

# IoT Devices & Cloud Platforms

KOOK Joongjin, Ph.D. Assistant Professor (kook@smu.ac.kr)  
Dept. of Information Security Engineering  
Sangmyung University

# REST



# REST

## ▶ REST의 정의

- ▶ REST(Representational State Transfer)는 월드 와이드 웹과 같은 분산 하이퍼 미디어 시스템을 위한 소프트웨어 아키텍처의 한 형식 [wikipedia]
- ▶ 자원을 이름(자원의 표현)으로 구분하여 해당 자원의 상태(정보)를 주고 받는 모든 것을 의미
- ▶ URI는 자원을 표현하는 데 집중하고, 행위에 대한 정의는 HTTP Method를 통해 표현하는 것이 REST API를 설계하는 중심 규칙

```
GET    /devices
GET    /users
GET    /device/actions
```

*Resource 이름은 명사를 사용하여 직관적으로 표현*

```
POST    /device
DELETE  /device
```

*행위는 HTTP Method로 표현  
-GET, POST, PUT, DELETE 등*

# REST

## ▶ HTTP Methods

- ↘ 주로 5가지의 Method(GET, POST, PUT, PATCH, DELETE)를 사용하여 CRUD(Create, Retrieve, Update, Delete)를 구현

Method	Action	역할
GET	index/retrieve	모든/특정 리소스를 조회
POST	create	리소스를 생성
PUT	update all	리소스의 전체를 갱신
PATCH	update	리소스의 일부를 갱신
DELETE	delete	리소스를 삭제

# REST

## ▶ REST APIs

- ↘ 자원 (Resource), 행위 (Verb), 표현 (Representations)의 3가지 요소로 구성
- ↘ REST는 자기서술적(Self-descriptiveness) 메시지로 구성되어 REST API만으로 요청의 이해 가능

구성 요소	내용	표현 방법
Resource	자원	HTTP URI
Verb	자원에 대한 행위	HTTP Method
Representations	내용	HTTP Message Payload

# REST API Test

## ▶ json-server

↘ 사용자 홈 디렉터리로 이동하여 'rest-api-exam' 디렉터리 추가

>*cd \users\user*

>*mkdir rest-api-exam*

```
1
2 C:\Users\kook>mkdir rest-api-exam
3
4 C:\Users\kook>cd rest-api-exam
5
```

# REST API Test

## ▶ json-server

↘ json-server 설치

> *npm init -y*

> *npm install json-server*

```
6 C:\Users\kook\rest-api-exam>npm init -y
7
8 C:\Users\kook\rest-api-exam>npm install json-server
9
10 C:\Users\kook\rest-api-exam>dir
11 C 드라이브의 볼륨에는 이름이 없습니다.
12 볼륨 일련 번호: 500D-3030
13
14 C:\Users\kook\rest-api-exam 디렉터리
15
16 2019-06-12 오후 11:16 <DIR> .
17 2019-06-12 오후 11:16 <DIR> ..
18 2019-06-12 오후 11:16 <DIR> node_modules
19 2019-06-12 오후 11:16 71,133 package-lock.json
20 2019-06-12 오후 11:16 281 package.json
21 2개 파일 71,414 바이트
22 3개 디렉터리 43,529,363,456 바이트 남음
23
```

# REST API Test

- ▶ API 호출을 위한 json 파일 생성 (devices.json)

```
{  
  "devices": [  
    { "id": 1, "content": "doorlock", "state": false },  
    { "id": 2, "content": "fan", "state": true },  
    { "id": 3, "content": "heater", "state": false },  
    { "id": 4, "content": "computer", "state": true }  
  ]  
}
```



# REST API Test

- ▶ json-server 실행을 위한 package.json 파일 수정

```
{
  "name": "rest-api-exam",
  "version": "1.0.0",
  "description": "",
  "scripts": {
    "start": "json-server --watch devices.json --port 8888"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "json-server": "^0.15.0"
  }
}
```

