

데이터베이스

# Web APIs & 공공데이터 활용

2023.가을

인천대학교

임베디드 시스템 공학과

wchkang@inu.ac.kr

강우철

# Outline

open API

- What is Web APIs?
- HTTP
- Web service layer
  - SOAP vs. REST
- Message Formatting
  - XML vs. JSON
- Lab
  - Naver geocode API
  - 공공데이터 포털 ([www.data.go.kr](http://www.data.go.kr))
  - Using Python to access Web APIs

# APIs?

- Application Programming Interface

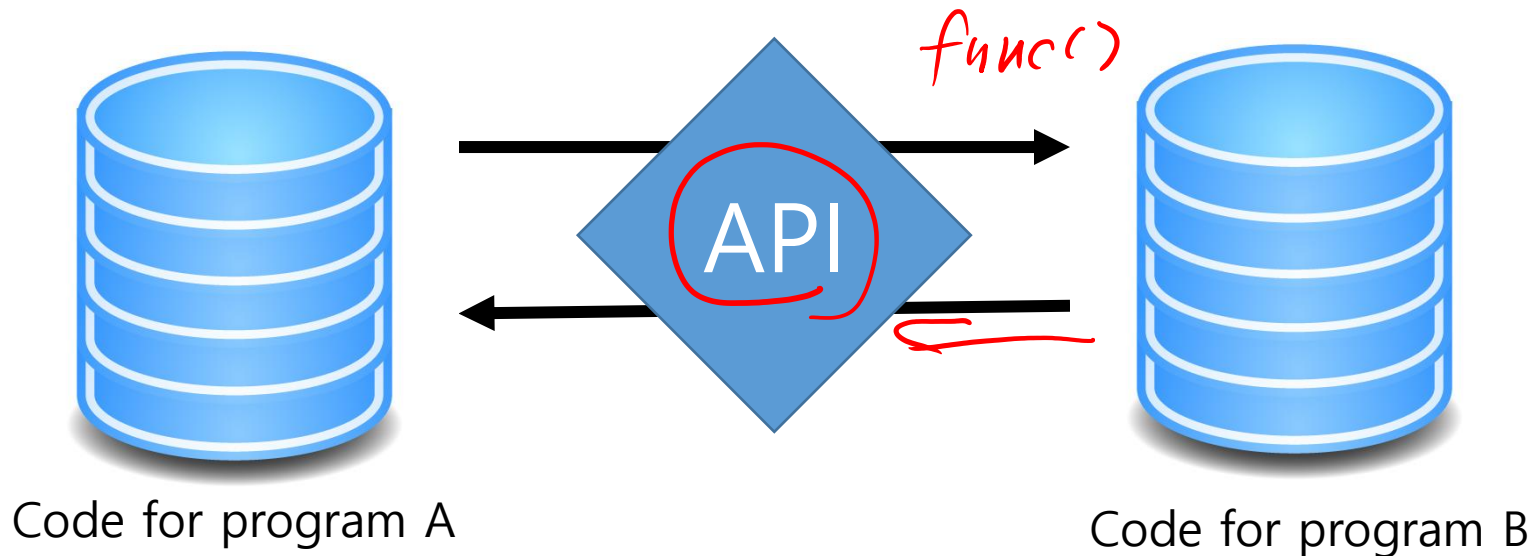
# APIs? 환경

- Example: A **music player application** makes sound using speakers



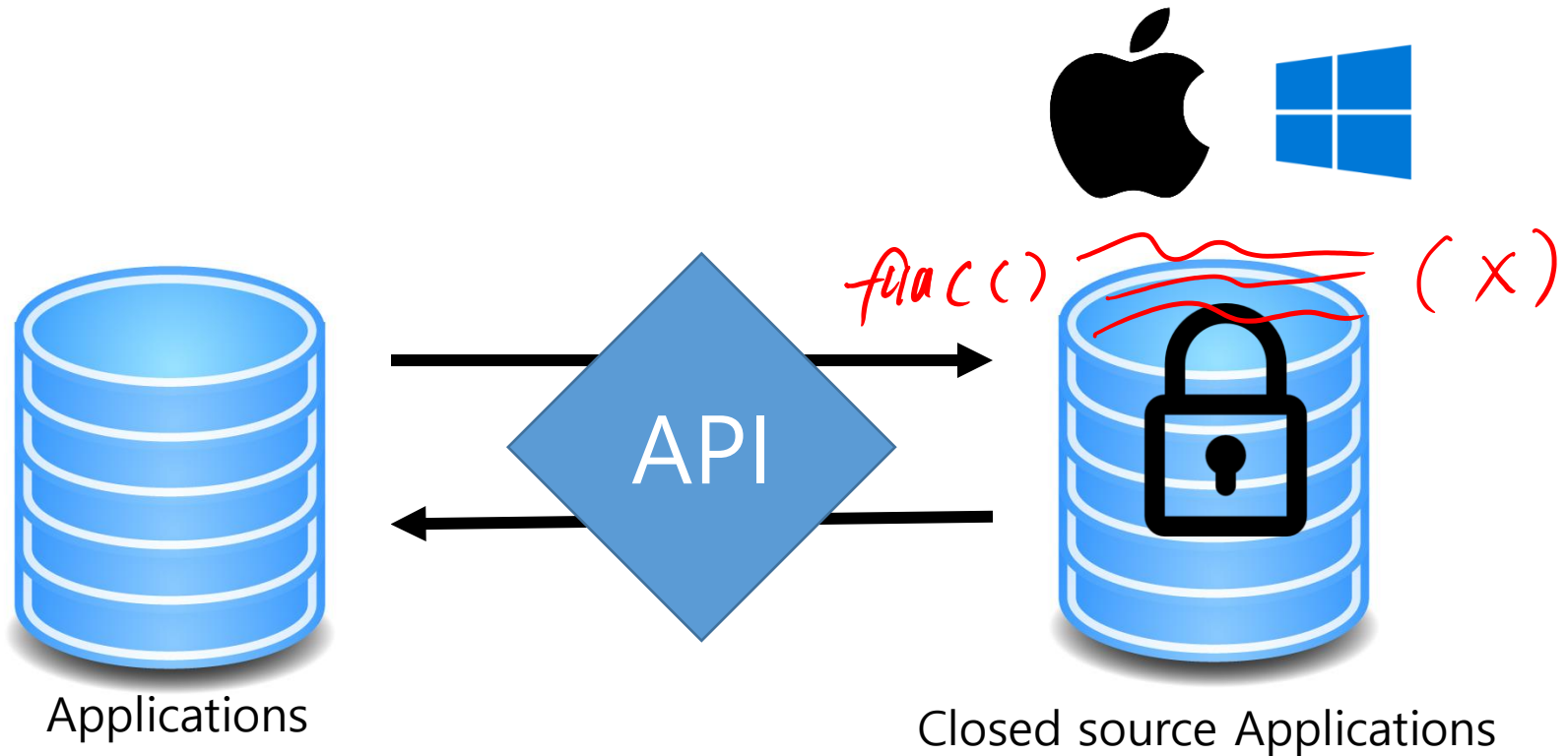
# APIs?

- Application Programming Interface
  - Any method of communication between any two entities of code
  - Connection points in code that allow one application to talk to another



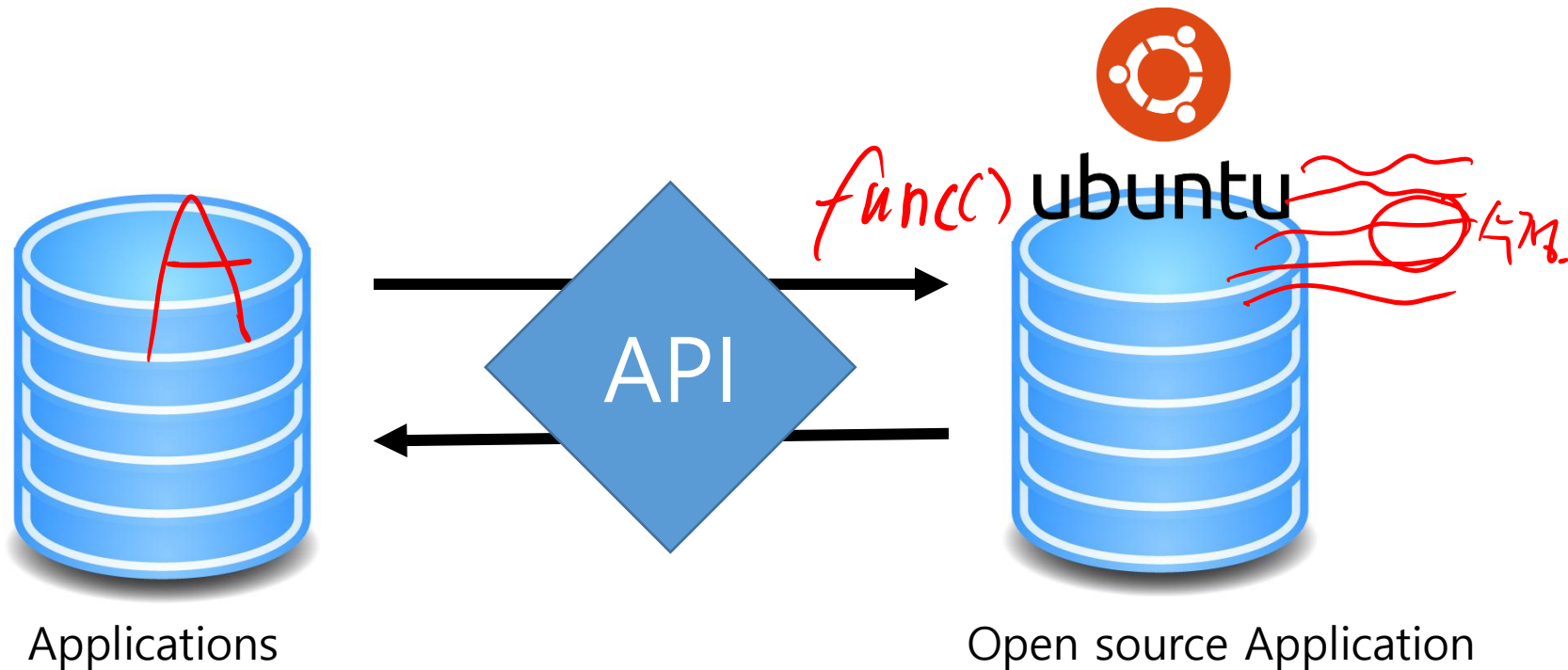
# APIs?

