

---

# 프론트, 백엔드, API

서종원

---

# Frontend & Backend

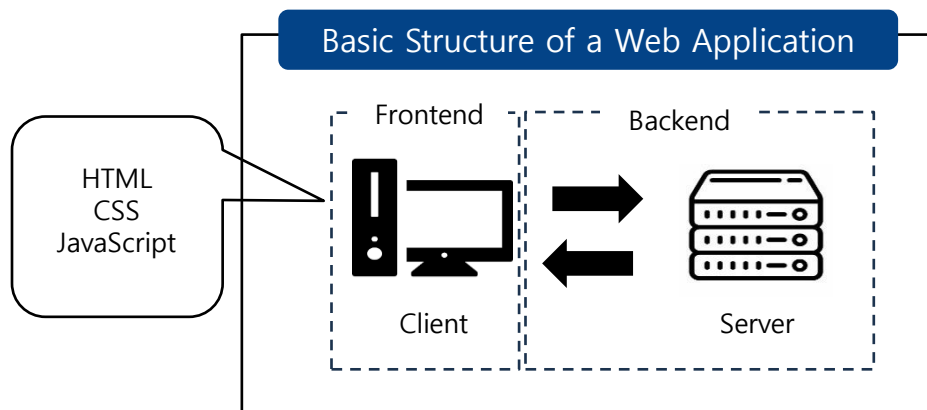
클라이언트 (Client), 서버 (Server)는 역할에 따른 구분을 의미하며,  
프론트엔드 (Frontend), 백엔드 (Backend)는 개발 영역에 따른 구분을 의미함

## Client (Frontend)

- 정의: 서비스를 요청하는 주체로 일반적으로 최종 사용자가 사용하는 기기 또는 애플리케이션을 의미함
- 역할: 서버에 데이터를 요청하거나 서버로부터 정보를 수신하여 사용자에게 표시함
- Frontend 범위: 웹페이지 레이아웃, 디자인, 대화형 버튼, 입력 형태와 같은 UI 요소 개발이 포함됨
- HTML, CSS

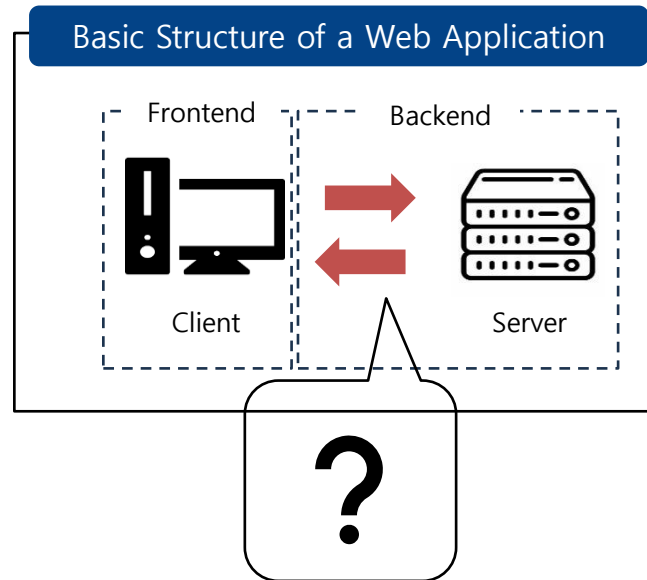
## Server (Backend)

- 정의: 클라이언트의 요청을 처리하고 데이터를 제공함
- 역할: 클라이언트의 요청을 처리하며, 데이터 베이스에서 필요한 데이터를 검색하거나 요청을 처리하여 클라이언트에 반환함
- Backend 범위: 데이터 처리, 데이터베이스 상호 작용, 인증 및 검증을 포함하며, Frontend 요구 사항을 반환함
- Node.js, DB



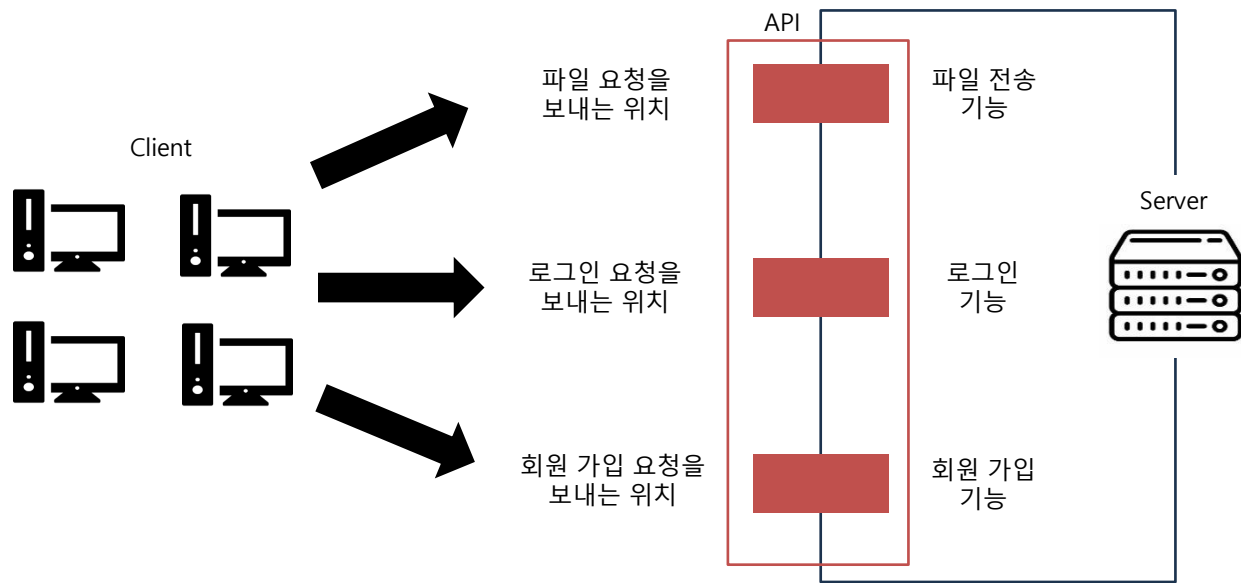
# Frontend & Backend

Frontend, Backend는 어떻게 서로 연결되고 통신할까?



# API (Application Programming Interface)

- 인터페이스: 서로 다른 시스템 간의 통신을 가능하게 하는 규칙
- API 는 서로 다른 애플리케이션 또는 서비스 간에 통신할 수 있는 방법을 제공함.
  - 예를 들어, 웹 애플리케이션이 데이터를 요청할 때 API는 클라이언트와 서버 사이의 다리 역할을 하여 각 시스템이 이해할 수 있는 형식으로 데이터를 교환 할 수 있도록 지원함



# API 용도별 분류

## Open API (Public API)

- 모든 사람에게 개방된 API
- 최소한의 제한; 무료인 경우가 많거나 유료 옵션이 포함된 부분 유료화 모델이 포함
- 기업이나 서비스가 외부 개발자와 협업하여 생태계를 확장 가능

## Internal API (Private API)

- 특정 조직 또는 회사 내부에서 사용하기 위해 만든 API
- 공개적으로 접근할 수 없으며 내부 시스템 내에서 데이터 통신에 사용됨
- 보안성이 높고 내부 시스템에 최적화되어 있음

## Partner API

- 특정 파트너 또는 공동 작업자만 액세스 할 수 있는 API
- 주로 비즈니스 파트너 간의 데이터 통합에 사용되며, 제한된 접근 권한
- 제한된 접근을 통해 보안을 강화하면서 효율적인 데이터 공유를 지원

## Composite API

- 여러 API 요청을 단일 요청으로 묶어놓은 API
- 여러 작업을 동시에 수행할 수 있어 API 요청 횟수를 줄이고 복잡한 데이터 처리를 간소화 할 수 있음
- 네트워크 트래픽을 줄이고 여러 개의 자원의 통합을 통해 데이터에 빠르게 접근을 지원

# API 설계에 따른 분류

## REST (Representational State Transfer) API

- HTTP를 기반으로 널리 사용되는 웹 API
- RESTful API는 REST API의 일종으로 HTTP 메서드 (GET, POST, PUT, DELETE)를 사용하여 리소스를 관리함
- 이해하기 쉽고, 확장성이 뛰어나며 다양한 웹 서비스와 호환됨
- Google Maps

## SOAP (Simple Object Access Protocol) API

- XML 기반의 API
- 엄격한 XML 표준을 사용하여 보안과 안정성을 보장함
- 은행, 금융 기관, 정부 기관 등 보안에 민감한 시스템에서 일반적으로 사용됨
- eBay, PayPal

