Open Web API

Introduction to Internet and Web

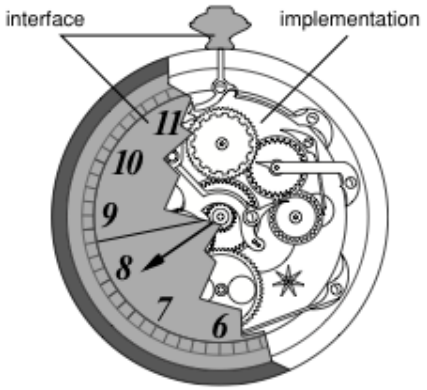
부산대학교 정보·의생명 공학대학
정보컴퓨터공학부

Contents

- ❖ Open Web API
- ❖ REST
- ❖ JSON/XML Introduction

OPEN WEB API

API

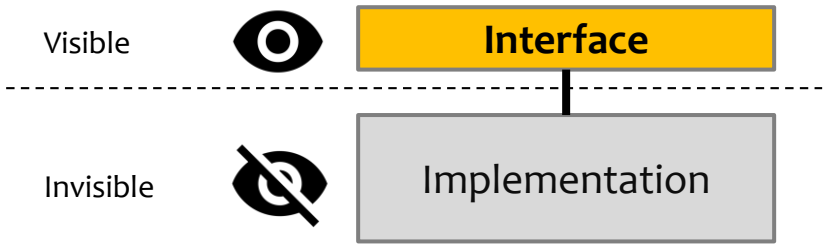


❖ What is an API ?

- API : Application Programming Interface
- What is an API ? – YouTube (#2)

❖ Separation of Interface from Implementation

- 기능/서비스를 제공하는 객체는 Interface와 Implementation으로 구성
- 기능/서비스를 이용하기 위해서는 Interface에 대한 이해만 있으면 충분함
- 예) 자동차와 운전
  - 가솔린, 디젤, 하이브리드, 순수 전기차 등 자동차 엔진들의 동작 원리와 차이점에 대해 이해하지 못해도 우리는 운전 핸들, 가속/브레이크 페달의 기능과 조작 방법만 이해하면 자동차를 운전할 수 있음
  - 자동차 엔진 : Implementation; 핸들, 가속/브레이크 페달 : Interface
- **Abstraction** in Computer Science : Separation of Interface from Implementation
  - Interface와 Implementation을 구분하는 것, Implementation에 독립적(Independent)인 Interface를 설계하는 것, 구현의 세부 내용을 숨기는 것 (Hiding the details of Implementation) S/W Engineering의 기본.