- APIs allow applications to use internal functionalities from the outside world without exposing all of the internal code

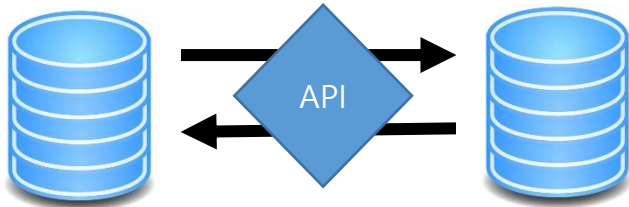


# APIs?

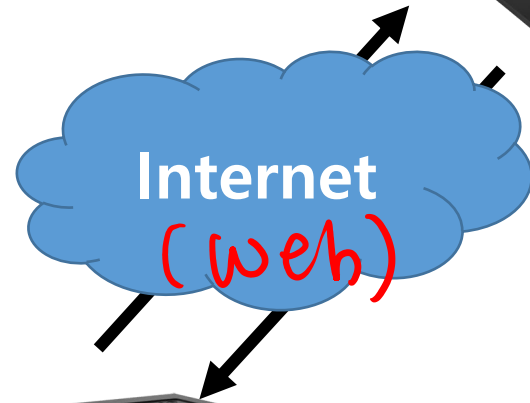
- APIs allow applications to use internal functionalities from the outside world without exposing all of the internal code



## APIs in a single machine

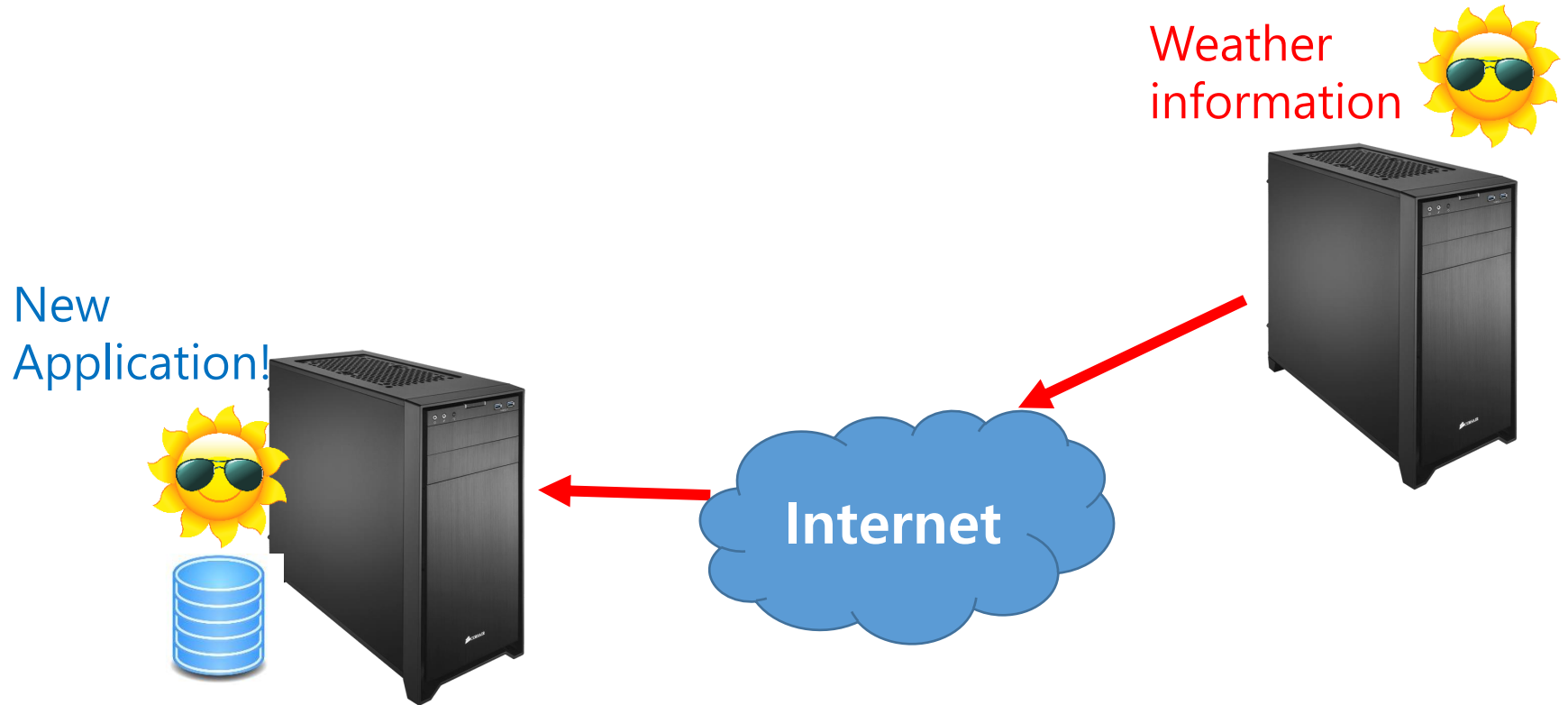


Web APIs



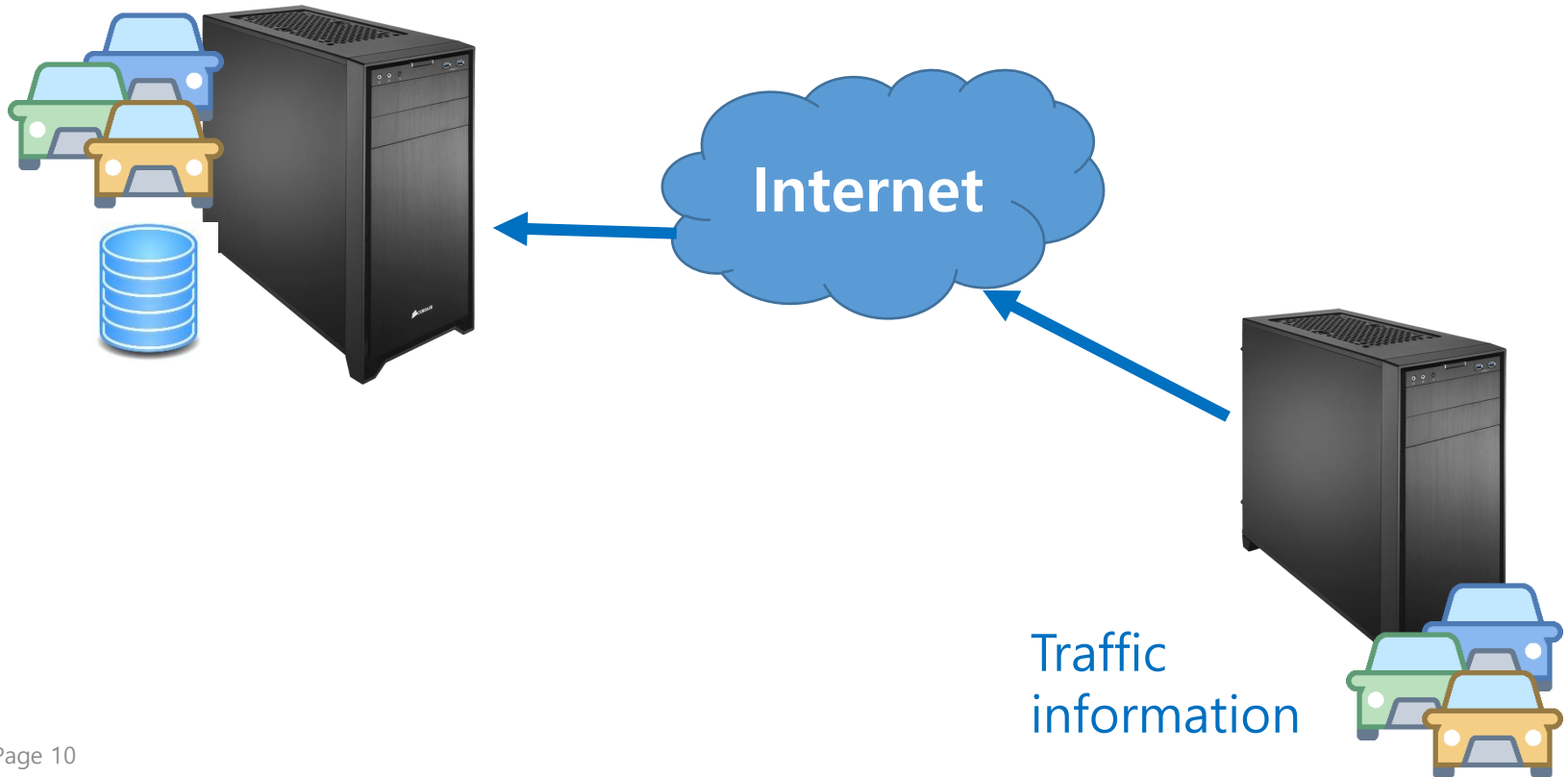


# Web APIs



# Web APIs

New  
Application!