## GraphQL API

- 필요한 데이터만 선택적으로 요청하는 기능이 포함된 Facebook에서 개발한 API
- 클라이언트는 필요한 데이터만 검색할 수 있어 데이터 전송을 최적화할 수 있음
- 데이터 전송량을 줄일 수 있는 것으로 알려져 있음

## gRPC (Google Remote Procedure Call)

- 구글에 의해 개발된 API
- 높은 속도와 짧은 지연 시간을 제공함
- 실시간 커뮤니케이션 환경에 적합함

## HTTP 메서드

### **GET** – 데이터 조회

- 서버 리소스를 읽어오기만 함

### **POST** – 데이터 생성

- 서버에 새 데이터 추가

### **PUT** – 데이터 수정 (전체)

- 리소스 교체

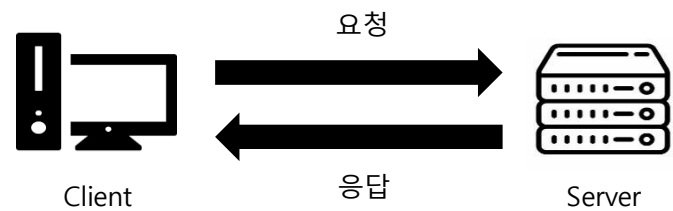
### **PATCH** – 데이터 수정 (부분 수정)

- 리소스 교체

### **DELETE** – 삭제

- 리소스 삭제

# API 요청 & 응답 (REST API)



## 요청

- Endpoint
  - HTTP Methods
  - HTTP Headers
  - Body
- 구조

https://api.example.com/products/123

GET

Content-Type: application/json  
Authorization: Author Token

```
{  
  "product" : 123  
}
```

예시

## 응답

- HTTP Status Codes
  - HTTP Headers
  - Body
- 구조

200 OK

Content-Type: application/json  
Date: Tue, 17 Oct 2023 12:34:56 GMT

```
{  
  "id": 123, "name": "Wireless Bluetooth Speaker",  
  "price": 49.99, "stock": 120, "description": "A compact  
  and powerful wireless speaker with Bluetooth 5.0  
  support."  
}
```

예시



# API 요청 (REST API): Endpoint, HTTP Method

- 서버에게 특정 작업을 수행하도록 요청하는 클라이언트의 메시지
- 요청은 Endpoint, HTTP Method, HTTP Headers, Body 구성된 특정 형식으로 서버로 전송됨

## Endpoint

- 특정 API 요청을 처리하도록 설정된 URL 주소
- 클라이언트가 특정 기능을 수행하기 위해 서버에 필요한 데이터를 요청하는 주소
- 하나의 API에는 각각 다른 기능을 수행하는 여러 개의 Endpoint가 있을 수 있음
- Endpoint를 통해 클라이언트는 특정 데이터를 요청할 수 있음
- https://api.example.com/users/123 유저 정보를 불러오는 API Endpoint가 될 수 있음  
https://api.example.com/orders/456 주문 정보를 불러오는 API Endpoint가 될 수 있음
- URL 구조는 일반적으로 도메인 + 경로 구성됨

## HTTP Method

- 요청의 유형을 정의함  
일반적인 HTTP methods 는 GET, POST, PUT, DELETE
- **GET**: 서버에서 정보를 읽음. 주로 서버의 상태를 변경하지 않고 데이터를 읽는데 사용됨
- **POST**: 새 데이터를 만들거나 서버로 전송하는데 사용됨. 데이터베이스에 새 항목을 추가하는 등 일반적으로 서버의 상태를 변경하는데 사용됨
- **PUT**: 기존 데이터에 대한 수정을 요구함
- **DELETE**: 서버에서 특정 데이터 삭제를 요청할 때 사용됨

# API 요청 (REST API): HTTP Headers, Body

- 서버에게 특정 작업을 수행하도록 요청하는 클라이언트의 메시지
- 요청은 Endpoint, HTTP Method, HTTP Headers, Body 구성된 특정 형식으로 서버로 전송됨

## HTTP Headers

- 요청에 대한 메타 데이터를 포함함:  
인증 세부 정보, 데이터 형식을 포함
- **Authentication:**  
클라이언트가 서버에 접근할 수 있는 권한이 있는 경우 API 인증 포함
- **Content-type:**  
요청 데이터 형식을 지정하여 서버가 예상하는 데이터 형식을 나타냄
  - application/json
  - application/xml
  - text/html
  - text/plain

## Body

- 요청의 일부로 서버로 전송되는 필수 데이터를 포함
- 서버가 처리해야 하는 특정 데이터를 포함
- 일반적으로 JSON, XML 과 같은 형식으로 작성됨

# API 요청 (REST API): HTTP Headers (Content-type)

## application/json

- 가장 많이 사용되는 형식으로 대부분의 RESTful API의 표준으로 사용됨

```
{  
  "name": "John",  
  "age": 30  
}
```

## application/xml

- XML (Extensible Markup Language) 형식의 데이터를 나타냄

```
<person>  
  <name>John</name>  
>  
  <age>30</age>  
</person>
```

## application/html

- 데이터를 HTML 형식으로 나타냄.

```
<html>  
  <body>  
    <h1>Hello, world!</h1>  
  </body>  
</html>
```

## application/plain

- 단순한 텍스트 데이터 형식

"Hello World!"

# JSON & XML Format

## JSON (JavaScript Object Notation)

- 사람과 컴퓨터가 모두 읽기 쉬운 경량화된 형태로 효율적인 데이터 교환을 위해 설계됨
- 키-값 쌍으로 구성되며 중첩된 구조 또는 배열을 포함할 수 있음
- 단순한 구조로 빠른 분석이 가능하며 파일 크기가 작아 빠른 데이터 전송이 가능
- 명확한 데이터 구조로 읽기 쉽고 이해하기 쉽도록 설계됨
- 데이터 형식을 강제하지 않으며 데이터 유효성 검사가 어려움
- 메타 데이터를 포함하는 기능 없이 데이터 자체를 주로 표현함

```
{
  "name": "Alice",
  "age": 25,
  "isStudent": true,
  "courses": ["Math", "Computer Science"],
  "address": {
    "street": "123 Main St",
    "city": "Wonderland"
  }
}
```

## XML (Extensible Markup Language)

- 태그로 데이터를 구조화 하는데 사용되는 언어
- HTML과 유사하지만 데이터 전송 및 저장을 위해 특별히 설계됨
- 열과 닫는 태그로 구성된 트리 형식으로 데이터를 나타냄
- 각 요소는 태그 사이에 데이터를 배치하거나 속성으로 추가하여 중첩할 수 있음
- 데이터와 함께 추가 메타데이터를 포함할 수 있어 복잡한 데이터를 처리하는데 유용함
- JSON에 비해 파일 크기가 더 크고 전송 속도가 느림
- 데이터 형식 표준화 및 확장성이 중요한 금융, 의료 등의 산업에서 일반적으로 사용됨

```
<person>
  <name>Alice</name>
  <age>25</age>
  <isStudent>true</isStudent>
  <courses>
    <course>Math</course>
    <course>Computer Science</course>
  </courses>
  <address>
    <street>123 Main St</street>
    <city>Wonderland</city>
  </address>
</person>
```

# API Response (REST API): HTTP Status Code

- 클라이언트의 요청에 대한 응답으로 서버가 처리한 결과를 반환하는 메시지
- 서버가 요청을 완료한 후 결과 데이터를 반환하거나 클라이언트에게 요청 상태를 알리는데 사용됨
- HTTP Status Code, HTTP Headers, Body로 구성됨

## HTTP Status Code

- 3자리 코드: 응답의 성공 또는 상태를 표현함

5가지 주요 카테고리:

- **100번대:** 정보 응답  
서버가 요청을 처리 중임을 나타냄
- **200번대:** 성공적인 응답  
클라이언트의 요청이 성공적으로 처리되었음을 확인함
- **300번대:** 리디렉션 (Redirection)  
요청된 리소스가 다른 위치로 이동했으며 자동으로 새 위치로 옮겨질 수 있음을 나타냄
- **400번대:** 클라이언트 에러  
클라이언트가 잘못된 요청을 보냈음을 나타냄
- **500번대:** 서버 에러  
요청을 처리하는 동안 서버에서 오류가 발생했음을 나타냄

