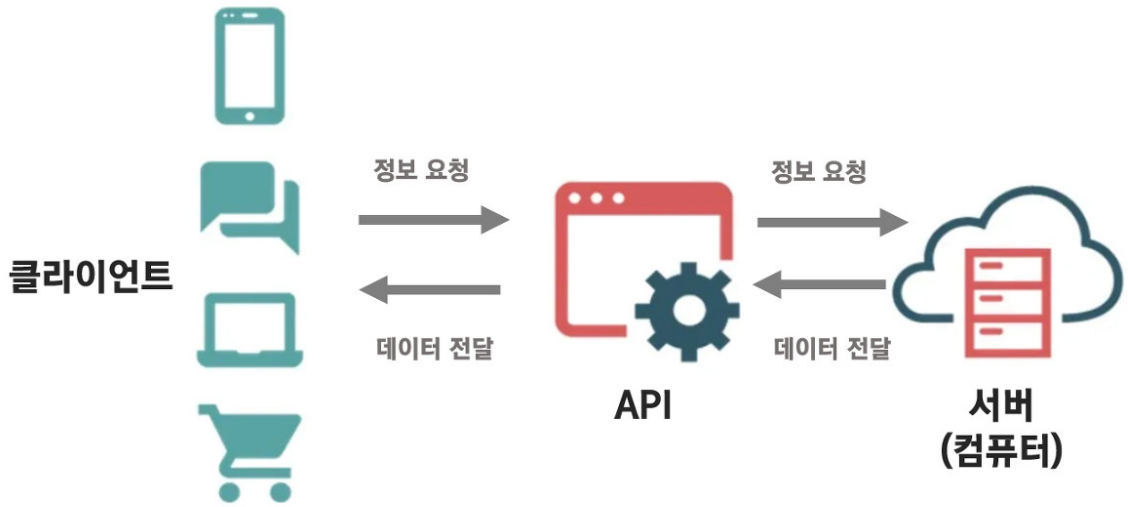
REST API

1.API란?



⇒ API(Application Programming Interface)로 서로 다른 소프트웨어 애플리케이션 간에 통신하고 상호작용할 수 있도록 돕는 중간 역활을 하여 한 프로그램이 다른 프로그램의 기능을 호출하거나 데이터를 요청할 수 있는 일종의 규칙이나 프로토콜이다.

⇒ 이를 통해 다양한 기기 및 어플리케이션간 통신을 한다.

다음과 같은 장점을 제공

- 추상화
  - 복잡한 내부 로직을 간단한 함수, 명령어를 통해 쉽게 사용할 수 있게 함

    - 내부 동작 원리를 몰라도 기능을 호출하여 사용을 편리하게 함
- 인터페이스
    - 서로 다른 기기 및 어플리케이션간 기능을 요청할 수 있는 접점
- 표준화된 통신방식
    - 규칙을 통한 설계로 어떤 형태로 요청하고 응답을 받을 지 알기 때문에 표준화 되어 같은 규격을 사용하는 기기 및 어플리케이션간 통신 및 개발에 용이

2.REST API란?

⇒ REST(Representational State Transfer)로 REST 원칙을 따르는 API이다.

⇒ 웹에서 자원을 REST 원칙을 이용하여 효율적으로 접근, 조작할 수 있는 아키텍쳐 스타일

3.REST는 왜 등장하게 되었을까?

⇒ 다양한 기기 간 통신에서 사용되는 API들에 일정한 **규칙**과 **원칙**을 적용하여, 웹 표준을 기반으로 **멀티 플랫폼에서 쉽게 적용**될 수 있는 API를 만들기 위해 등장하였다.

