $(docker ps -a -q)`

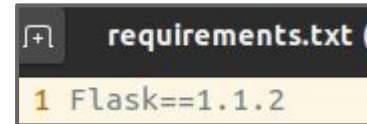


Container로 Image를 만들어보자.



- Docker는 일련의 과정을 손쉽게 기록하고 수행할 수 있는 빌드(build) 명령어를 제공
 - a. 완성된 이미지를 생성하기 위해 컨테이너에 설치해야 하는 패키지, 추가해야 하는 소스코드, 실행해야 하는 명령어와 셸 스크립트 등을 하나의 파일에 기록해 두는 것

Dockerizing - Dockerfile



```
requirements.txt  
1 Flask==1.1.2
```

- Dockerfile 생성 → \$ vi Dockerfile

```
1 FROM ubuntu:20.04  
2  
3 MAINTAINER Dongjun "enter031@cs-cnu.org"  
4  
5 RUN apt update -y && \  
6     apt install -y python3-pip python3-dev  
7  
8 COPY ./requirements.txt /app/requirements.txt  
9  
10 WORKDIR /app  
11  
12 RUN pip3 install -r requirements.txt  
13  
14 COPY . /app  
15  
16 ENTRYPOINT [ "python3" ]  
17  
18 CMD [ "app.py" ]
```

→ ubuntu 20.04 도커 이미지 다운

→ 이미지 작성자 정보 입력 → 이름 “이메일”

→ 도커 이미지 다운 후, 내부 명령어 실행 명령

→ 외부에서 컨테이너 내부로 파일 복사

→ 작업 디렉토리 설정 (아래로는 모두 해당 디렉토리에서 작업이 이루어짐)

→ 실행 파일로 실행되도록 설정

→ 내부 명령어 실행 (여기서는 파일 이름)

Dockerizing build & run

- Dockerfile build
 - a. `docker build -t flask-tutorial:latest .`
 - i. `-t`는 tag 옵션으로 flask-tutorial은 도커 이미지 이름 / `:latest`는 버전을 나타냄
 - b. 실행 중 permission 오류 발생 시 `sudo` 를 가장 앞에 붙여 실행할 것
- 빌드 완료 후, 컨테이너 실행
 - a. `docker run -d -p 5000:5000 flask-tutorial`
 - b. 브라우저 접속하여 “localhost:5000/” 접속

Dockerizing 결과 화면

```
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/basic_flask$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
2week_practice       latest             f3f5ba9deb90       4 minutes ago      394MB
ubuntu               20.04             4e2eef94cd6b       6 days ago         73.9MB
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/basic_flask$ sudo docker run -d -p 5000:5000 2week_practice
ee09bbcc7db006b057fd848411666594932a9a37c3bc39e9f9c12db95760e112
datanetwork@datanetwork-VirtualBox:~/ubuntu20/computer_networking/basic_flask$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
ee09bbcc7db0       2week_practice     "python3 app.py"    3 seconds ago      Up 2 seconds       0.0.0.0:5000->5000/tcp   wonderful_la
mport
```

- Dockerizing 후, 도커 이미지를 빌드하여 run명령어를 통해 이미지 실행



- 결과는 Docker를 사용하지 않았을 때의 결과와 동일

과제 제출

- 과제 제출 기한
 - 2021년 9월 23일 17:59 까지 사이버 캠퍼스에 제출
- 제출 파일 (.zip)
 - GET / POST 요청 보내는 .py 파일 (양식 : 학번_app.py)
 - Dockerizing 결과 캡처 (보고서에 작성)
- 딜레이 1일당 20%감점
 - ex) 10점 만점 기준, 1일 딜레이인 경우 8점
- 과제 카피 적발시 보여준사람 본사람 모두 0점처리
 - 카피 적발기준 : 과제 유사도 80%이상

유의사항

- 파일명 : **CN02_02_학번_이름.zip**
 - python파일과 보고서를 함께 압축하여 제출
 - 보고서 : **PDF로 제출할것 (파일명 : 02_학번_이름.pdf)**
 - 과제 목표 (도출해야할 결과)
 - 코드 설명과 과제 해결 방법 (실행 결과 사진도 보고서에 첨부, 코드 및 설명 필요)
 - Dockerizing 완료 후, 결과 사진 첨부
 - 과제 느낀점 및 하고싶은 말 (**선택사항**)
 - 형식 지켜지지 않을시 채점대상에서 제외 (보고서도 HWP, DOC은 채점대상에서 제외)
- 질문
 - 질문은 디스코드에서 받음

부록

HTTP Protocol

- HTML 문서와 같은 리소스들을 가져올 수 있도록 해주는 프로토콜
 - Hypertext Transfer Protocol
- 웹 상에서 데이터를 주고 받기위해 사용함
- Application layer (응용 계층) 위의 프로토콜
- POST / GET Message로 동작

Dockerize란

- Dockerfile을 제작하여 Docker에 올림
 - 자신이 Docker Image를 제작하는 과정
 - Dockerfile : Docker Image를 만들기 위한 설정 파일