# API Response (REST API): HTTP Status Code Example

Status Code	Message	Description
100	Continue	서버가 클라이언트 요청의 일부를 수신 했으며 클라이언트가 나머지 요청을 계속 보낼 수 있는 상태를 의미함
200	OK	요청이 성공적으로 처리됨. 일반적으로 GET 요청에 대한 응답으로 사용됨
201	Created	요청이 성공적으로 처리되었으며 새 리소스가 생성되었음을 나타냄. 일반적으로 POST 요청에 대한 응답으로 사용됨
204	No Content	요청이 성공했지만 반환할 데이터가 없음
301	Moved Permanently	요청된 리소스가 영구적으로 새 위치로 이동했으며 클라이언트는 새 URL을 사용해야 함
400	Bad Request	요청 형식이 올바르지 않아 서버에서 이해할 수 없음
401	Unauthorized	인증이 필요하지만 클라이언트가 유효한 자격 증명을 제공하지 않음
403	Forbidden	클라이언트가 리소스에 접근할 수 있는 권한이 없음
404	Not Found	요청된 자원을 서버에서 찾을 수 없음
409	Conflict	요청이 현재 서버 상태와 충돌함
429	Too Many Request	클라이언트가 단기간에 너무 많은 요청을 보냄
500	Internal Server Error	서버에서 예기치 않은 오류가 발생함
501	Not Implemented	서버가 요청된 기능을 지원하지 않음
502	Bad Gateway	유효하지 않은 응답을 받음
504	Gateway Timeout	응답을 받지 못하여 시간 초과가 발생함

# API Response (REST API): HTTP Headers, Body

- 클라이언트의 요청에 대한 응답으로 서버가 처리한 결과를 반환하는 메시지
- 서버가 요청을 완료한 후 결과 데이터를 반환하거나 클라이언트에게 요청 상태를 알리는데 사용됨
- HTTP Status Code, HTTP Headers, Body로 구성됨

## HTTP Headers

- 응답에 대한 메타 데이터를 제공함
- **Content-Type:** 응답 본문의 형식을 지정함  
예시) *Content-Type: application/json*
- **Content-Length:** 응답의 크기를 바이트 단위로 지정함  
예시) *Content-Length: 1500*
- **Data:** 서버 시간을 기준으로 응답이 생성된 시간을 표시함  
예시) *Date: Tue, 17 Oct 2023 12:34:56 GMT*
- **Server:** 응답을 생성한 서버 소프트웨어에 대한 정보를 제공함  
예시) *Server: Apache/2.4.41 (Ubuntu)*

## Body

- 서버가 클라이언트에 전달하고자 하는 실제 데이터를 포함
- 본문은 요청에 따라 변경될 수 있으며,  
클라이언트의 요청과 관련된 결과 또는 데이터를 포함

# API 보안: API Key, JWT, CORS

API는 적절한 인증을 통해 권한이 부여된 사용자만 접근할 수 있도록 해야 하며, 데이터 무결성을 보장해야 함

## API Key

- 클라이언트에 할당된 간단한 텍스트 문자열로 요청 헤더 또는 URL에 포함시켜 인증에 사용됨
- 예를 들어 클라이언트는 API 키를 사용하여 요청을 보냄 `GET/api/resource?api_key=YOUR_API_KEY`.
- 구현하기 쉽지만 보안이 취약하며 키가 노출되면 권한이 없는 사용자가 API에 쉽게 접근할 수 있음

## JWT (JSON Web Token)

- 로그인 후 각 요청에 대해 신원을 확인하기 위해 사용하는 토큰
- 토큰 자체에 인증 및 사용자 정보가 포함되어 있으며 **Header**, **Payload**, and **Signature** 로 구성됨

```
{
  "alg": "HS256",
  "typ": "JWT"
}
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
)
```

## CORS (Cross-Origin Resource Sharing)

- 다른 도메인에서 API에 접근할 때 접근을 제어하는 보안 정책
- 일반적으로 웹 애플리케이션은 서로 다른 도메인의 API에 접근할 수 있음 (example.com 이 api.example.com 에 접근). 기본적으로 브라우저는 이를 차단
- 예를 들어 `example.com`이 `api.example.com`에 접근하려면 `example.com`은 CORS를 허용해야함
- For `example.com` to access resources on `api.example.com`, the latter must set CORS to allow `example.com`
- 악성 사이트가 API를 통해 데이터를 불법적으로 사용하거나 훔치는 것을 방지

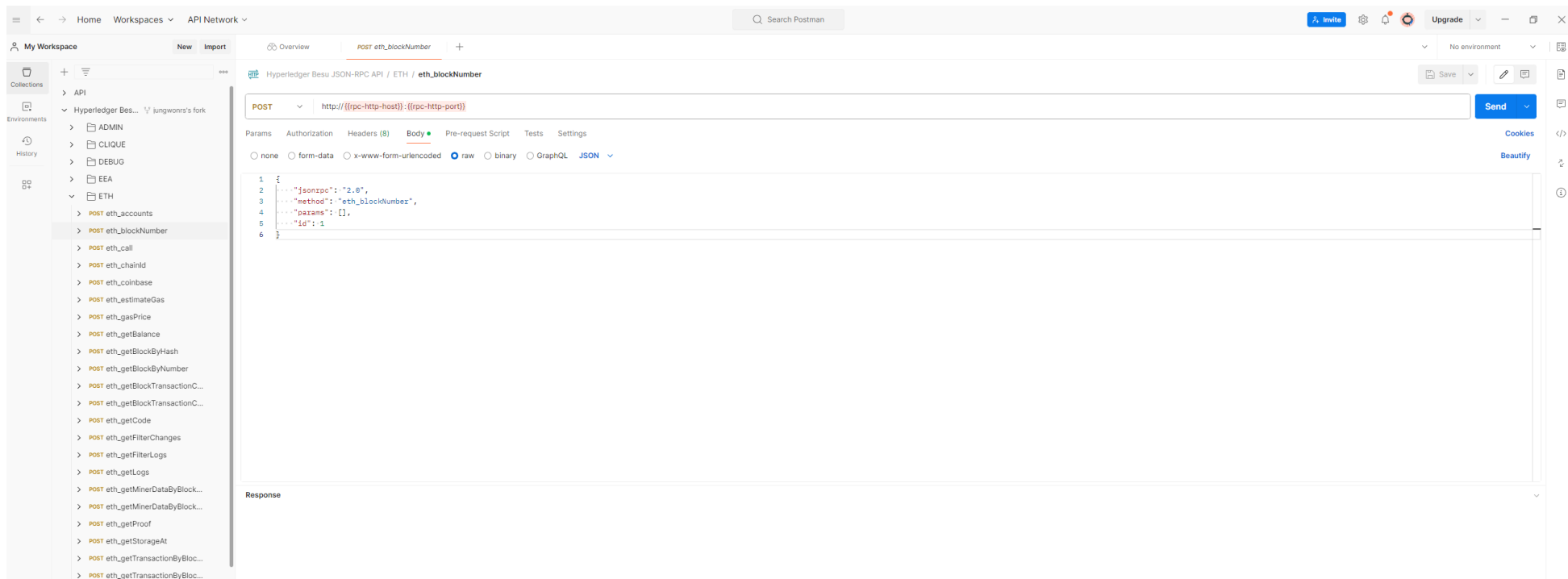


# API 테스트: Postman

- API 테스트는 클라이언트와 서버 간의 정확한 통신을 보장하기 위해 필수적임
- **기능 검증:** API가 예상대로 작동하는지 확인함
- **오류 방지:** 개발 초기에 문제를 감지하여 클라이언트와 서버 간의 통신 오류를 줄이고 빠르게 수정할 수 있음
- **안정성 보장:** 여러 사용자 요청에 대해 API가 안정적으로 응답하는지 확인하고 응답 시간이 적절한지 확인함
- API 테스트 도구로 기능과 성능을 검증하여 서비스 품질을 향상하고 안정성을 보장함

## Postman

- 사용자가 직관적인 사용자 인터페이스를 통해 Endpoint에 쉽게 요청을 보내고 응답을 확인할 수 있는 RESTful API 테스트에 널리 사용되는 도구