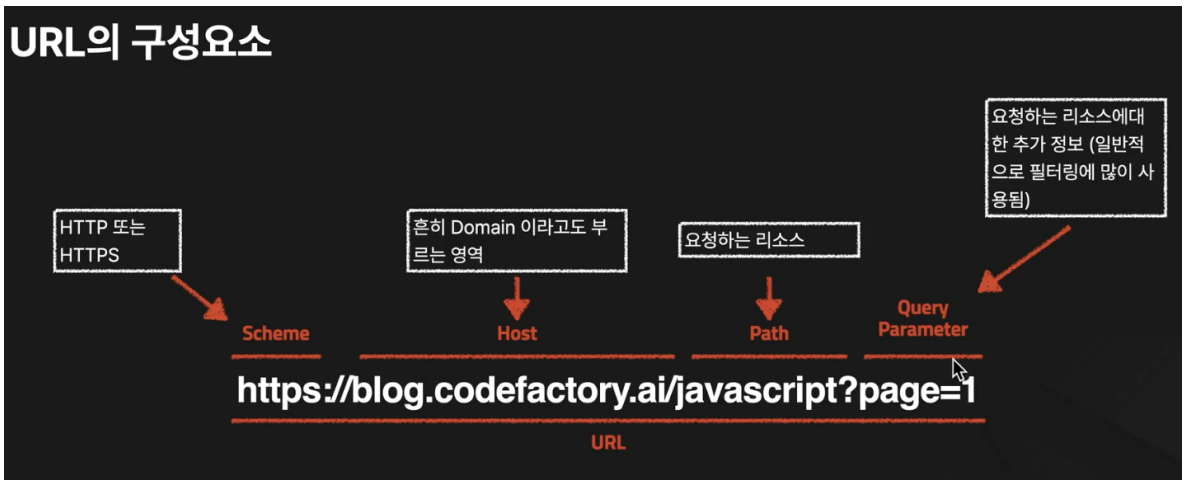
https://img1.daumcdn.net/thumb/R1280x0/?scode=mtistory2&fname=https://blog.kakaocdn.net/dn/cFl8Nb/btsc3zhNSfW/kW0R87obCIloLnS3PI6B5k/img.png

- 클라이언트-서버(멀티 플랫폼) 통신에서 대부분 REST API 원칙을 이용하여 통신한다.

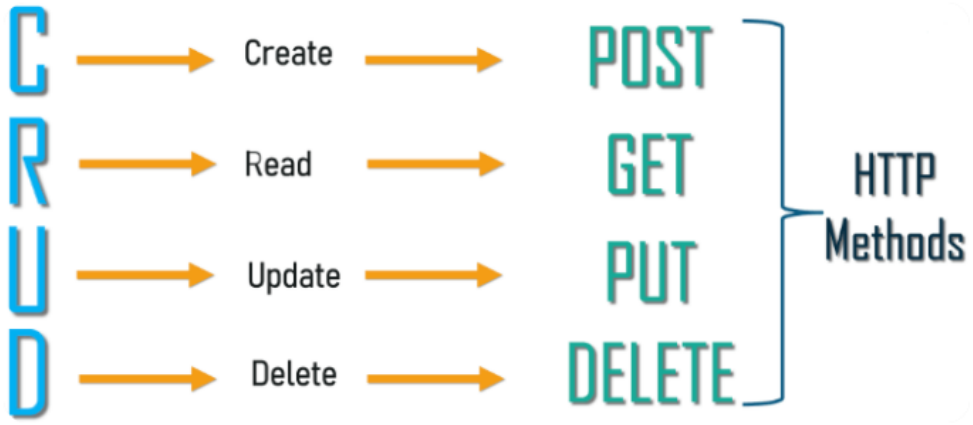
⇒ REST API는 HTTP 프로토콜을 기반으로 동작하며, 이를 통해 웹 애플리케이션, 모바일 앱, 클라우드 서비스 등 다양한 환경에서 효율적이고 확장성 있는 통신을 구현한다.