Interface vs. Implementation

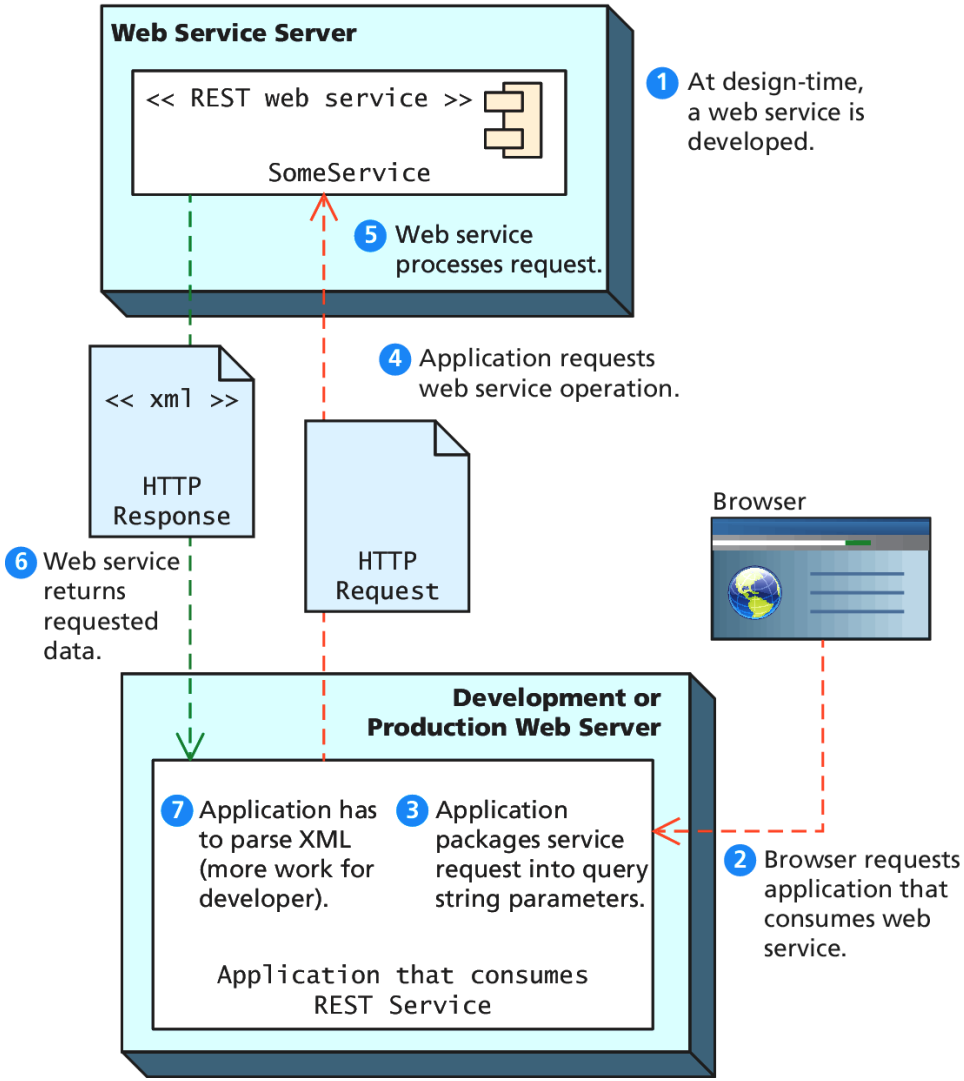
가솔린 엔진과
디젤 엔진 작동 원리의 차이

가솔린 엔진		디젤 엔진
▶ 가솔린+공기 흡입	흡입	▶ 공기 흡입
▶ 가솔린과 공기를 10분의 1로 압축	압축	▶ 흡입한 공기를 20분의 1로 압축
▶ 압축한 가솔린과 공기를 전기불꽃으로 점화	팽창	▶ 압축된 공기에 경유를 분사해 자연 발화
연소가스 배출	배기	▶ 연소가스 배출

Website vs. Web Services

A website has a user interface or **GUI**.	A web service **doesn't have a user interface**
Websites are meant for **use by humans**.	Web services are meant **for other applications** to be interacted with over the internet.
Websites are **accessed by using their GUI components** – buttons, text boxes, forms, etc.	Web services are accessed by **HTTP methods / Web API** – GET, POST, PUT, DELETE, etc.
Websites are cross-platform as they require tweaking to operate on different browsers, operating systems, etc.	Web services are platform independent as they use open protocols
e.g. naver.com is a website that has a collection of related web pages containing tutorials.	e.g. Google maps API is a web service that can be used by websites to display Maps by passing coordinates to it.

Web Service


Web Service Server
<< REST web service >>
SomeService
1 At design-time, a web service is developed.
5 Web service processes request.
4 Application requests web service operation.
<< xml >>
HTTP Response
HTTP Request
6 Web service returns requested data.
Browser
Development or Production Web Server
7 Application has to parse XML (more work for developer).
3 Application packages service request into query string parameters.
Application that consumes REST Service
2 Browser requests application that consumes web service.

Open Web API

❖ Open API / Public API

- A **publicly available** API that provides developers with programmatic access to a proprietary software application or web service – Open API from Wikipedia.

❖ Web API

- An API for either a web server or a web browser - Web API from Wikipedia.

❖ Examples

- 대한민국 정부 공공데이터 API : www.data.go.kr
- KAKAO, NAVER Search API
- Google Map API
- Philips HUE API

Open Web API

❖ 공공데이터 예제: link

오픈API 상세

URL 복사

XML 부산광역시_부산버스정보시스템

활용신청

입력 파라미터를 이용하여 부산 버스도착정보 목록 및 상세정보를 조회 할 수 있는 서비스
부산시 버스정보시스템에 대한 정류소 정보, 노선 정보, 정류장 버스 도착예정정보를 제공

2 0 관심

오류신고 및 담당자 문의

OpenAPI 정보 메타데이터 다운로드

분류체계	교통및물류 - 도로	제공기관	부산광역시
관리부서명	대중교통과	관리부서 전화번호	051-888-3964
API 유형	REST	데이터포맷	XML
활용신청	184	키워드	부산,버스,대중교통
등록	2021-10-26	수정	2021-10-26
심의유형	개발단계 : 허용 / 운영단계 : 허용		
비용부과유무	무료		
이용허락범위	이용허락범위 제한 없음		
참고문서	OpenAPI활용가이드 부산버스정보시스템 v2.0.docx		

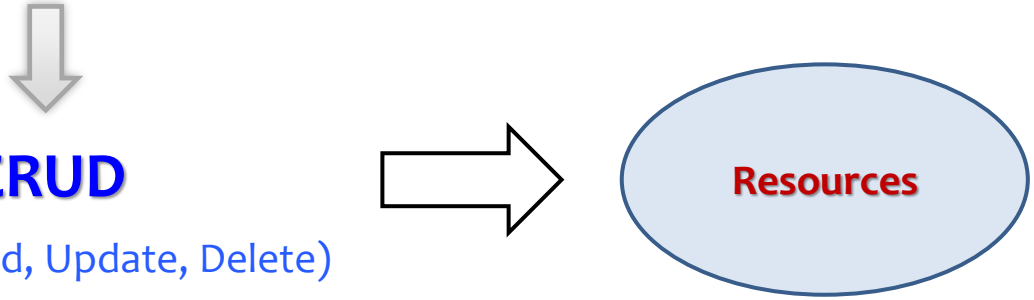
REST

REST ?