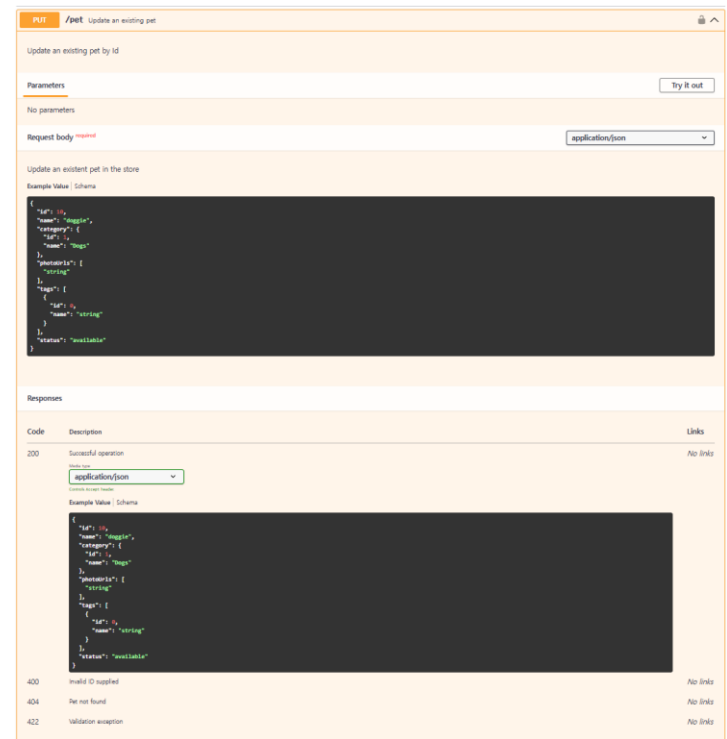
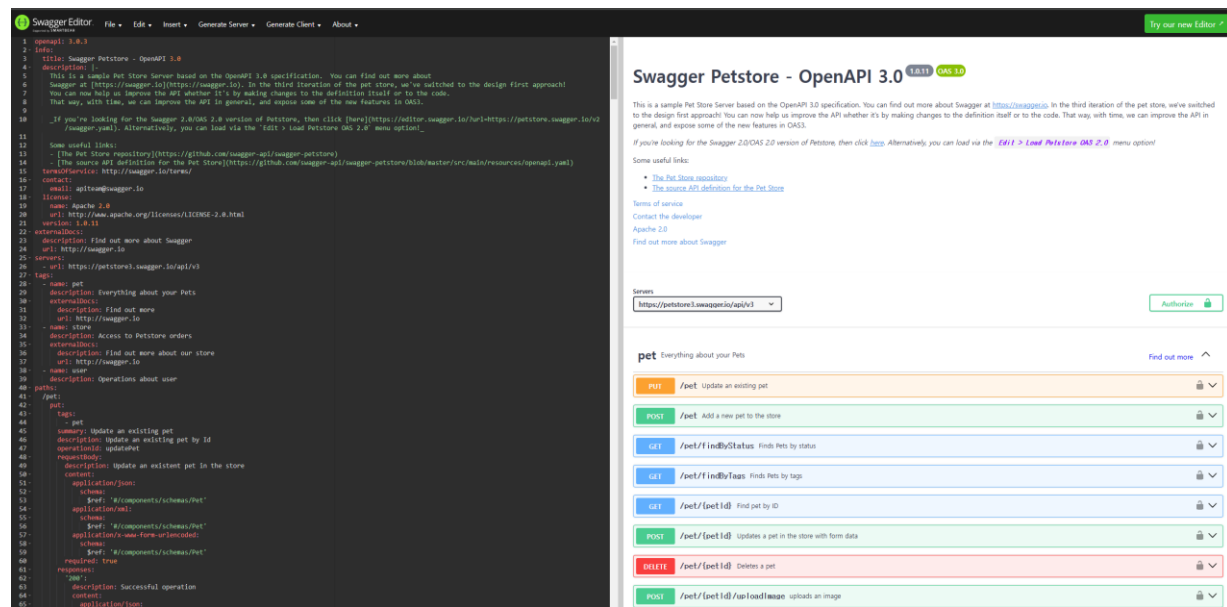


# API 문서: Swagger


- API 문서는 API 이해를 돕기 위해 클라이언트와 서버 간의 통신 표준을 정의하는 API 설계 및 개발의 중요한 단계임
- **사용성 향상:** 잘 문서화된 API는 각 Endpoint의 기능을 쉽게 이해할 수 있게 도와줌. API 사용자는 클라이언트 개발자가 요청 형식, 매개변수, 응답 형식을 확인 할 수 있어 오류를 줄일 수 있음
- **유지 관리의 용이성:** 문서화된 API는 시간이 지나도 일관된 정보를 제공하므로 API를 체계적으로 업데이트 하거나 확장하기가 더 쉬워 짐
- **협업 강화:** 팀 협업에서 API 문서는 다른 팀 (예: 프론트엔드, 모바일)이 API 사용 방법을 빠르게 이해할 수 있도록 도와주므로 효율성이 향상됨

## Swagger

- Swagger는 *API 문서 자동화를 위한 오픈 소스 도구임*
- 코드에서 API 정의를 추출하여 웹 페이지 형식의 문서를 만들어 줌




# REST API 실습: Postman

 Postman  
<https://www.postman.com>

Postman: The World's Leading API Platform | Sign Up for Free

Postman is the platform where teams build those APIs together. With built-in support for the Model Context Protocol (MCP), Postman helps you design, test, and ...

[Download Postman](#) [Plans & Pricing](#) [The Postman API Network](#) [API Client](#)

 Postman  
<https://www.postman.com> > downloads

Download Postman | Get Started for Free

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

[Postman Agent](#) [Contact Sales](#) [Install the Postman CLI](#) [Postman](#)

## Download Postman

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

### The Postman app

Download the app to get started with the Postman API Platform.

[Windows 64-bit](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes](#)

Not your OS? Download for Mac ([Intel Chip](#), [Apple Chip](#)) or Linux ([x64](#), [arm64](#))

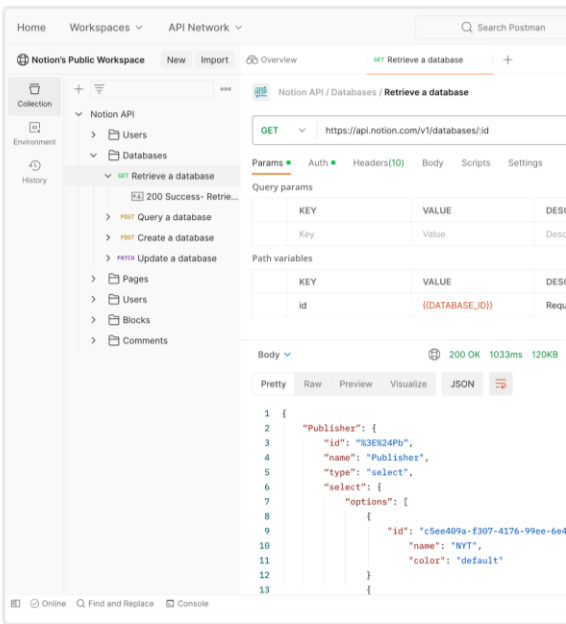
### Postman on the web

Access the Postman API Platform through your web browser. Create a free account, and you're in.

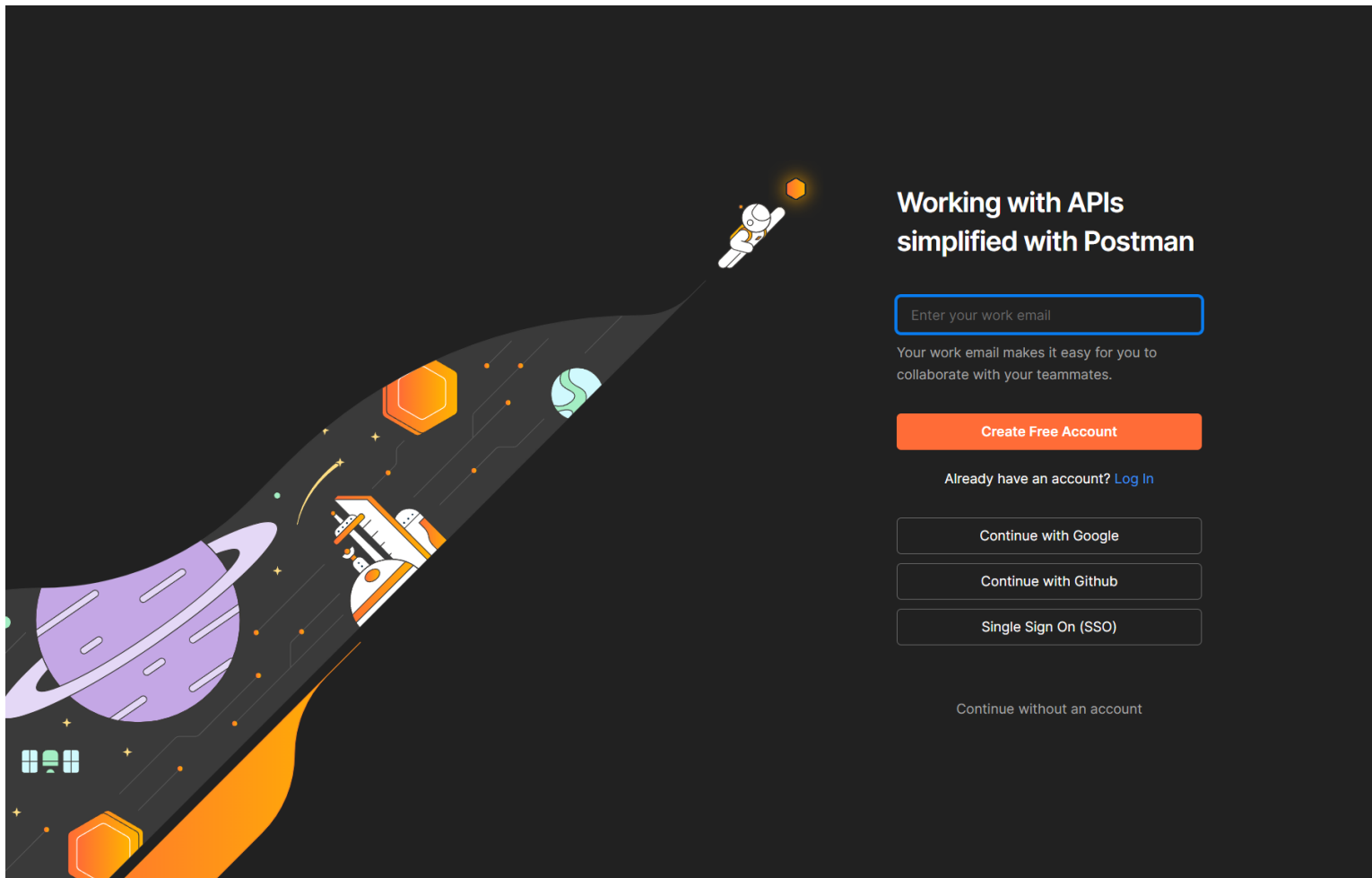
[Try the Web Version](#)