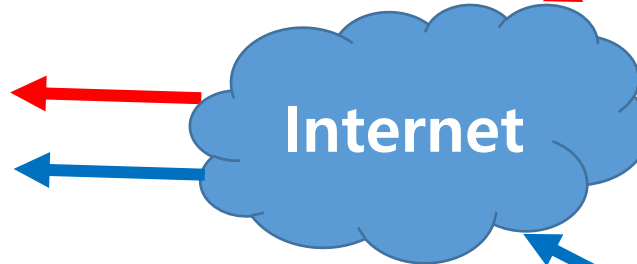
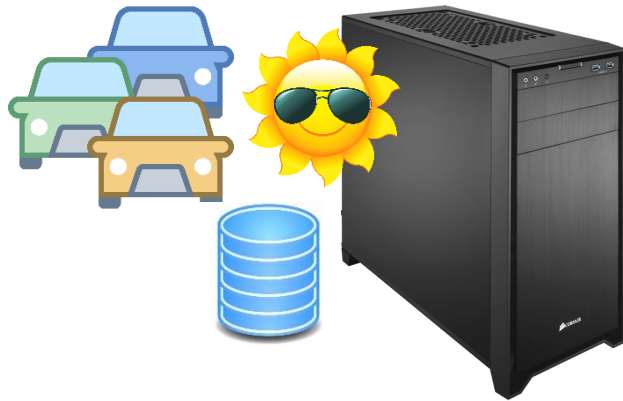


# Web APIs

New  
Application!

## Mashup 메시업

Weather  
information 

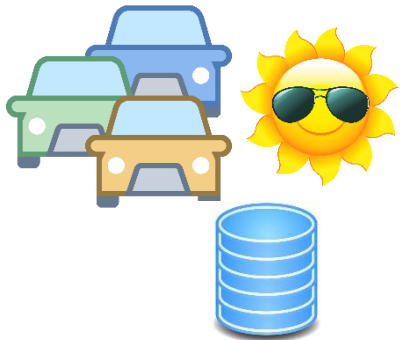


Traffic  
information

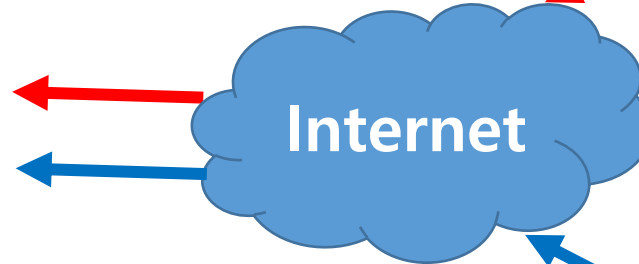


# Web APIs

New  
Application!



## Mashup 메시업



Weather  
information



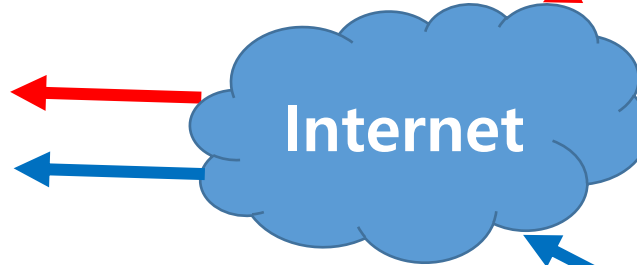
Traffic  
information



# Web APIs

New  
Application!

## Mashup 메시업



**NAVER**



Google  
maps



# Web APIs

- Many Internet companies provide web APIs

**NAVER**



follow us on  
**twitter**



**Google**  
Translate API

**Google**  
maps



Fork me on GitHub

Web APIs:

*Why is this important for Internet companies?*



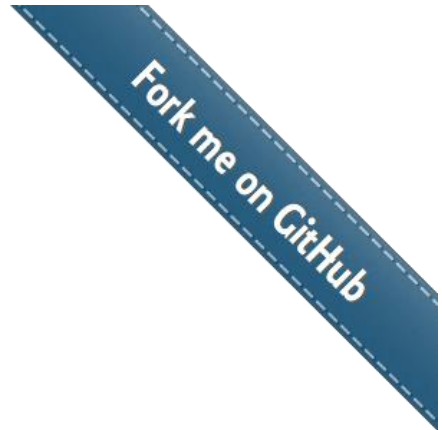
# Web APIs

- more traffic = more users = more money

**NAVER**



follow us on  
**twitter**



**Google**  
Translate API

**Google**  
maps





# Web API Protocols

- How these different types of devices can communicate?



# Web API Protocols

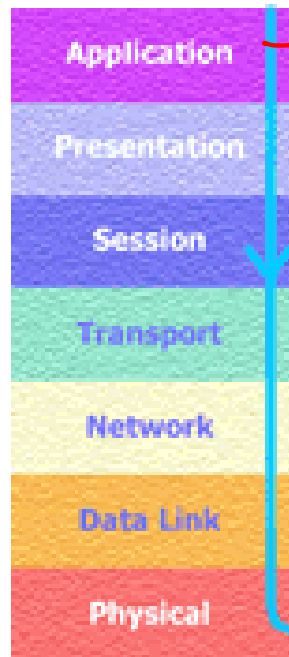
- How these different types of devices can communicate?



# Open Systems Interconnection (OSI) Model

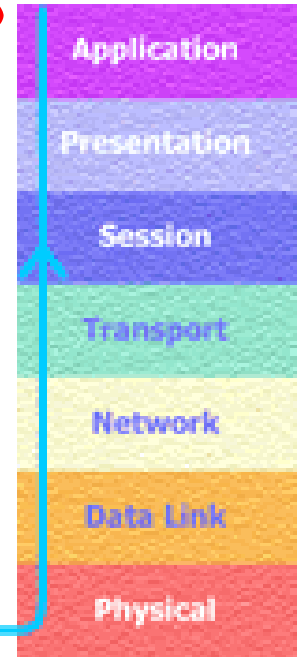


Computer A

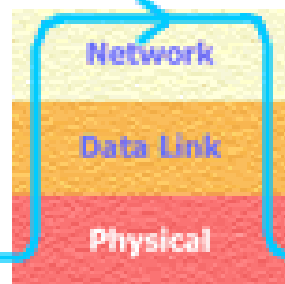


통신 흐름.

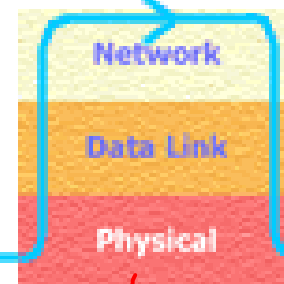
Computer B



router 1

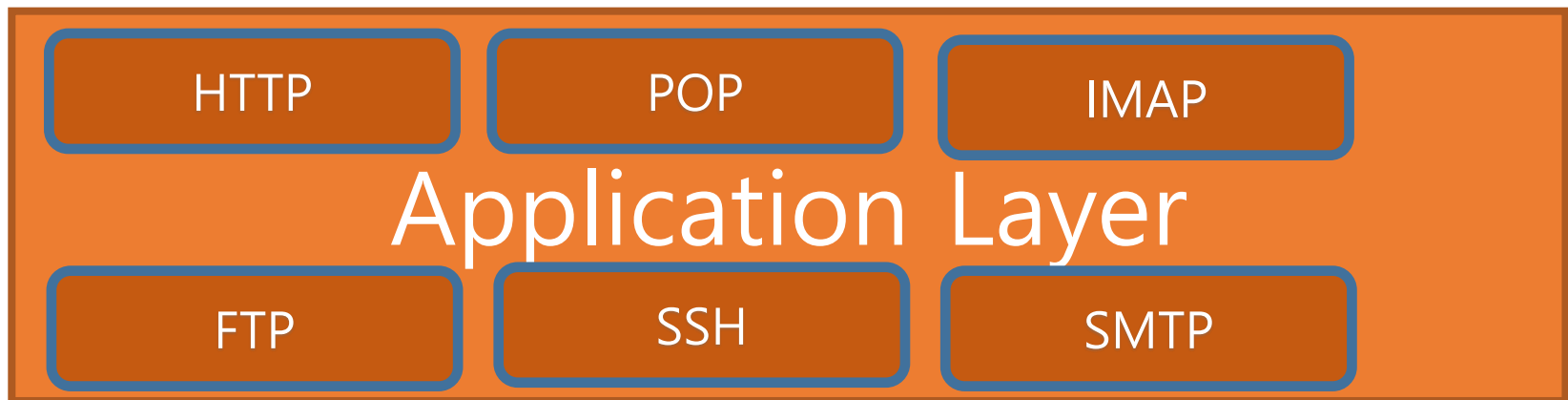


router 2



# Application Layer

- Applications directly exploits application layer to talk to other machines



# Application Layer

- Web APIs need 2 more layers



The diagram illustrates the layers required for web APIs. It consists of three main stacked rectangular blocks. The top block is blue and labeled 'Messaging Format'. The middle block is gray and labeled 'Web Service (API)'. The bottom block is orange and labeled 'Application Layer'. Inside the orange block, there is a smaller, rounded orange rectangle with a blue border, labeled 'HTTP'.