# REST API Test

## ▶ rest-api-exam 디렉터리 구성

```
2 C:\Users\kook\rest-api-exam>dir
3 C 드라이브의 볼륨에는 이름이 없습니다.
4 볼륨 일련 번호: 500D-3030
5
6 C:\Users\kook\rest-api-exam 디렉터리
7
8 2019-06-12 오후 11:49 <DIR> .
9 2019-06-12 오후 11:49 <DIR> ..
10 2019-06-12 오후 11:48      257 devices.json
11 2019-06-12 오후 11:16 <DIR> node_modules
12 2019-06-12 오후 11:16    71,133 package-lock.json
13 2019-06-12 오후 11:16     281 package-origin.json
14 2019-06-12 오후 11:43     256 package.json
15          4개 파일              71,927 바이트
16          3개 디렉터리  43,464,003,584 바이트 남음
17
18 C:\Users\kook\rest-api-exam>
```

# REST API Test

## ▶ json-server 실행

```
C:\Users\User\rest-api-exam>npm start
> rest-api-exam@1.0.0 start C:\Users\User\rest-api-exam
> json-server --watch devices.json --port 8888

⚡(^_^)/ hi!

Loading devices.json
Done

Resources
http://localhost:8888/devices

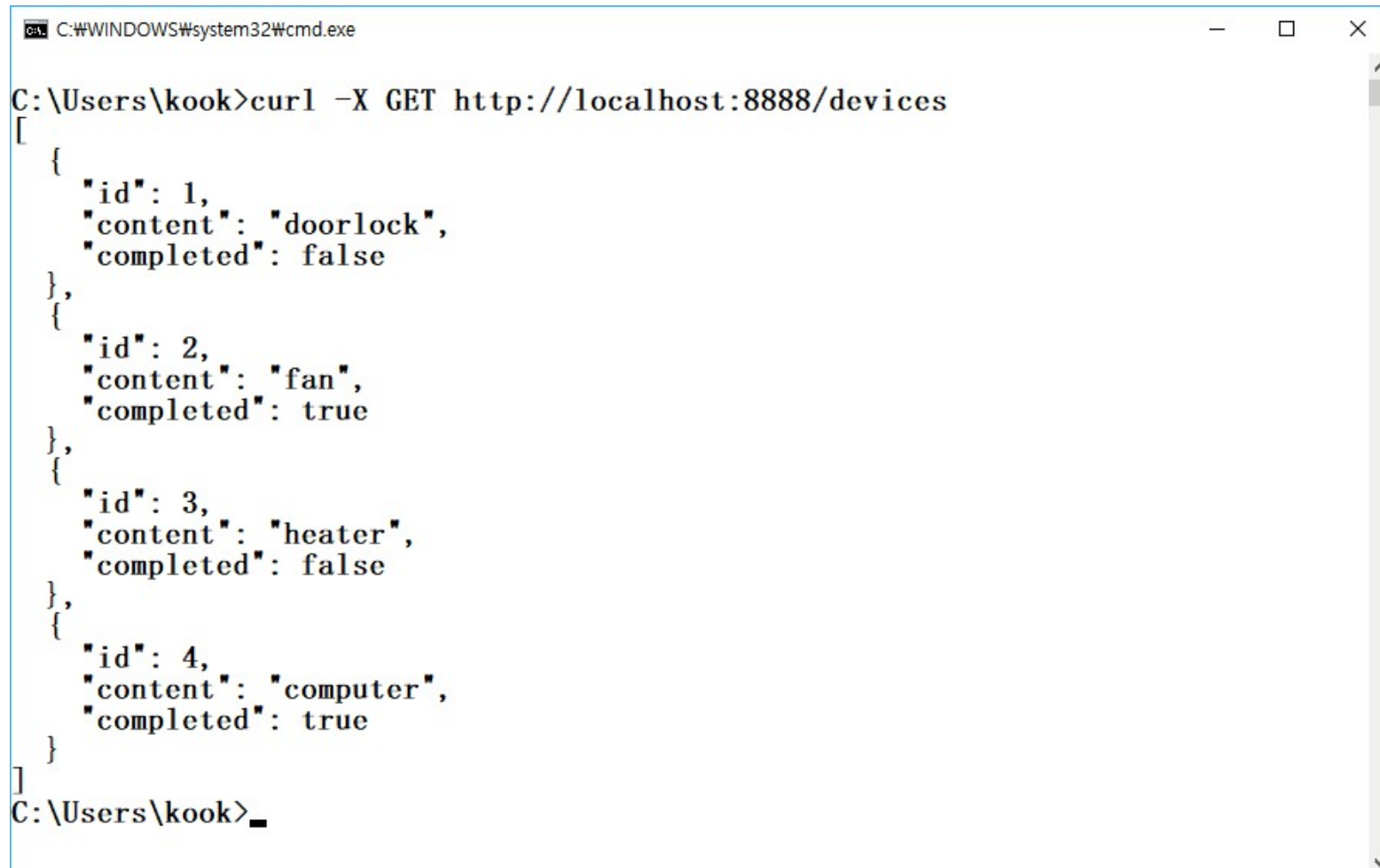
Home
http://localhost:8888

Type s + enter at any time to create a snapshot of the database
Watching...
```