### Postman Enterprise

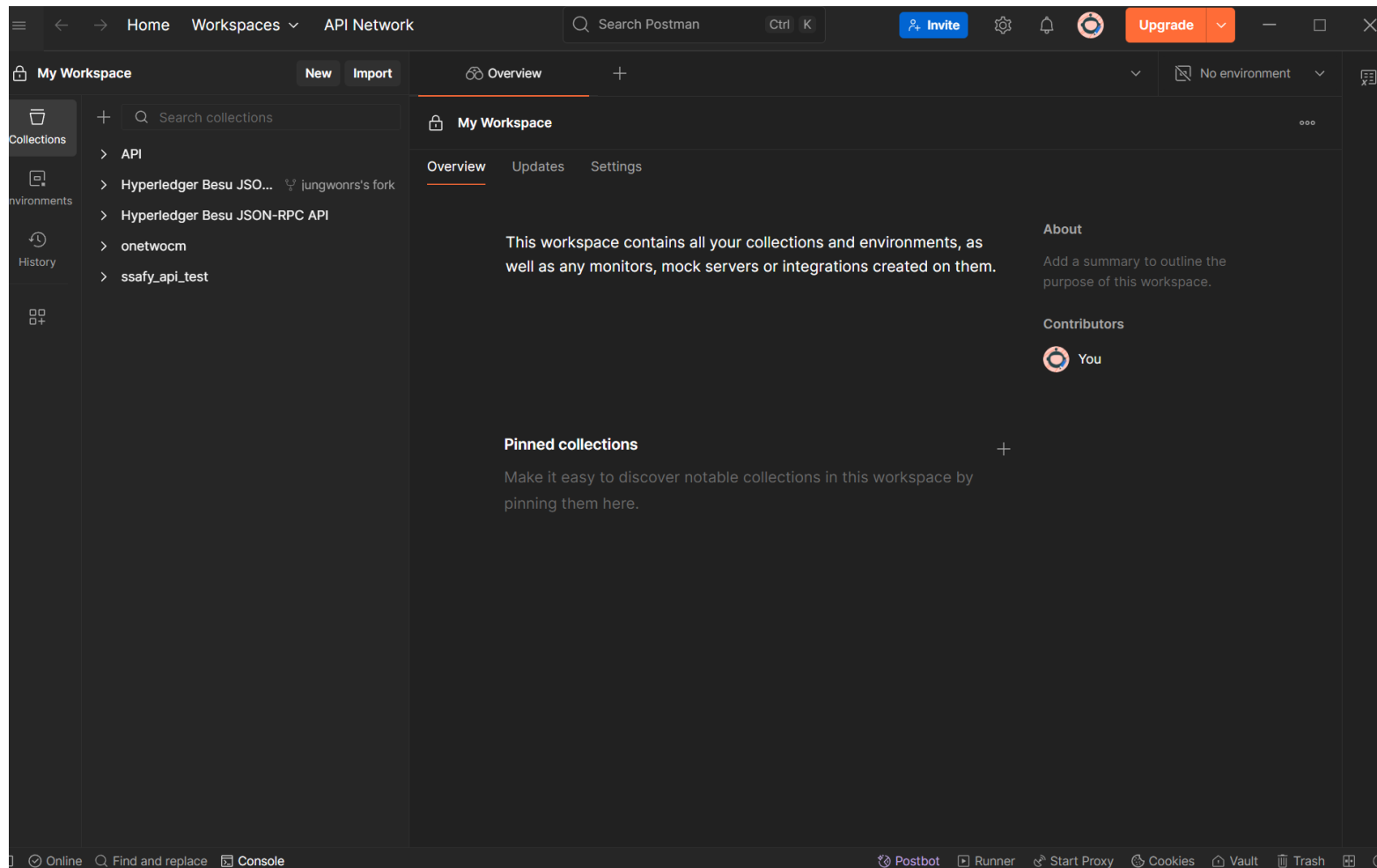
Postman Enterprise is designed for organizations who need to deploy Postman at scale.

[Learn more](#)

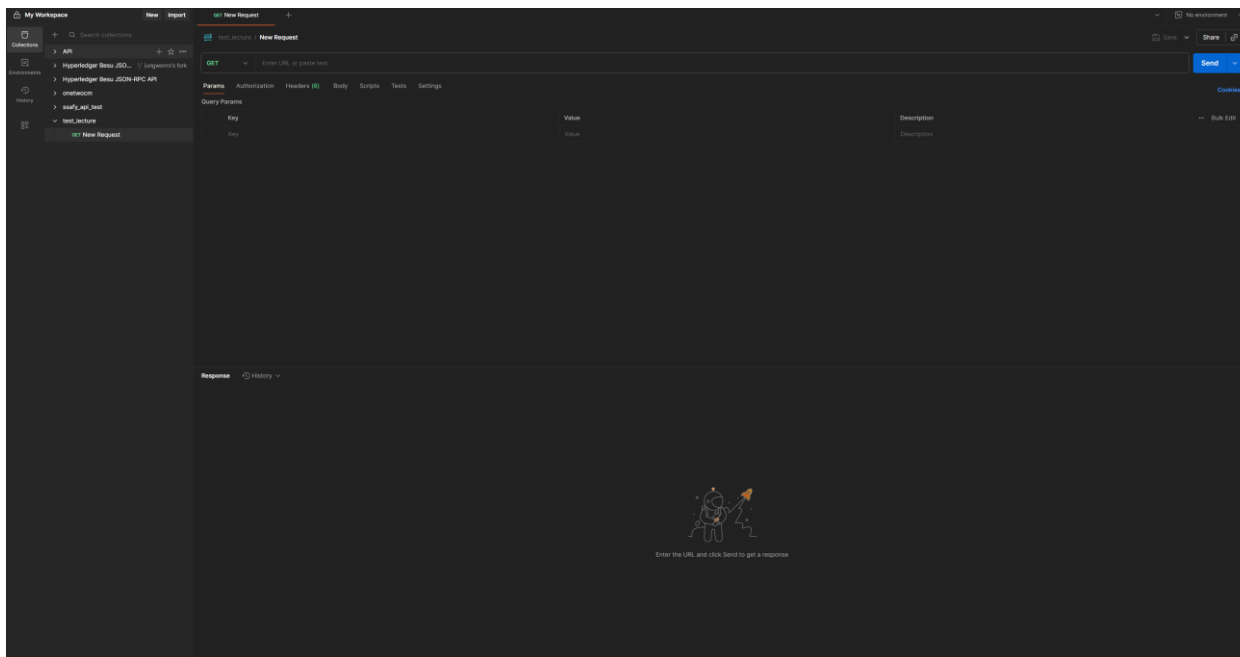
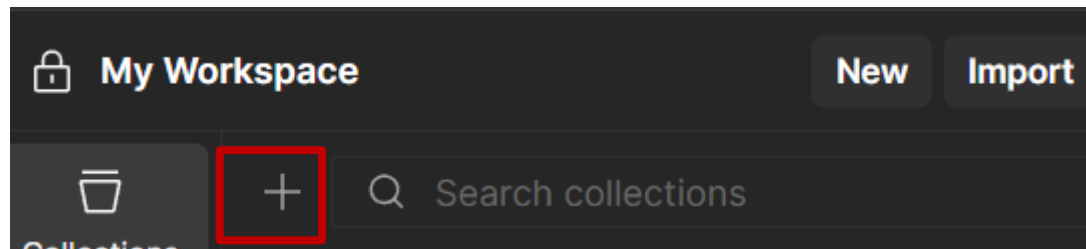
# REST API 실습: Postman