Messaging Format

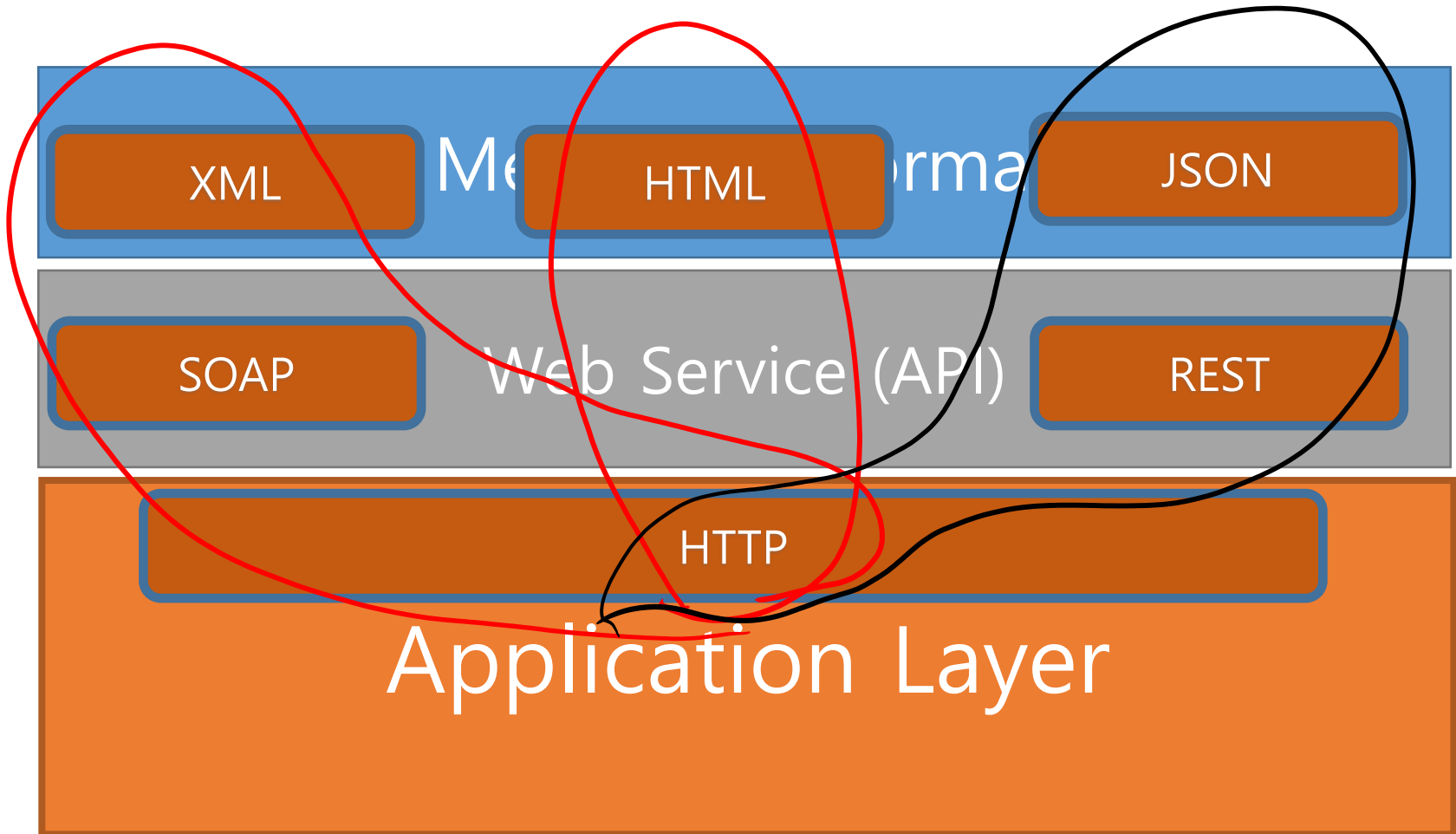
Web Service (API)