# REST API Test

## ▶ REST API Test #1

↘ Method: GET



```
C:\WINDOWS\system32\cmd.exe

C:\Users\kook>curl -X GET http://localhost:8888/devices
[
  {
    "id": 1,
    "content": "doorlock",
    "completed": false
  },
  {
    "id": 2,
    "content": "fan",
    "completed": true
  },
  {
    "id": 3,
    "content": "heater",
    "completed": false
  },
  {
    "id": 4,
    "content": "computer",
    "completed": true
  }
]
C:\Users\kook>
```

# REST API Test

## ▶ REST API Test #2

↘ Method: GET/id

```
C:\WINDOWS\system32\cmd.exe

C:\Users\kook>curl -X GET http://localhost:8888/devices/1
{
  "id": 1,
  "content": "doorlock",
  "completed": false
}
C:\Users\kook>curl -X GET http://localhost:8888/devices/4
{
  "id": 4,
  "content": "computer",
  "completed": true
}
C:\Users\kook>
```

# REST API Test

## ▶ REST API Test #3

↘ Method: POST

```
C:\WINDOWS\system32\cmd.exe
C:\Users\kook>curl -X POST http://localhost:8888/devices -H "Content-type:application/json" -d '{"id":5, "content":"lamp", "state":"off"}'
{
  "id": 5,
  "content": "lamp",
  "state": "off"
}
C:\Users\kook>
```

# REST API Test

## ▶ REST API Test #3

➤ Method: POST

```
C:\WINDOWS\system32\cmd.exe
C:\Users\kook>curl -X POST http://localhost:8888/devices -H "Content-type:application/json" -d '{"id":5, "content":"lamp", "state":"off"}'
{
  "id": 5,
  "content": "lamp",
  "state": "off"
}
C:\Users\kook>
```