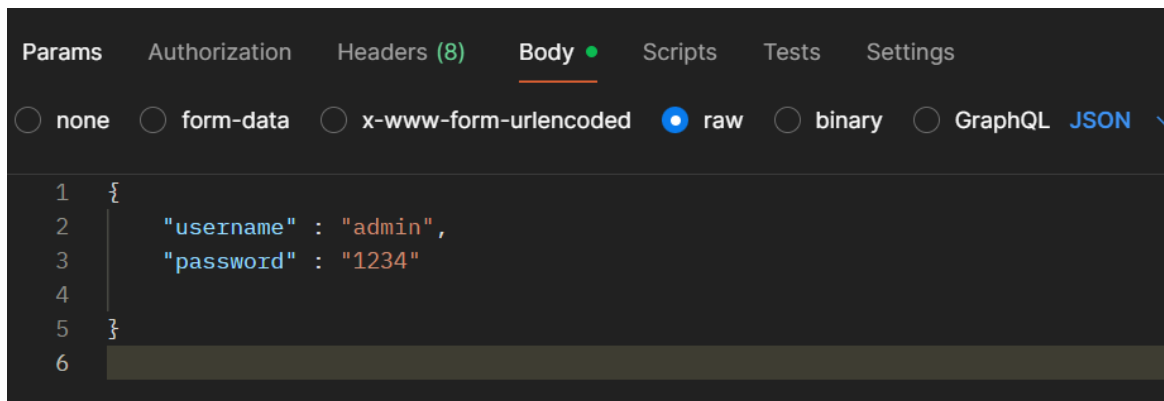
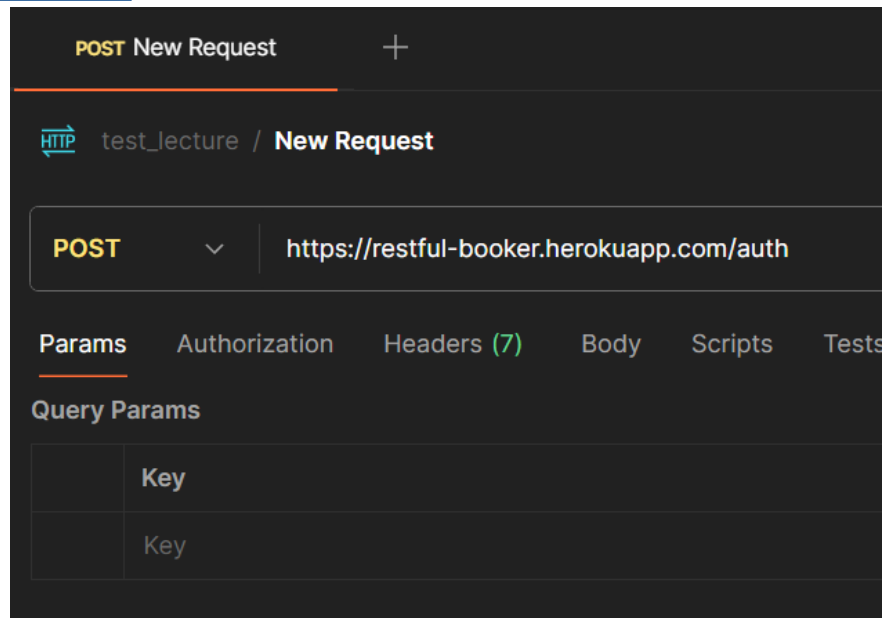
# REST API 실습: Postman



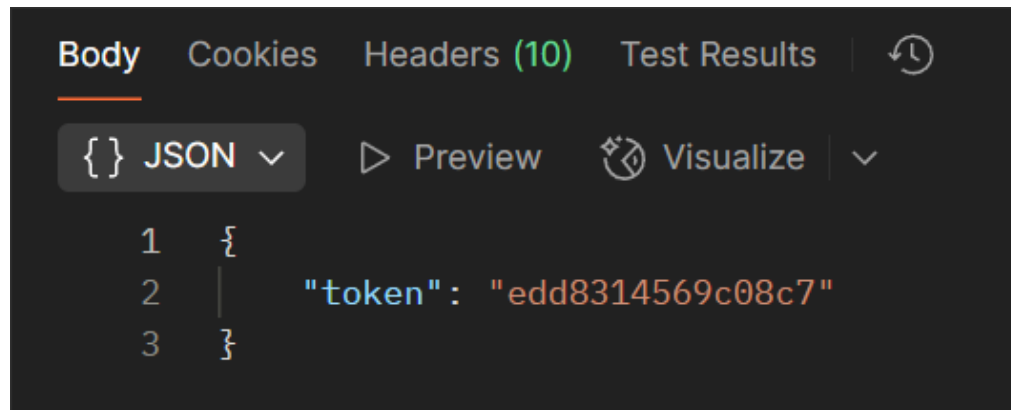
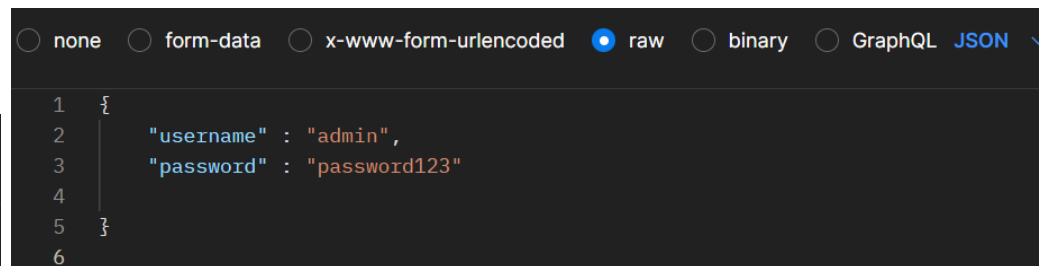
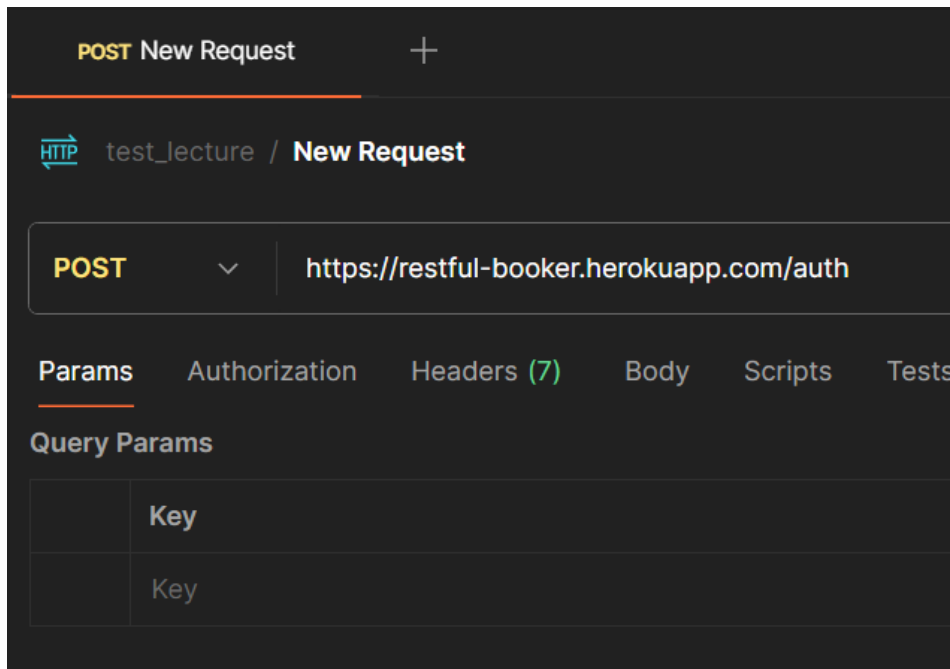
# REST API 실습: Postman