HTTP

Application Layer

# Application Layer

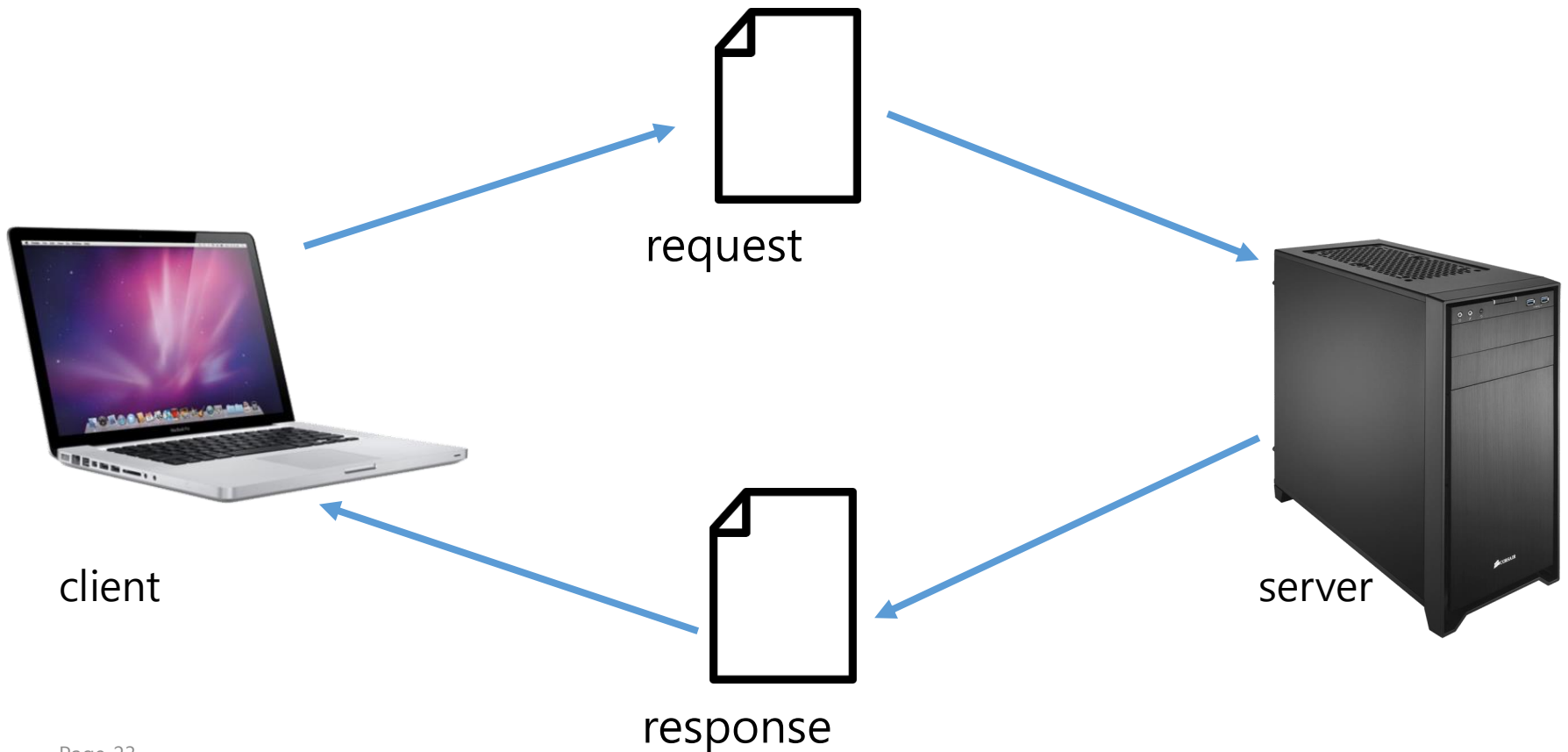
- Web APIs need 2 more layers



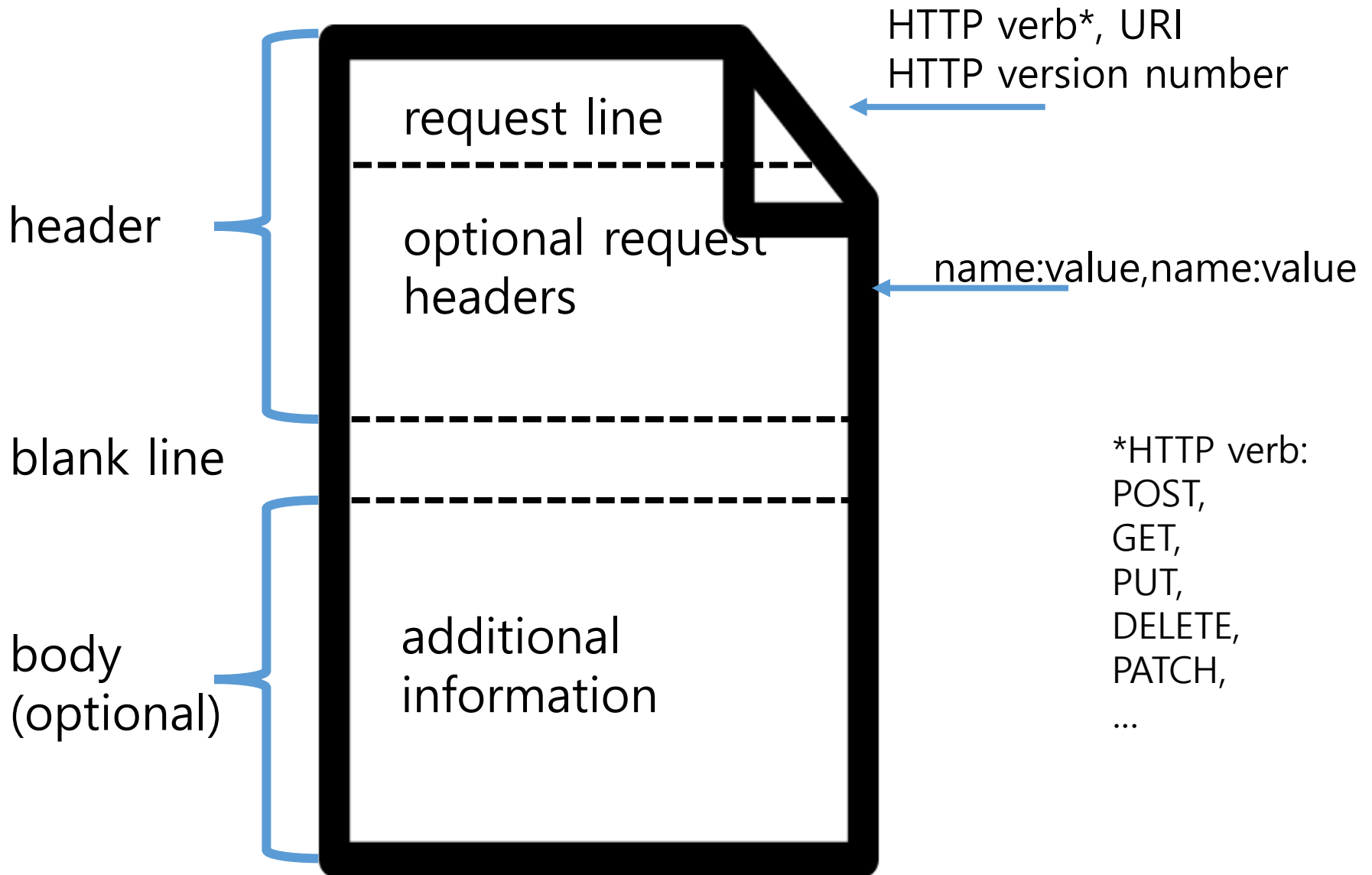
# HTTP request and response

## "Pull Protocol"

=> Communication is initiated by a client



# HTTP Requests





# HTTP Requests

```
GET puppies.html HTTP/1.1
Host: www.puppyshelter.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35
```

```
puppyId=12345&name=Fido+Simpson
```

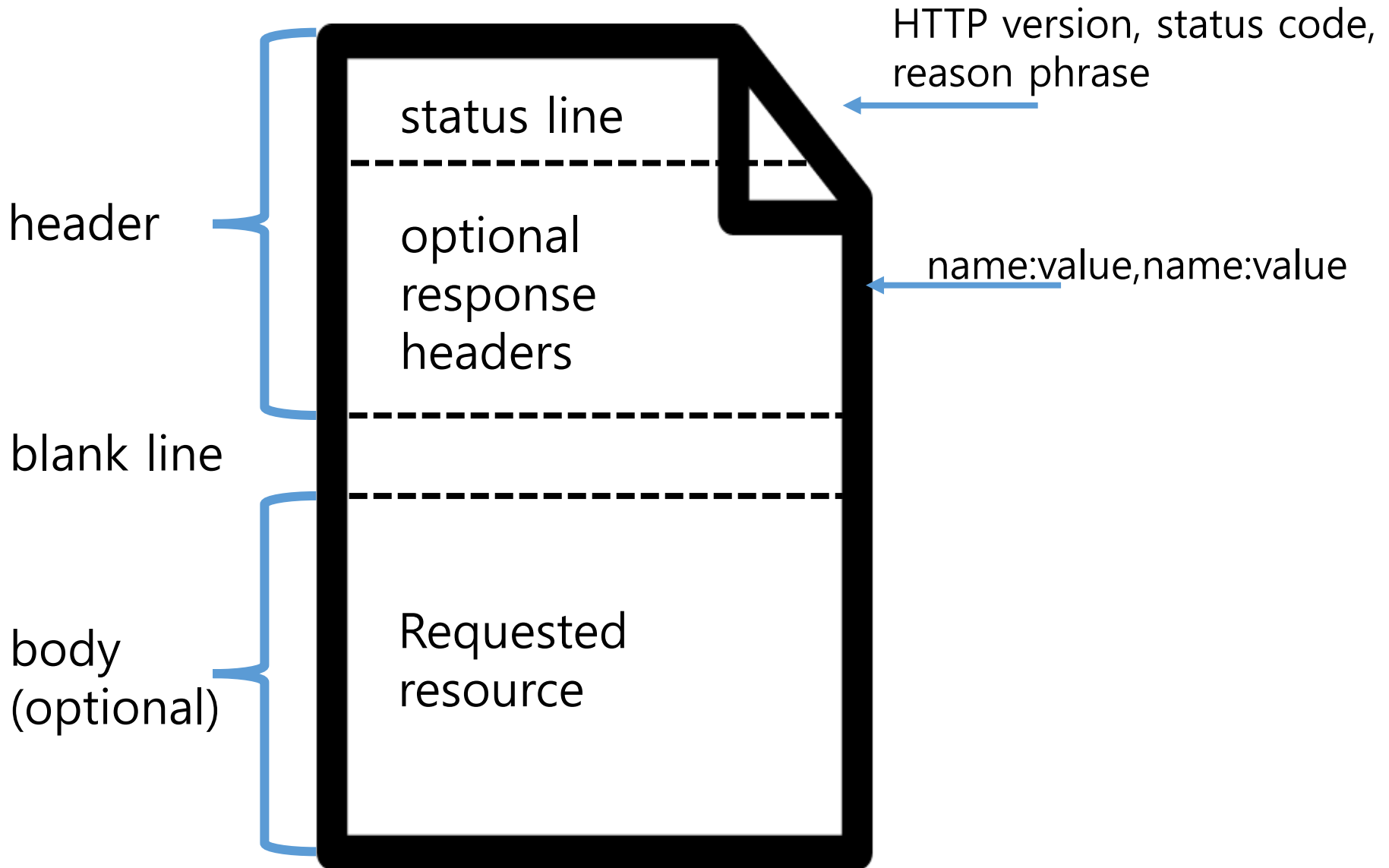
# HTTP Request

GET /home.html HTTP/1.1

POST /index.html HTTP/1.1

DELETE /query.html HTTP/1.1

# HTTP Response



# HTTP Response

"200 OK"

"404 Not Found"

"403 Forbidden"

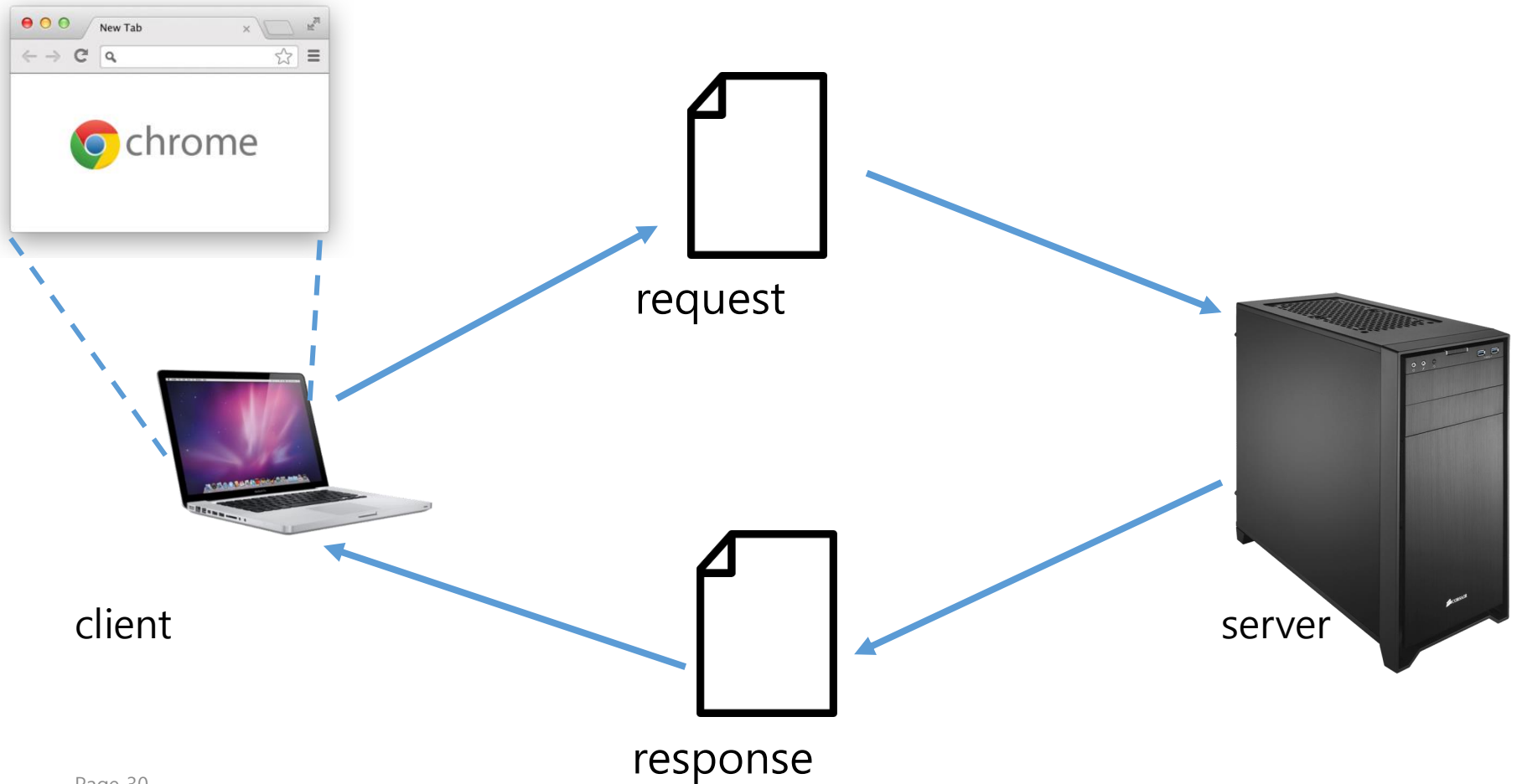
"500 Internal Server Error"