❖ **Representational State Transfer**

❖ One way of providing interoperability between computer systems on the Internet.

- REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.



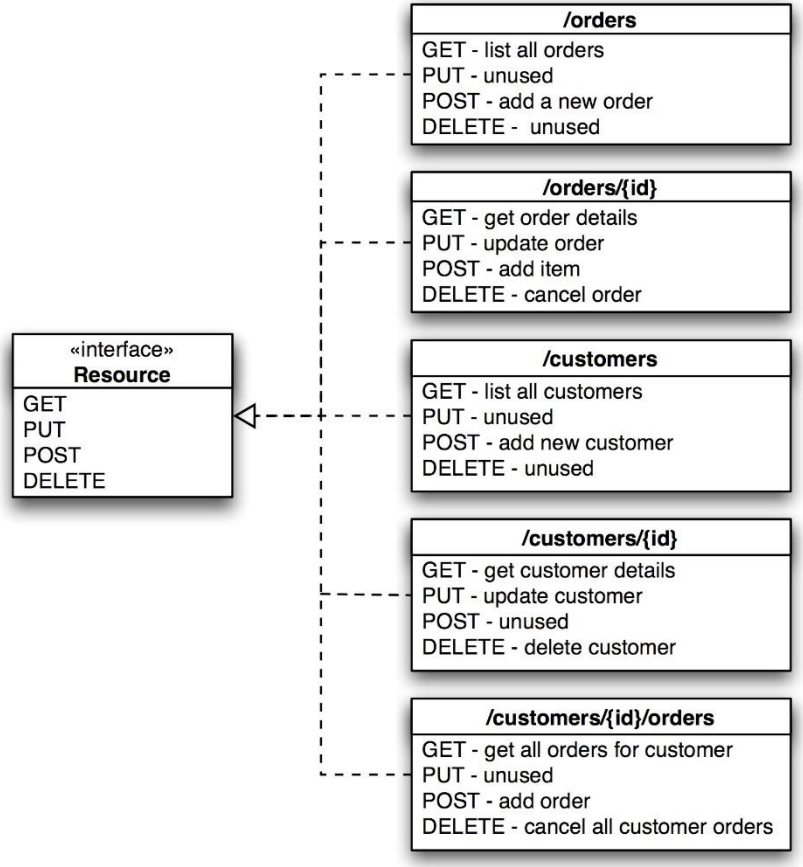
❖ HTTP & REST

- Any web service that is defined on the principles of REST can be called a RestFul web service. A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components.
- CREATE – HTTP POST, **READ – HTTP GET, UPDATE – HTTP PUT**, DELETE – HTTP DELETE

REST example

❖ REST – Simple and Easy to Learn

- Things are identified by URLs
  - Consists of nouns
  - Customer, orders
- HTTP Verb to dictate the operation on that resource
- Multiple URLs pointing to same resource



REST – Resource Oriented !!!

❖ It's not a protocol, it's an architectural approach.

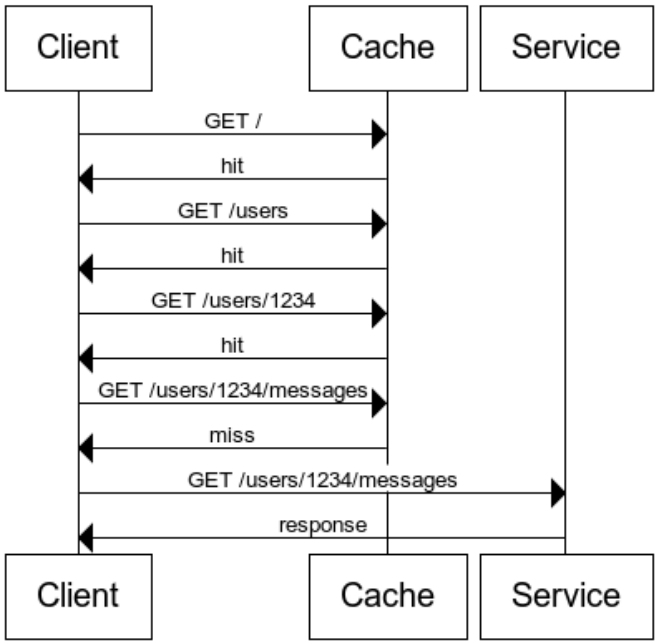
- Can be used with legacy XML or modern JSON information transfer format

❖ It's a Guidelines

- HTTP methods and corresponding CRUD (Create, Read, Update, Delete) operation, recommendation about URL design.