# REST API 실습: Postman

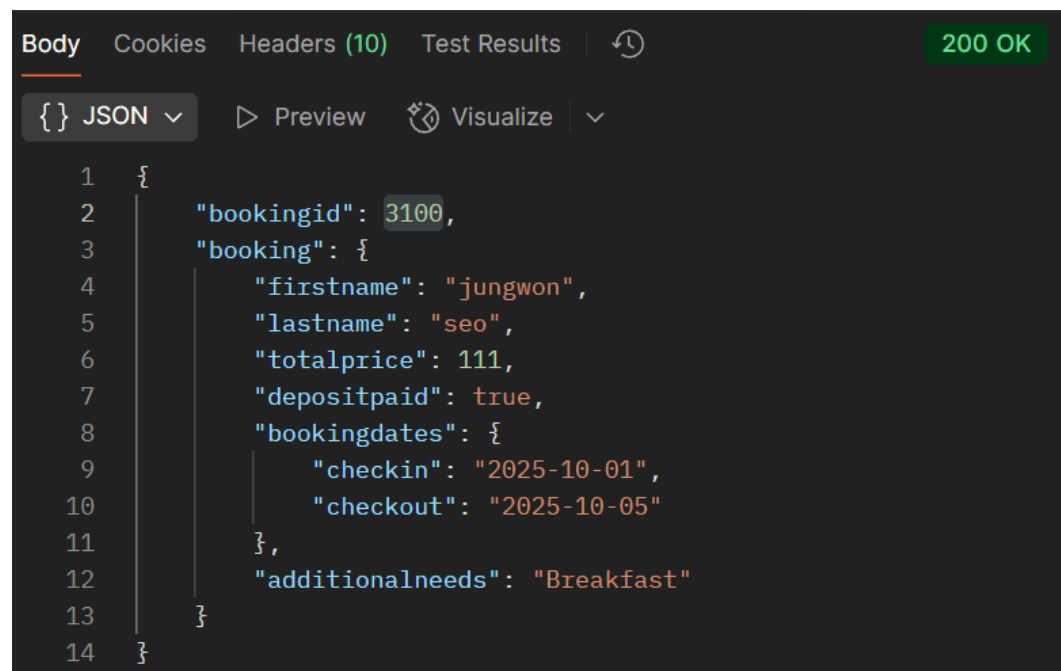
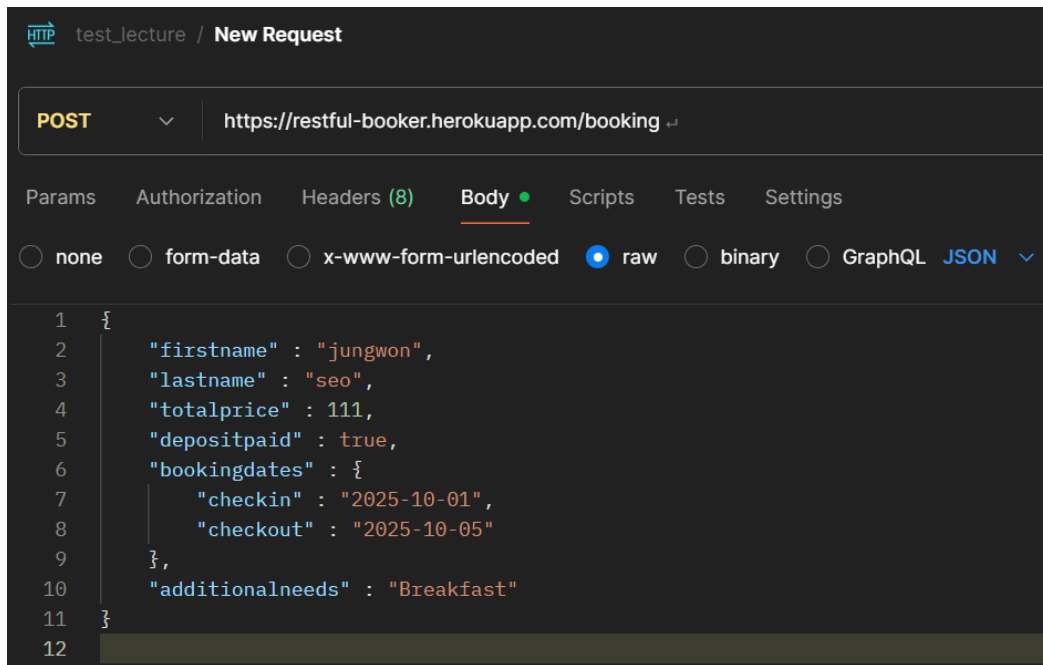


# REST API 실습: Postman

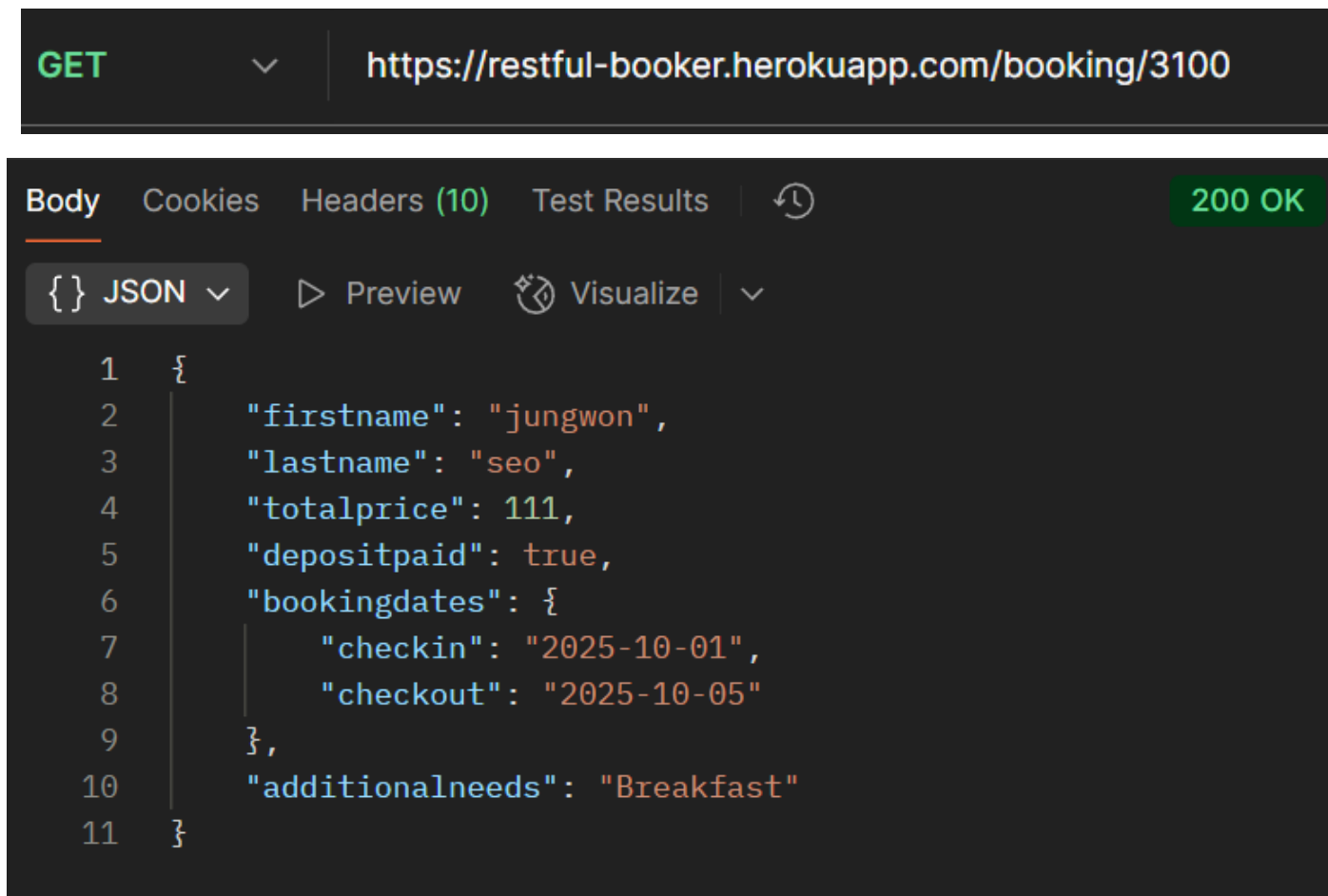




# REST API 실습: Postman



# REST API 실습: Postman



# REST API 실습: Postman

PUT

https://restful-booker.herokuapp.com/booking/3100

Params Authorization **Headers (9)** Body Scripts Tests Settings

Headers 7 hidden

	Key	Value
<input checked="" type="checkbox"/>	Content-Type	application/json
<input checked="" type="checkbox"/>	Cookie	token=edd8314569c08c7
	Key	Value