# HTTP Response

```
HTTP/1.1 200 OK
Date: Fri, 04 Sep 2015 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb 2014
ETag: "0-23-4024c3a5:
ContentType: text/html
ContentLength: 35
Connection: KeepAlive
KeepAlive: timeout=15, max = 100

<h1>Welcome to my home page!</h1>
```

# HTTP request and response



# SOAP vs. REST

SOAP	REST
Developed by Microsoft in 1998	Dissertation by <i>Roy Thomas Fielding</i> in 2000
Must use XML for message format	Can use any type of message format
Works with HTTP and other application layer protocols	Uses HTTP verbs to manage resources
WS-security WS-Addressing WS-ReliableMessaging WSDL	REST constraints <ol style="list-style-type: none"><li>1. Client-server</li><li>2. Stateless</li><li>3. Cacheable</li><li>4. Uniform interface</li><li>5. Layered system</li><li>6. (optional) code on demand</li></ol>

# SOAP vs. REST

- Simple web service as an example: querying a phonebook application for the details of a given user
- Using Web Services and SOAP, the request would look something like this:

```
<?xml version="1.0"?>
```

```
<soap:Envelope
```

```
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
```

```
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
```

```
    <soap:body pb="http://www.acme.com/phonebook">
```

```
      <pb:GetUserDetails>
```

```
        <pb:UserID>12345</pb:UserID>
```

```
      </pb:GetUserDetails>
```

```
    </soap:Body>
```

```
</soap:Envelope>
```



# SOAP vs. REST

- Simple web service as an example: querying a phonebook application for the details of a given user
- And with REST? The query will probably look like this:  
<http://www.acme.com/phonebook/UserDetails?firstName=John&lastName=Doe>

# XML vs. JSON

XML	JSON
eXtensible Markup Language	JavaScript Object Notation
Developed in 1997	Developed in 2001
Uses identifying tags similar to HTML	Derived from JavaScript
	Can be condensed to reduce file size

# Bookstore XML schema

```
<?xml version="1.0" ?>
<!-- XSD for Bookstore-XSD.xml -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Bookstore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" type="BookType"
          minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="Author" type="AuthorType"
          minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:key name="BookKey">
      <xsd:selector xpath="Book" />
      <xsd:field xpath="@ISBN" />
    </xsd:key>
    <xsd:key name="AuthorKey">
      <xsd:selector xpath="Author" />
      <xsd:field xpath="@Ident" />
    </xsd:key>
    <xsd:keyref name="AuthorKeyRef" refer="AuthorKey">
      <xsd:selector xpath="Book/Authors/Auth" />
      <xsd:field xpath="@authIdent" />
    </xsd:keyref>
    <xsd:keyref name="BookKeyRef" refer="BookKey">
      <xsd:selector xpath="Book/Remark/BookRef" />
      <xsd:field xpath="@book" />
    </xsd:keyref>
  </xsd:element>
  <xsd:complexType name="BookType">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" />
      <xsd:element name="Authors">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Auth" maxOccurs="unbounded">
              <xsd:complexType>
```

# Bookstore XML document

```
<Bookstore>
  <Book ISBN="ISBN-0-13-713526-2" Price="100">
    <Title>A First Course in Database Systems</Title>
    <Authors>
      <Auth authIdent="JU" />
      <Auth authIdent="JW" />
    </Authors>
  </Book>
  <Book ISBN="ISBN-0-13-815504-6" Price="85">
    <Title>Database Systems: The Complete Book</Title>
    <Authors>
      <Auth authIdent="HG" />
      <Auth authIdent="JU" />
      <Auth authIdent="JW" />
    </Authors>
    <Remark>
      Amazon.com says: Buy this book bundled with
      <BookRef book="ISBN-0-13-713526-2" /> - a great deal!
    </Remark>
  </Book>
  <Author Ident="HG">
    <First_Name>Hector</First_Name>
    <Last_Name>Garcia-Molina</Last_Name>
  </Author>
  <Author Ident="JU">
    <First_Name>Jeffrey</First_Name>
    <Last_Name>Ullman</Last_Name>
  </Author>
  <Author Ident="JW">
    <First_Name>Jennifer</First_Name>
    <Last_Name>Widom</Last_Name>
  </Author>
</Bookstore>
```

# Bookstore JSON document



```
{ "Books":  
  [  
    { "ISBN": "ISBN-0-13-713526-2",  
      "Price": 85,  
      "Edition": 3,  
      "Title": "A First Course in Database Systems",  
      "Authors": [ { "First_Name": "Jeffrey", "Last_Name": "Ullman" },  
                   { "First_Name": "Jennifer", "Last_Name": "Widom" } ] },  
    { "ISBN": "ISBN-0-13-815504-6",  
      "Price": 100,  
      "Remark": "Buy this book bundled with 'A First Course' - a great deal!",  
      "Title": "Database Systems: The Complete Book",  
      "Authors": [ { "First_Name": "Hector", "Last_Name": "Garcia-Molina" },  
                   { "First_Name": "Jeffrey", "Last_Name": "Ullman" },  
                   { "First_Name": "Jennifer", "Last_Name": "Widom" } ] },  
  ],  
  "Magazines":  
  [  
    { "Title": "National Geographic",  
      "Month": "January",  
      "Year": 2009 },  
    { "Title": "Newsweek",  
      "Month": "February",  
      "Year": 2009 },  
  ],  
}
```

# JSON Introduction

- De-facto standard for “serializing” data objects, usually in files
- Human-readable, useful for data interchange
- Also useful for representing & storing semistructured data
- Basic constructs

- Base values
  - number, string, Boolean, ..
- Objects: { }
- Sets of label-value pairs
- Array: [ ]
  - Lists of values

```
{ "Books":  
  [  
    { "ISBN":"ISBN-0-13-713526-2",  
      "Price":85,  
      "Edition":3,  
      "Title":"A First Course in Database Systems",  
      "Authors":[ { "First_Name":"Jeffrey", "Last_Name":"Ullman"},  
                   { "First_Name":"Jennifer", "Last_Name":"Widom" } ] }  
    ,  
    { "ISBN":"ISBN-0-13-815504-6",  
      "Price":100,  
      "Remark":"Buy this book bundled with 'A First Course' - a great deal!",  
      "Title":"Database Systems:The Complete Book",  
      "Authors":[ { "First_Name":"Hector", "Last_Name":"Garcia-Molina"},  
                   { "First_Name":"Jeffrey", "Last_Name":"Ullman"},  
                   { "First_Name":"Jennifer", "Last_Name":"Widom" } ] }  
  ],  
  "Magazines":  
  [  
    { "Title":"National Geographic",  
      "Month":"January",  
      "Year":2009 }  
    ,  
    { "Title":"Newsweek",  
      "Month":"February",  
      "Year":2009 }  
  ]  
}
```

# Relational Model vs. JSON

	Relational	JSON
Structure	Tables	Nested sets and arrays
Schema	Fixed in advance	"self-describing" Flexible
Queries	Simple expressive languages (SQL)	Not widely used
implementation	Native systems using DBMS	Coupled with programming languages

# 실습: OpenAPI 사용

- Naver Geocoding
- Naver Reverse Geocoding
- 공기질 서비스 (환경공단)
- 실습을 위해서
  - Python 개발환경 설치
  - [www.ncloud.com](http://www.ncloud.com)에 계정생성하고, 결제수단 등록
  - [www.data.go.kr](http://www.data.go.kr)에 계정생성

# Delving into APIs: Naver Maps API

공공기관용 금융클라우드

로그아웃 Languages ▾



소개 서비스 솔루션 마켓플레이스 요금 고객지원 파트너 가이드센터 마이페이지



문의하기 ▾

콘솔



Application Services

GeoLocation

Maps

CAPTCHA

nShortURL

Simple & Easy Notification Service

API Gateway

Search Trend

RabbitMQ ▶

https://www.ncloud.com



## Maps

네이버 지도 기능을 활용해 다양한 위치 기반 서비스를 만들 수 있습니다.

이용 신청하기

요금 계산하기



특징

상세기능

적용 서비스

요금

API 안내

## 최고 수준의 지도 서비스를 제공하는 네이버 지도 API

웹 서비스 또는 애플리케이션에 네이버 지도를 활용할 수 있도록 다양한 기능을 제공합니다.

간단한 약도부터 주변 맛집이나 유명 관광지 표시까지, 요청하는 여러 정보들을 지도 위에 표현할 수 있습니다

### 신뢰할 수 있는 데이터

네이버 지도는 네이버와 관련된 전문기관들의 신뢰할 수 있는 정보들로 구성되어 있으며, 다년간 네이버 지도를 서비스하며 축적해 온 노하우를 활용하여 지속적으로 개선하고 발전시켜가고 있습니다.

Page 42

### 국내 No. 1 지도 서비스

대한민국에서 가장 많은 사용자가 이용하는 국내 최대 지도 서비스로, 수시 업데이트를 통해 최신의 지도 정보를 유지합니다. 또한, 네이버 지도를 이용하는 사용자들로부터 매일 접수 받은 수 많은 "지도 수정 요청"을 실시간으로 반영하여 최신의 데이터를 제공합니다.