4.REST API의 주요개념

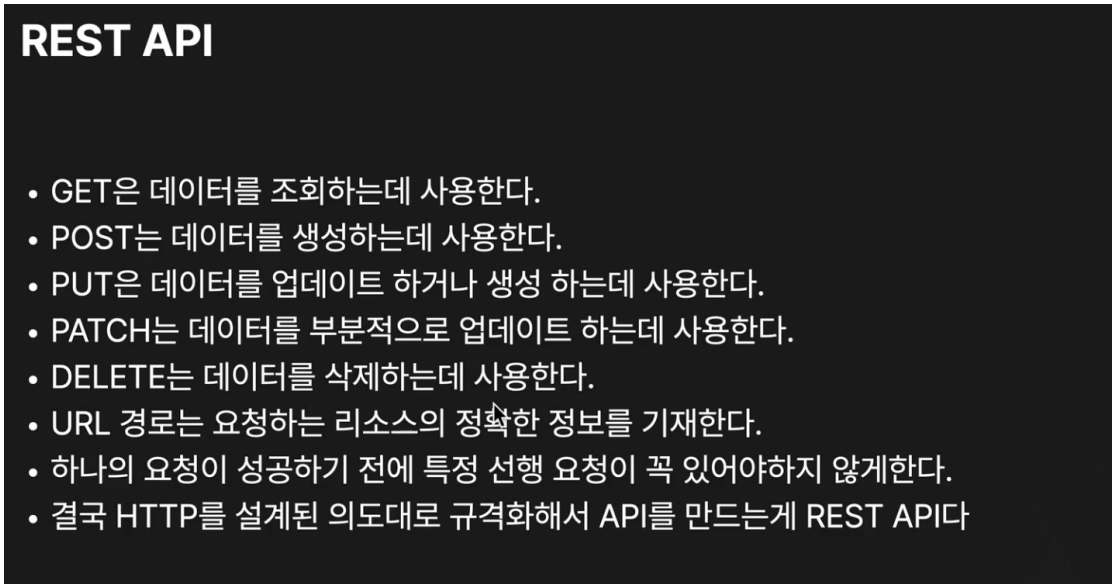
1. 자원(Resource) 기반 구조
    a. 고유한 HTTP URL(Uniform Resource Identifier)을 이용하여 자원을 관리
    b. 이것을 이용한 CRUD(Create, Read, Update, Delete)기능 제공



2. HTTP Method을 활용한 자원 조작
    a. POST, GET, PUT, DELETE, PATCH 등의 메서드를 활용한다.
    b. 이것을 이용하여 CRUD Operation을 제공한다.
    c. 각 기능에 알맞는 메서드가 있다.



- update엔 patch도 들어간다.



3. 무상태성(Stateless)

- **요청 독립성**: 각 요청은 완전히 독립적이라 클라이언트가 서버에 요청을 보낼 때 해당 작업을 수행하는 데 필요한 모든 정보가 포함되어야 한다.
- **서버 확장성**: 서버는 이전 요청의 상태를 저장할 필요가 없기 때문에 서버의 부하를 줄일 수있고 쉽게 확장할 수 있다. ⇒ 서버에 많은 사용자가 접근해도 상태 정보를 관리할 필요가 없기 때문에 확장성이 올라간다.
- **유연한 처리**: 클라이언트가 요청을 보낼 때마다 무상태이기 때문에 반드시 같은 서버에 연결될 필요가 없어서 여러 서버에 로드 밸런싱(load

balancing)을 적용할 수 있는 유연성을 제공한다.

4. URL을 통한 자원 식별

5. 작성 규칙

1. 소문자를 사용한다.
2. 언더바(_) 대신 하이픈(-)을 사용한다.
3. 마지막에 슬래시(/)를 포함하지 않는다.
4. 행위를 포함하지 않는다.

5. 파일 확장자는 URL에 포함시키지 않는다.

6. 전달하고자 하는 명사를 사용하되, 컨트롤 자원을 의미하는 경우 예외적으로 동사를 사용한다.

7. URI에 작성되는 영어를 복수형으로 작성한다.

8. URI 사이에 연관 관계가 있는 경우 /리소스/고유ID/관계 있는 리소스 순으로 작성한다.

동작 예시

▼ **GET 요청 (자원 조회)**

⇒ 사용자가 ID가 123인 사용자의 정보를 조회하는 요청, **GET** 메서드를 사용하며 URI는 자원을 가리키며, **페이로드**는 없다.(페이로드는 조회할 때 필요하지 않음).

<request>

```
GET /users/123 HTTP/1.1
Host: example.com
```

- **자원(Resource)**: /users/123 (ID가 123인 사용자)
- **HTTP 메서드(Method)**: GET (자원 조회)
- **페이로드(Representations)**: 없음

<response>

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": 123,
  "name": "John Doe",
```

```
  "email": "john@example.com"
}
```

▼ **POST 요청 (새로운 자원 생성)**

⇒ 사용자가 새로운 사용자를 생성하는 요청이며 이때는 POST 메서드를 사용하고, 자원의 내용은 페이로드로 전달된다.

<request>

```
POST /users HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "Jane Doe",
  "email": "jane@example.com"
}
```

- **자원(Resource)**: /users (사용자 목록에 새 사용자 추가)
- **HTTP 메서드(Method)**: POST (자원 생성)
- **페이로드(Representations)**: 새 사용자에 대한 정보(JSON 형식)

<responsse>

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "id": 124,
  "name": "Jane Doe",
  "email": "jane@example.com"
}
```

▼ **PUT 요청 (자원 수정)**

⇒ 사용자가 ID가 123인 사용자의 정보를 업데이트하는 요청이며 PUT 메서드를 사용하여 자원의 내용을 수정한다.