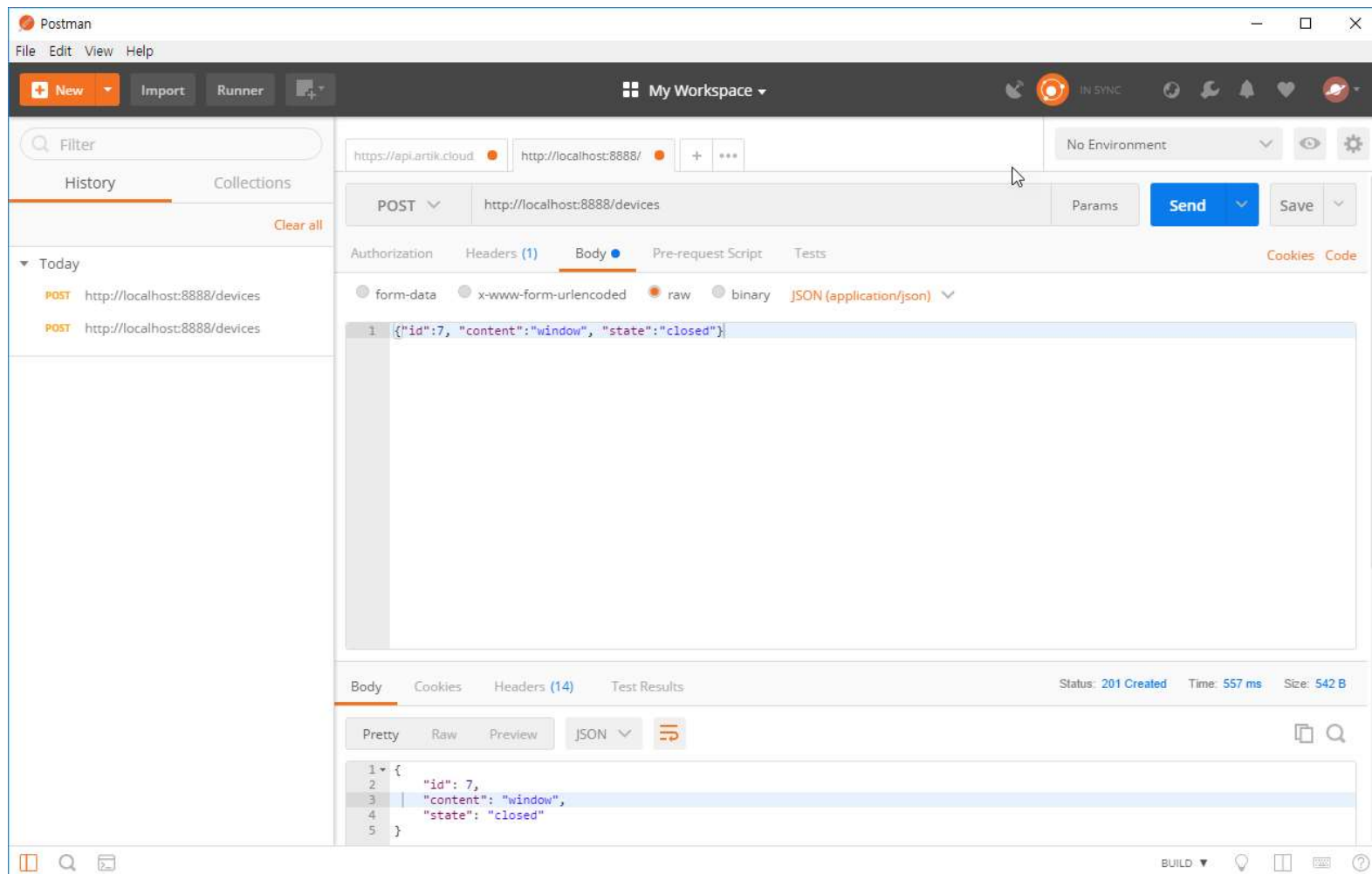
```
1 {
2   "devices": [
3     {
4       "id": 1,
5       "content": "doorlock",
6       "completed": false
7     },
8     {
9       "id": 2,
10      "content": "fan",
11      "completed": true
12    },
13    {
14      "id": 3,
15      "content": "heater",
16      "completed": false
17    },
18    {
19      "id": 4,
20      "content": "computer",
21      "completed": true
22    },
23    {
24      "id": 5,
25      "content": "lamp",
26      "state": "off"
27    }
28  ]
29 }
```

JSON file      length : 414    lines : 29      Ln : 23    Col : 1    Sel : 76 | 6      Windows (CR LF)    ANSI    INS

# REST API Test

## ▶ REST API Test #3

➤ Method: POST (using Postman)





# REST API Test

## ▶ REST API Test #3

➤ Method: POST (using Postman)

The screenshot shows the Postman application interface. A 'GENERATE CODE SNIPPETS' dialog box is open, displaying Python code for a POST request. The code is as follows:

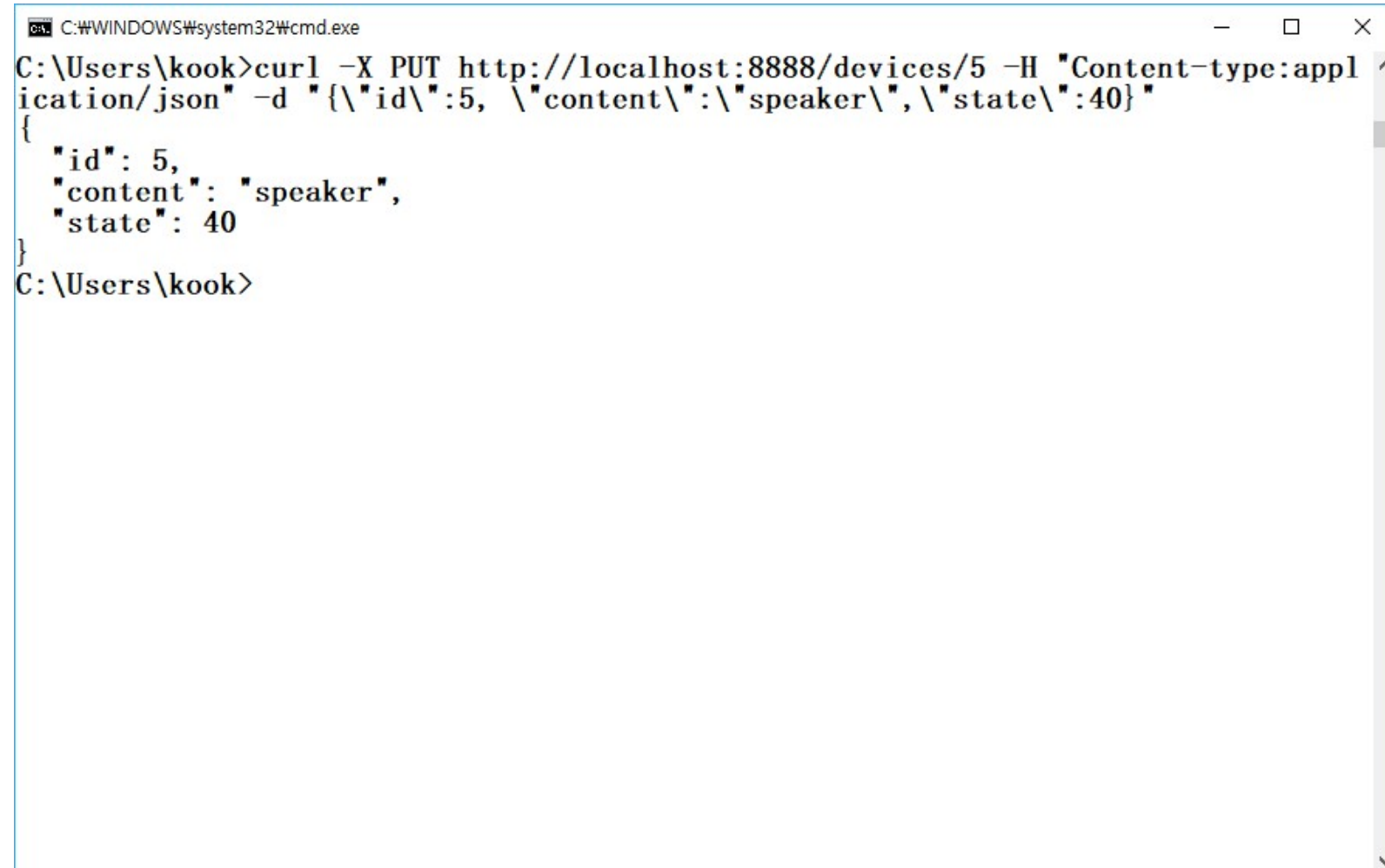
```
1 import requests
2
3 url = "http://localhost:8888/devices"
4
5 payload = {"id":7, "content":"window", "state":"closed"}
6 headers = {
7     'Content-Type': "application/json",
8     'Cache-Control': "no-cache",
9     'Postman-Token': "fc611814-8d8a-45aa-9788-b56a0b2adcb0"
10 }
11
12 response = requests.request("POST", url, data=payload, headers=headers)
13
14 print(response.text)
```

A red dashed circle highlights the 'Code' button in the bottom right corner of the dialog box. A yellow callout box with the text 'Click "Code"' points to this button. The background shows the Postman interface with a 'History' tab selected, displaying two POST requests to 'http://localhost:8888/devices'.