### 다양한 위치기반 서비스 제공

물류, 관제, 통신, 유통 등 여러 사업 분야에서 이용할 수 있는 Maps, Directions, Places 관련 다양한 API 기능들이 제공됩니다. 높은 품질의 지도 데이터와 다양한 기능의 API를 이용하여 위치와 이동 관련 비즈니스를 하는 사업자들이 보다 쉽고 편리하게 경쟁력 있는 서비스를 구축할 수 있습니다.





네이버 클라우드 플랫폼에서 제공하는 **NAVER** 최신기술 기반의 **AI** 와 **NAVER Service** 를 이용하여,  
더욱 경쟁력 있는 서비스를 쉽고 편리하게 구축할 수 있습니다. [Region 통합 서비스](#)

## AI Service [자세히 보기](#)

CLOVA, Papago 등 네이버의 풍부한 데이터를 기반으로 학습된 최신 인공지능 서비스를 이용하실 수 있습니다.



- CLOVA Speech Recognition (CSR)
- CLOVA Face Recognition (CFR)
- CLOVA Voice - Premium
- CLOVA Sentiment
- CLOVA Summary



- Papago Text Translation
- Papago Language Detection
- Papago Website Translation
- Papago Doc Translation
- Papago Image Translation(Text)
- Papago Image Translation(Image)

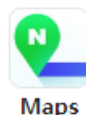


Machine  
Learning

- Pose Estimation
- Object Detection

## Application Service [자세히 보기](#)

네이버에서 사용하는 기술과 서비스를 API로 제공합니다. 이를 이용하여 다양한 서비스를 개발할 수 있습니다.



- Mobile Dynamic Map
- Web Dynamic Map
- Static Map
- Directions 5
- Geocoding
- Reverse Geocoding
- Directions 15



- CAPTCHA (Image)
- nShortURL
- Korean Name Romanizer
- CAPTCHA (Audio)
- Search Trend

# Application 1

등록한 Application 정보를 확인하고 관리합니다.

+ Application 등록






개발 가이드 

상품 더 알아보기 

새로 고침 

▼

삭제

<input type="checkbox"/> App 이름	서비스구분	당일 사용량		당월 사용량		한도 및 알림 설정	등록일
<div><input type="checkbox"/> OPENAPI-DBCLASS</div> <div><div>인증 정보 </div><div>수정</div></div>	 Geocoding 	0% <div></div>	0/3,000,000 회	0% <div></div>	0/3,000,000 회	<div>한도 및 알림 설정</div>	2022-09-28
	 Reverse Geocoding 	0% <div></div>	0/3,000,000 회	0% <div></div>	0/3,000,000 회	<div>한도 및 알림 설정</div>	

# Application 1

등록한 Application 정보를 확인하고 관리합니다.

[+ Application 등록](#)[개발 가이드](#)[상품 더 알아보기](#)[새로 고침](#)

## 인증 정보



### Application key

Application 이름

OPENAPI-DBCLASS

Client ID  
(X-NCP-APIGW-API-KEY-ID)



Client Secret  
(X-NCP-APIGW-API-KEY)



재발급

### 서비스환경

Web 서비스 URL

Android 앱 패키지 이름

iOS Bundle ID

- 정보 변경을 원하시면, Application 선택 후 변경 버튼을 통해 진행해주세요.

✓ 확인

이 값은 API를 호출할 때 HTTP 헤더값에 포함해서  
전송해야 호출이 가능합니다.

한도 및 알림 설정

등록일

73,000,000  
회

한도 및 알림 설정

2022-09-  
28

73,000,000  
회

한도 및 알림 설정

# Delving into APIs: Geocoding API

필터

## > Maps 시작

## ▼ Maps 사용

Web Dynamic Map API

Mobile Dynamic Map

Android SDK

Mobile Dynamic Map iOS  
SDK

Static Map API

Directions 5 API

Directions 15 API

Geocoding API

Reverse Geocoding API

지도 앱 연동 URL Scheme

Maps 권한 관리

Maps 용어

Maps 릴리스 노트

## > CAPTCHA

## Geocoding API

인쇄 공유 PDF

태그

Maps Developers rest api

Classic/VPC Classic/VPC 환경에서 이용 가능합니다.

Geocoding API는 REST 형식을 따르는 네이버 지도 인터페이스로, HTTP GET 메소드를 이용하여 지번 도로명을 좌표값으로 반환 받는 API입니다.

Geocoding API 이용 방법은 API 가이드의 Geocoding에서 상세하게 설명하고 있습니다. API 가이드의 Geocoding으로 이동하여 정보를 확인하려면 **API 가이드**를 클릭해 주십시오.

기능	바로가기
Geocoding API	<a href="#">API 가이드</a>

### 참고

- Geocoding 메서드는 Web Dynamic Maps API로도 제공됩니다. Geocoder 서브 모듈 사용에 대한 자세한 정보는 다음과 같습니다.
  - [Geocoder를 활용한 주소와 좌표 검색 API 호출하기](#)
- Geocoding API 사용에 도움이 되는 링크는 다음과 같습니다.
  - [인증키 발급](#)
  - [API 버그 신고 및 문의](#)

## [Lab] geocoding: 주소=> 좌표

- Naver Maps의 geocode API를 사용하여 주소로 좌표(위도/경도)로 변환하라

파라미터	데이터 타입	필수 여부	설명
query	string	Y	주소
coordinate	string	N	- 검색 중심 좌표 'lon,lat' 형식으로 입력
filter	string	N	- 검색 결과 필터링 조건 '필터 타입@코드1;코드2;...' 형식으로 입력 제공하는 필터 타입은 다음과 같음: HCODE : 행정동 코드 BCODE : 법정동 코드  예) HCODE@4113554500;4113555000
page	number	N	- 페이지 번호 기본값은 1
count	number	N	- 결과 목록 크기 입력 범위: 1~100 기본값: 10

# [Lab] *requests* python module

## Requests: HTTP for Humans™

Release v2.31.0. ([Installation](#))

downloads/month 310M | license Apache 2.0 | wheel yes | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11

**Requests** is an elegant and simple HTTP library for Python, built for human beings.

---

### Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type": "User" ... '
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
```

See [similar code](#), sans Requests.

**Requests** allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to [urllib3](#).

- <https://requests.readthedocs.io>

## Requesting from Python Code

```
import requests

def getGeoCode(address, client_id, client_secret):

    header = {
        "X-NCP-APIGW-API-KEY-ID": client_id,
        "X-NCP-APIGW-API-KEY": client_secret,
    }

    endpoint = "https://naveropenapi.apigw.ntruss.com/map-geocode/v2/geocode"
    url = f"{endpoint}?query={address}"

    res = requests.get(url, headers=header)

    return res
```

## Requesting from Python Code

```
18 if __name__ == '__main__':
19     address = "인천광역시 연수구 아카데미로 119"
20     client_id = "x[REDACTED]2"
21     client_secret = "wAcV[REDACTED]f1109181Lk"
22
23     response = getGeoCode(address, client_id, client_secret)
24
25     if (response.status_code == 200):
26         result = response.json()
27         print(f'Getting geocode for :{address}')
28         print(f"X: {result['addresses'][0]['x']}")
29         print(f"Y: {result['addresses'][0]['y']}")
30     else:
31         print(f'Error code: {response}')
32
```

문제    출력    디버그 콘솔    터미널    JUPYTER