❖ Architectural Constraints

- Client-Server
- Stateless
- Cacheable
- Uniform Interface
  - Identification of resources : ex) URL
    - Directory like resource structure, Use proper MIME types
  - CRUD Operation: Create, Read, Update, Delete
    - Use HTTP methods for CRUD operations
- Layered system
- Code on demand (optional)



https://blog.ploeh.dk/2014/01/20/rest-efficiency/

공공데이터 예제

❖ 참조: OpenAPI활용가이드 부산버스정보시스템 v2.0.docx

- https://www.data.go.kr/data/15092750/openapi.do

❖ http://apis.data.go.kr/6260000/BusanBIMS/busStopList?bstopnm=부산시청&arsno=13708&pageNo=1&numOfRows=10&serviceKey=서비스키

- 서비스 URL: http://apis.data.go.kr/6260000/BusanBIMS
  - 부산버스도착정보 조회 서비스
- 상세기능명: busStopList
  - 정류소정보 조회
- 요청 메시지 명세:
  bstopnm=부산시청&arsno=13708&pageNo=1&numOfRows=10&serviceKey=서비스키
  - bstopnm: 정류소 명, arsno: 정류소 번호, pageNo: 페이지 번호, numbOfRows: 한페이지 결과 수, serviceKey: 인증 키

XML AND JSON INTRODUCTION

XML Syntax

❖ **XML prolog is optional**

- 예시:

```
<?xml version="1.0" encoding="UTF-8"?>
```

- If it exits, it must come first in the document.

❖ **All XML elements have a closing tag**

❖ **XML tags are case sensitive**

❖ **XML elements must be properly nested**

❖ **XML attribute values must always be quoted**

```
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

http://apis.data.go.kr/6260000/BusanBIMS/busStopList?bstopnm=부산시청&arsno=13708&pageNo=1&numOfRows=10&serviceKey=서비스키

```
<response>
    <header>
        <resultCode>00</resultCode>
        <resultMsg>NORMAL SERVICE.</resultMsg>
    </header>
    <body>
        <items>
            <item>
                <bstopid>505780000</bstopid>
                <bstopnm>부산시청</bstopnm>
                <arsno>13708</arsno>
                <gpsx>129.07691415917</gpsx>
                <gpsy>35.179937855685</gpsy>
                <stoptype>일반</stoptype>
            </item>
        </items>
        <numOfRows>10</numOfRows>
        <pageNo>1</pageNo>
        <totalCount>2</totalCount>
    </body>
</response>
```

JSON Syntax

❖ **JSON: JavaScript Object Notation.**

❖ **JSON is a syntax for storing and exchanging data.**

❖ **JSON Syntax Rules**

- Data is in name/value pairs
  - string value: double quotes
- Data is separated by commas(,)
- Curly braces({}) hold objects
- Square brackets([]) hold arrays

```
{ "name":"John" }
{ "age":30 }
{ "sale":true }
{ "middlename":null }
```

```
{
"employee":{ "name":"John", "age":30, "city":"New York" }
}
{
"employees":[ "John", "Anna", "Peter" ]
}
```

JSON vs XML

- JSON doesn't use end tag, JSON is shorter
- XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

```json
{
    "response": {
        "header": {
            "resultCode": "00",
            "resultMsg": "NORMAL_SERVICE"
        },
        "body": {
            "dataType": "JSON",
            "items": {
                "item": [
                    {
                        "baseDate": "20200510",
                        "baseTime": "1800",
                        "category": "PTY",
                        "nx": 100,
                        "ny": 75,
                        "obsrValue": "0"
                    },
                    {
                        "baseDate": "20200510",
                        "baseTime": "1800",
                        "category": "REH",
                        "nx": 100,
                        "ny": 75,
                        "obsrValue": "86"
                    },
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<response>
    <header>
        <resultCode>00</resultCode>
        <resultMsg>NORMAL_SERVICE</resultMsg>
    </header>
    <body>
        <dataType>XML</dataType>
        <items>
            <item>
                <baseDate>20200510</baseDate>
                <baseTime>1800</baseTime>
                <category>PTY</category>
                <nx>100</nx>
                <ny>75</ny>
                <obsrValue>0</obsrValue>
            </item>
            <item>
                <baseDate>20200510</baseDate>
                <baseTime>1800</baseTime>
                <category>REH</category>
                <nx>100</nx>
                <ny>75</ny>
                <obsrValue>86</obsrValue>
            </item>
```

요 약

- Open Web API
- REST
- JSON/XML Introduction