# REST API Test

## ▶ REST API Test #4

↘ Method: PUT



```
C:\WINDOWS\system32\cmd.exe
C:\Users\kook>curl -X PUT http://localhost:8888/devices/5 -H "Content-type:application/json" -d '{"id":5, "content":"speaker", "state":40}'
{"id": 5,
 "content": "speaker",
 "state": 40}
C:\Users\kook>
```

# REST API Test

## ▶ REST API Test #4

### ↘ Method: PUT

```
C:\WINDOWS\system32\cmd.exe
C:\Users\kook>curl -X PUT http://localhost:8888/devices/5 -H "Content-type: application/json" -d '{"id":5, "content":"speaker", "state":40}'
{"id": 5, "content": "speaker", "state": 40}
C:\Users\kook>
```

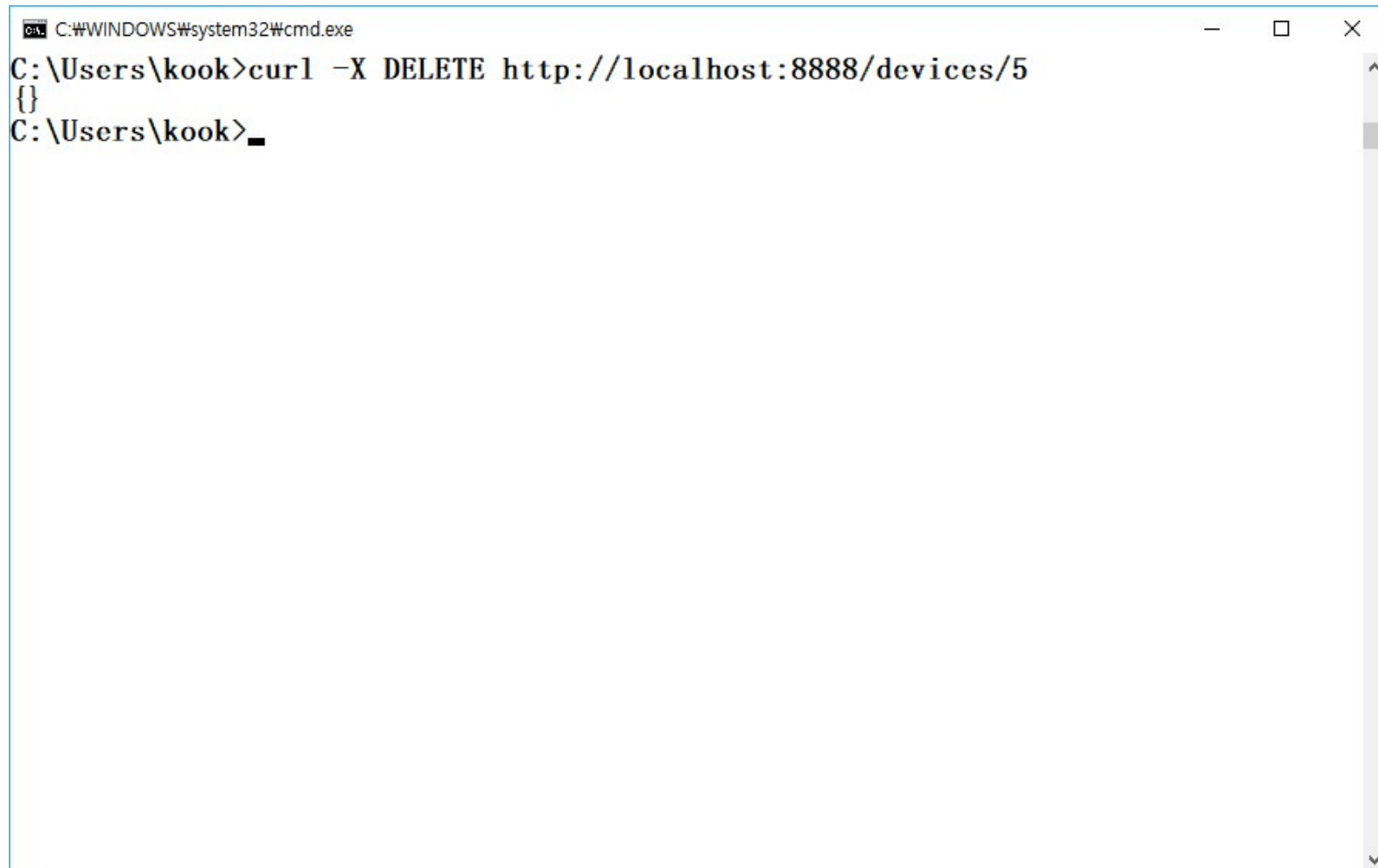
```
listener.ora  insnames.ora  start.ps1  connect_test.js  new 2  new 3  package.json  devices.json
1  {
2    "devices": [
3      {
4        "id": 1,
5        "content": "doorlock",
6        "completed": false
7      },
8      {
9        "id": 2,
10       "content": "fan",
11       "completed": true
12     },
13     {
14       "id": 3,
15       "content": "heater",
16       "completed": false
17     },
18     {
19       "id": 4,
20       "content": "computer",
21       "completed": true
22     },
23     {
24       "id": 5,
25       "content": "speaker",
26       "state": 40
27     }
28   ]
29 }
```