```
• wochul@rtsslalab:~/class/db2022$ python3 geocodetest.py
Getting geocode for :인천광역시 연수구 아카데미로 119
X: 126.6332532
Y: 37.3754977
```



## [Lab] Reverse geocoding: 위치 -> 주소

- Naver Maps의 reversegeocode API를 사용하여 좌표(위도/경도)를 주소로 변환하는 함수 getReverseGeoCode를 작성하라
- 반환된 json을 파싱하여 주소 정보를 표시하라

# Delving into APIs: 공공데이터 포털 ■ <http://www.data.go.kr>



이 누리집은 대한민국 공식 전자정부 누리집입니다. 목록등록관리시스템

로그인

회원가입

사이트맵

ENG

**DATA** 공공데이터포털  
.GO.KR

데이터찾기

국가데이터맵

데이터요청

데이터활용

정보공유

이용안내

어떤 공공데이터를 찾으시나요?



인기검색어



3. 전기차



검색조건



분류체계



서비스유형



확장자



① 검색도움말

콘텐츠 바로가기

테마별

카테고리별

국가중점데이터별

제공기관유형별



Page 52



교육



국토관리



공공행정



재정금융



산업고용



사회복지



식품건강



문화관광



# Delving into APIs: 대기오염정보 조회 서비스

오픈API 상세



URL 복사

## 1. 활용신청

**XML** **JSON** 한국환경공단\_에어코리아\_대기오염정보

각 측정소별 대기오염정보를 조회하기 위한 서비스로 시간별, 시도별 대기오염 정보와 통합대기환경지수 나쁨 이상 측정소 내역, 대기질(미세먼지/오존) 예보 통보 내역 등을 조회할 수 있다.

※ 운영계정으로 사용하고자 할 경우 에어코리아 OpenAPI 사용자 관리시스템(<https://apiweb.airkorea.or.kr>)에 접속하여 인증키 및 개발보고서를 등록해주셔야 담당자 승인 후 사용이 가능합니다.



26



12

관심

활용신청

오류신고 및

담당자 문의

다른 사용자들이 활용한 데이터

오픈 API

공공데이터활용지원센터\_  
보건복지부 코로나19해외  
발생 현황

파일데이터

기상청\_단기예보



OpenAPI 정보

메타데이터 다운로드

분류체계	환경 - 대기	제공기관	한국환경공단
관리부서명	대기환경처 대기정책지원부	관리부서 전화번호	032-590-3508
API 유형	REST	데이터포맷	JSON+XML
활용신청	12864	키워드	미세먼지,코로나,날씨
등록 Page 53	2020-12-06	수정	2022-09-27
비용부과유무	무료	신청가능 트래픽	개발계정 : 500 / 운영계정 : 활용사례 등록시 신청하면 트래픽 증가 가능

# Delving into APIs: 대기오염정보 조회 서비스

DATA 공공데이터포털  
.GO.KR

데이터찾기

국가데이터맵

데이터요청

데이터활용

정보공유

이용안내

## 마이페이지

오픈API

개발계정

운영계정

인증키 발급현황

DATA

나의 문의

나의 관심

나의 제공신청

나의 분쟁조정

회원정보 수정

## 개발계정 상세보기

### 기본정보

데이터명	한국환경공단_에어코리아_대기오염정보			상세설명
서비스유형	REST	심의여부	자동승인	
신청유형	개발계정   활용신청	처리상태	승인	
활용기간	2022-09-28 ~ 2024-09-28			

### 서비스정보

참고문서	<a href="#">에어코리아 대기오염정보 조회 서비스 기술문서 v1.0.docx</a>
데이터포맷	JSON+XML
End Point	http://apis.data.go.kr/B552584/ArpltnInforInquireSvc

API 환경 또는 API 호출 조건에 따라 인증키가 적용되는 방식에 다를 수 있습니다.  
포털에서 제공하는 Encoding/Decoding 된 인증키를 적용하면서 구동되는 키를 사용하시기 바랍니다.  
\* 향후 포털에서 더 명확한 정보를 제공하기 위해 노력하겠습니다.

일반 인증키 (Encoding)	e1RLZ9yv8w9Al44eWiGQiOuwBNfohNjWvMZOPVEUDC6pZKgal74qqVDhpC45R0SYUUIn0%2BtXVRshgq7qn12h3w%3D%3D
일반 인증키 (Decoding)	e1RLZ9yv8w9Al44eWiGQiOuwBNfohNjWvMZOPVEUDC6pZKgal74qqVDhpC45R0SYUUIn0+tXVRshgq7qn12h3w==

2. 마이페이지에서 인증키 확인

# Delving into APIs: 대기오염정보 조회 서비스

## 3. 개발문서에서 요청 메시지 포맷과 파라미터 확인(p17)

3) 시도별 실시간 측정정보 조회 상세기능명세

a) 상세기능명세

상세기능 번호	3	상세기능 유형	조회(목록)
상세기능명(국문)	시도별 실시간 측정정보 조회		
상세기능 설명	시도명을 검색조건으로 하여 시도별 측정소목록에 대한 일반 항목과 CAI 최종 실시간 측정값과 지수 정보 조회 기능을 제공하는 시도별 실시간 측정정보 조회		
Call Back URL	http://apis.data.go.kr/B552584/ArpltnInforInquireSvc/getCtprvnRltmMesureDnsty		
최대 메시지 사이즈	[1000] byte		
평균 응답 시간	[500] ms	초당 최대 트래픽선	[50] tps

b) 요청 메시지 명세

항목명(영문)	항목명(국문)	항목크기	항목구분	샘플데이터	항목설명
serviceKey	서비스키	-	1	인증키(URL Encode)	서비스키
returnType	데이터표출방식	4	0	xml	데이터 표출방식 xml 또는 json
numOfRows	한 페이지 결과 수	4	0	100	한 페이지 결과 수
pageNo	페이지 번호	4	0	1	페이지 번호
sidoName	시도 명	10	1	서울	시도 이름 (전국, 서울, 부산, 대구, 인천, 광주, 대전, 울산, 경기, 강원, 충북, 충남, 전북, 전남, 경북, 경남, 제주, 세종)
ver	오퍼레이션 버전	4	0	1.0	버전별 상세 결과 아래쪽 참고

※ 항목구분 : 필수(1), 옵션(0)

c)

c) 응답 메시지 명세

항목명(영문)	항목명(국문)	항목크기	항목구분	샘플데이터	항목설명
resultCode	결과코드	2	1	00	결과코드
resultMsg	결과메시지	50	1	NORMAL SERVICE	결과메시지
numOfRows	한 페이지 결과 수	4	1	100	한 페이지 결과 수
pageNo	페이지 번호	4	1	1	페이지 번호
totalCount	전체 결과 수	4	1	40	전체 결과 수
items	목록	-	0..n	-	목록
stationName	측정소명	30	1	증구	측정소명
mangName	측정망 정보	10	1	도시대기	측정망 정보 (도시대기, 도로변대기, 국가배출농도, 교외대기, 항만)
sidoName	시도명	10	1	서울	시도 이름 (서울, 부산, 대구, 인천, 광주, 대전, 울산, 경기, 강원, 충북, 충남, 전북, 전남, 경북, 경남, 제주, 세종)
dateTime	측정일시	20	1	2020-11-25 11:00	오염도 측정 연-월-일 시간 : 분
so2Value	아황산가스 농도	10	1	0.007	아황산가스 농도(단위 : ppm)
coValue	일산화탄소 농도	10	1	0.7	일산화탄소 농도(단위 : ppm)
o3Value	오존 농도	10	1	0.043	오존 농도(단위 : ppm)
no2Value	이산화질소 농도	10	1	0.043	이산화질소 농도(단위 : ppm)
pm10Value	미세먼지(PM <sub>10</sub> ) 농도	10	1	68	미세먼지(PM <sub>10</sub> ) 농도 (단위 : $\mu\text{g}/\text{m}^3$ )
pm10Value24	미세먼지(PM <sub>10</sub> ) 24시간예측이동농도	10	1	56	미세먼지(PM <sub>10</sub> ) 24시간예측이동농도(단위 : $\mu\text{g}/\text{m}^3$ )
pm25Value	미세먼지(PM <sub>2.5</sub> ) 농도	10	1	39	미세먼지(PM <sub>2.5</sub> ) 농도 (단위 : $\mu\text{g}/\text{m}^3$ )

## Expected results

석모리	PM10: 54 ug/m3, PM2.5: 47 ug/m3
덕적도	PM10: 49 ug/m3, PM2.5: 29 ug/m3
백령도	PM10: 52 ug/m3, PM2.5: 32 ug/m3
영흥도	PM10: 62 ug/m3, PM2.5: 49 ug/m3
연평도	PM10: 47 ug/m3, PM2.5: - ug/m3
울도	PM10: 46 ug/m3, PM2.5: 37 ug/m3
신흥	PM10: 73 ug/m3, PM2.5: 58 ug/m3
서해	PM10: 73 ug/m3, PM2.5: 49 ug/m3
영종	PM10: 75 ug/m3, PM2.5: 52 ug/m3
송림동	PM10: 64 ug/m3, PM2.5: 46 ug/m3
구월동	PM10: 73 ug/m3, PM2.5: 67 ug/m3
송의	PM10: 62 ug/m3, PM2.5: 52 ug/m3
석바위	PM10: 60 ug/m3, PM2.5: 43 ug/m3
부평역	PM10: 62 ug/m3, PM2.5: 47 ug/m3
남동	PM10: 68 ug/m3, PM2.5: 59 ug/m3
주안	PM10: 56 ug/m3, PM2.5: 45 ug/m3
부평	PM10: 52 ug/m3, PM2.5: 42 ug/m3
연희	PM10: 42 ug/m3, PM2.5: 37 ug/m3
검단	PM10: 67 ug/m3, PM2.5: 46 ug/m3
중봉	PM10: 66 ug/m3, PM2.5: 46 ug/m3
계산	PM10: 59 ug/m3, PM2.5: 36 ug/m3
효성	PM10: - ug/m3, PM2.5: 51 ug/m3
고잔	PM10: 59 ug/m3, PM2.5: 38 ug/m3
서창	PM10: 56 ug/m3, PM2.5: 40 ug/m3
석남	PM10: 51 ug/m3, PM2.5: 46 ug/m3
송해	PM10: 44 ug/m3, PM2.5: 35 ug/m3
동춘	PM10: 71 ug/m3, PM2.5: 55 ug/m3
운서	PM10: 59 ug/m3, PM2.5: 52 ug/m3
송현	PM10: 65 ug/m3, PM2.5: 46 ug/m3
논현	PM10: 61 ug/m3, PM2.5: 40 ug/m3
청라	PM10: 57 ug/m3, PM2.5: 40 ug/m3
송도	PM10: 64 ug/m3, PM2.5: 44 ug/m3
아암	PM10: 58 ug/m3, PM2.5: 57 ug/m3

# Requesting from Python Code

```
import requests
import json

def getAirQualityByCity(city, key):

    endpoint = "http://apis.data.go.kr/B552584/ArpltnInforInquireSvc/getCtprvnRltmMesureDnsty"
    url = f"{endpoint}?sidoName={city}&pageNo=1&returnType=json&numOfRows=100&serviceKey={key}&ver=1.0"

    res = requests.get(url)

    return res
```

# Parsing your Response

```
if __name__ == '__main__':  
    city = "인천"  
    key = "e1RLZ9y[REDACTED]2BtX"  
  
    response = getAirQualityByCity(city, key)  
  
    if (response.status_code == 200):  
        response_body = response.json()['response']['body']  
        for item in response_body['items']:  
            print(f"{item['stationName']}\tPM10: {item['pm10Value']} ug/m3")  
    else:  
        print(f'Error code: {response}')
```



# Question?