<request>

```
PUT /users/123 HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "John Doe Jr.",
  "email": "johnjr@example.com"
}
```

- **자원(Resource)**: /users/123 (ID가 123인 사용자)
- **HTTP 메서드(Method)**: PUT (자원 수정)
- **페이로드(Representations)**: 수정할 사용자 정보(JSON 형식)

<response>

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": 123,
  "name": "John Doe Jr.",
  "email": "johnjr@example.com"
}
```

▼ **DELETE 요청 (자원 삭제)**

⇒ 사용자가 ID가 123인 사용자를 삭제하는 요청이며 DELETE 메서드를 사용한다.

<request>

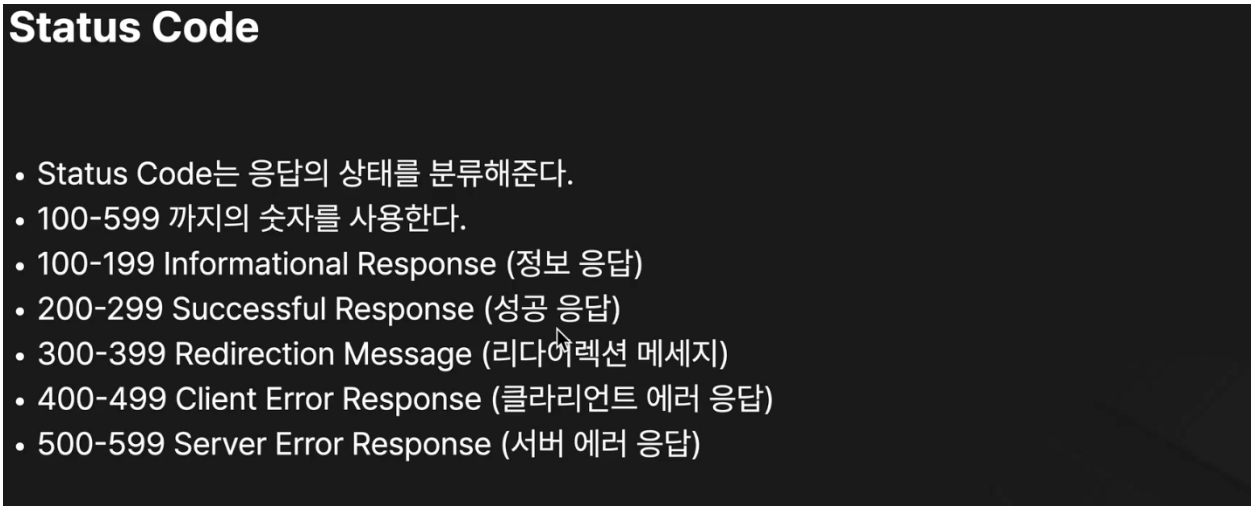
```
DELETE /users/123 HTTP/1.1
Host: example.com
```

- **자원(Resource)**: /users/123 (ID가 123인 사용자)
- **HTTP 메서드(Method)**: DELETE (자원 삭제)
- **페이로드(Representations)**: 없음

<response>

```
HTTP/1.1 204 No Content
```

상태 코드



주요 Status Code 정리

- 200 (OK) - 문제없이 요청이 잘 실행됨
- 201 (Created) - 문제없이 데이터 생성이 잘 됨 (POST 요청에서 많이 사용)
- 301 (Moved Permanently) - 리소스가 영구적으로 이동됨
- 400 (Bad Request) - 요청이 잘못됨 (필수 값 부족 등)
- 401 (Unauthorized) - 인증 토큰/키가 잘못됨
- 403 (Forbidden) - 접근 불가능한 리소스. 401과 달리 인증은 된 상태
- 404 (Not Found) - 존재하지 않는 리소스.
- 405 (Method Not Allowed) - 허가되지 않은 요청 Method
- 500 (Internal Server Error) - 알 수 없는 서버 에러