JSON file      length : 414    lines : 29      Ln : 27    Col : 6    Sel : 72 | 5      Windows (CR LF)    ANSI    INS

# REST API Test

## ▶ REST API Test #5

↘ Method: DELETE

A screenshot of a Windows command prompt window. The title bar at the top reads "C:\WINDOWS\system32\cmd.exe". The command prompt shows the user "kook" at the "C:\Users\" directory. The user has entered the command "curl -X DELETE http://localhost:8888/devices/5". The output of the command is "{}", indicating a successful DELETE operation. The prompt is now waiting for the next command.

```
C:\WINDOWS\system32\cmd.exe
C:\Users\kook>curl -X DELETE http://localhost:8888/devices/5
{}
C:\Users\kook>
```

# REST API Test

## ▶ REST API Test #5

↘ Method: DELETE

The screenshot displays a REST API client interface. The top panel shows a DELETE request to `http://localhost:8888/devices/5` with an empty response body. The bottom panel shows a JSON file named `devices.json` containing an array of device objects. The status bar at the bottom indicates the file is a JSON file, 340 characters long, and 24 lines.

```
C:\WINDOWS\system32\cmd.exe
C:\Users\kook>curl -X DELETE http://localhost:8888/devices/5
{}
C:\Users\kook>
```

```
{
  "devices": [
    {
      "id": 1,
      "content": "doorlock",
      "completed": false
    },
    {
      "id": 2,
      "content": "fan",
      "completed": true
    },
    {
      "id": 3,
      "content": "heater",
      "completed": false
    },
    {
      "id": 4,
      "content": "computer",
      "completed": true
    }
  ]
}
```

JSON file      length : 340    lines : 24      Ln : 24    Col : 2    Sel : 0 | 0      Windows (CR LF)    ANSI    INS

# Profile

KOOK Joongjin, Ph.D. Assistant Professor (kook@smu.ac.kr, tipsiness@gmail.com)

Department of Information Security Engineering, Sangmyung University

-KEA, Technical Adviser, Electronics IoT Collaboration Center, KEA

-한국저작권위원회 SW감정인

-KISTEP, 기술수준평가 초연결 사물인터넷 기술 핵심 전문가

-KISTEP, [국가과학기술표준분류체계 개정 프로세스 개선 및 전면 개정을 위한 기획 연구] 정보통신분야 분과위원

-산업통상자원부 IoT가전/스마트홈 기획위원

## <저서 및 역서>

- 라즈베리 파이로 구현하는 사물인터넷 프로젝트 (위키북스)
- 임베디드의 모든 것 (위키북스)
- 리눅스 커널 in a Nutshell (프리렉)
- 언로킹 안드로이드 (프리렉)
- 안드로이드 ADK와 아두이노 (한국전자정보통신산업진흥회)
- 안드로이드 내부구조의 이해 (한국전자정보통신산업진흥회)

## <교육>

- 삼성전자, LG전자, SK텔레시스, KT Tech, 팬택, 모토로라, Foxconn