Params Authorization Headers (10) **Body** Scripts Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

```
1 {
2   "firstname" : "jw",
3   "lastname" : "s",
4   "totalprice" : 111,
5   "depositpaid" : true,
6   "bookingdates" : {
7     "checkin" : "2025-10-01",
8     "checkout" : "2025-10-05"
9   },
10  "additionalneeds" : "Ocean view room please"
11 }
```

**Body** Cookies Headers (10) Test Results **200 OK**

**JSON** Preview Visualize

```
1 {
2   "firstname": "jw",
3   "lastname": "s",
4   "totalprice": 111,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin": "2025-10-01",
8     "checkout": "2025-10-05"
9   },
10  "additionalneeds": "Ocean view room please"
11 }
```

# REST API 실습: Postman

DELETE



https://restful-booker.herokuapp.com/booking/

Params	Authorization	Headers (8)	Body	Scripts	Tests	Settings
Headers  6 hidden						
Key		Value				
<input checked="" type="checkbox"/>	Content-Type	application/json				
<input checked="" type="checkbox"/>	Cookie	token=4940f938ea93785				

Body Cookies Headers (10) Test Results 201 Created

Raw

Preview

Visualize

1 Created

# REST API 실습: Swagger



Swagger

<https://swagger.io>

## Swagger: API Documentation & Design Tools for Teams

Simplify API development for users, teams, and enterprises with our open source and professional toolset. Find out how **Swagger** can help you and get started ...

### Editor

This is a sample Pet Store Server based on the OpenAPI 3.0 ...

### Swagger UI

Swagger UI allows anyone — be it your development team or your ...

### Swagger Editor

Design, describe, & document your API on the first open source ...

### Swagger Documentation

A user-friendly tool for creating, editing, and visualizing API ...

### Find your tool

Explore the key features, differences, and advantages of ...

[swagger.io 검색결과 더보기 »](#)



[Tools](#) [Learn](#) [Resources](#)



[Sign In](#)

[Get started](#)

## API Development for Everyone

Simplify your API development with our open-source and professional tools, built to help you and your team efficiently design and document APIs at scale.

[Find your tool](#)

[Read the docs →](#)

TRUSTED BY



**pet** Everything about your Pets

**POST** `/pet/{petId}/uploadImage` uploads an image

**POST** `/pet` Add a new pet to the store

**PUT** `/pet` Update an existing pet

**GET** `/pet/findByStatus` Finds Pets by status

**GET** `/pet/{petId}` Find pet by ID

**POST** `/pet/{petId}` Updates a pet in the store with form data

**DELETE** `/pet/{petId}` Deletes a pet

# REST API 실습: Swagger

Tools ▾ Learn ▾ Resources ▾



PRO

## API Hub

Accelerate API development with quality and consistency across OpenAPI and AsyncAPI.

### Design

Collaborate on API Design

### Portal

Deliver Up-to-date API Documentation

### Explore

Quickly Test and Explore APIs

### Testing

Automated API Testing

### Contract Testing

Block API Breaking Changes

OPEN SOURCE

## Swagger Open Source

Ideal for individuals and small teams to design, build, and document APIs.

### Swagger Codegen

Generate server stubs and client SDKs from OpenAPI Specification definitions.

### Swagger Editor

API editor for designing APIs with the OpenAPI and AsyncAPI specifications.

### Swagger UI

Visualize OpenAPI Specification definitions in an interactive UI.

## API Hub Enterprise

Standardize your APIs with projects, style checks, and reusable domains.

[Explore all tools >](#)



Tools ▾ Learn ▾ Resources ▾



[Sign In](#)

## Swagger Open Source

[Editor](#) [Codegen](#) [UI](#)

[Try API Hub](#)

## Swagger Editor

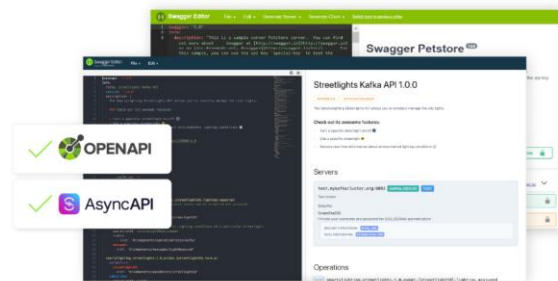
Design, describe, and document your API on the first open source editor supporting multiple API specifications and serialization formats. The Swagger Editor offers an easy way to get started with the OpenAPI Specification (formerly known as Swagger) as well as the AsyncAPI specification, with support for Swagger 2.0, OpenAPI 3.\*, and AsyncAPI 2.\* versions.

[Try Swagger Editor ↗](#)

[Try Swagger Editor Next \(beta\) ↗](#)

[Download Swagger Editor](#)

[Compare Editor Versions >](#)



# REST API 실습: Swagger

Swagger Editor

File Edit Insert Generate Server Generate Client About

Sign in to Swagger Pro Try Swagger Pro

```
1 openapi: 3.0.4
2 info:
3   title: Swagger Petstore - OpenAPI 3.0
4   description: |-
5     This is a sample Pet Store Server based on the OpenAPI 3.0 specification.
6     You can find out more about
7     Swagger at [https://swagger.io](https://swagger.io). In the third iteration
8     of the pet store, we've switched to the design first approach!
9     You can now help us improve the API whether it's by making changes to the
10    definition itself or to the code.
11    That way, with time, we can improve the API in general, and expose some of
12    the new features in OAS3.
13
14    Some useful links:
15    - The Pet Store repository [https://github.com/swagger-api/swagger-petstore]
16    - The source API definition for the Pet Store [https://github.com/swagger
17      -api/swagger-petstore/blob/master/src/main/resources/openapi.yaml]
18
19    termsOfService: https://swagger.io/terms/
20    contact:
21      email: apiteam@swagger.io
22    license:
23      name: Apache 2.0
24      url: https://www.apache.org/licenses/LICENSE-2.0.html
25    version: 1.0.12
26
27 externalDocs:
28   description: Find out more about Swagger
29   url: https://swagger.io
30
31 servers:
32   - url: https://petstore3.swagger.io/api/v3
33
34 tags:
35   - name: pet
36     description: Everything about your Pets
37   externalDocs:
38     description: Find out more
39     url: https://swagger.io
40   - name: store
41     description: Access to Petstore orders
42   externalDocs:
43     description: Find out more about our store
44     url: https://swagger.io
45   - name: user
46     description: Operations about user
47
48 paths:
49   /pet:
50     put:
51       tags:
52         - pet
53       summary: Update an existing pet.
54       description: Update an existing pet by Id.
55       operationId: updatePet
56       requestBody:
57         description: Update an existent pet in the store
58         content:
59           application/json:
60             schema:
61               $ref: '#/components/schemas/Pet'
62           application/xml:
63             schema:
64               $ref: '#/components/schemas/Pet'
65           application/x-www-form-urlencoded:
66             schema:
67               $ref: '#/components/schemas/Pet'
68         required: true
69       responses:
70         '200':
71           description: Successful operation
72           content:
73             application/json:
74               schema:
75                 $ref: '#/components/schemas/Pet'
76             application/xml:
77               schema:
78                 $ref: '#/components/schemas/Pet'
79         '400':
80           description: Invalid ID supplied
81         '404':
82           description: Pet not found
83         '422':
84           description: Validation exception
85       default:
86         description: Unexpected error
87       content:
88         application/json:
89           schema:
90             $ref: '#/components/schemas/Error'
91       security:
92         - petstore_auth:
93           - write:pets
94           - read:pets
95     post:
96       tags:
97         - pet
98       summary: Add a new pet to the store.
99       description: Add a new pet to the store.
100      operationId: addPet
101      requestBody:
102        description: Create a new pet to the store.
```

Swagger Petstore - OpenAPI 3.0 1.0.12

OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at <https://swagger.io>. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

Some useful links:

- [The Pet Store repository](#)
- [The source API definition for the Pet Store](#)

Terms of service

Contact the developer

Apache 2.0

[Find out more about Swagger](#)

Servers

https://petstore3.swagger.io/api/v3

Authorize

pet

Everything about your Pets

Find out more

PUT /pet Update an existing pet.

POST /pet Add a new pet to the store.

GET /pet/findByStatus Finds Pets by status.

GET /pet/findByTags Finds Pets by tags.

GET /pet/{petId} Find pet by ID.

POST /pet/{petId} Updates a pet in the store with form data.

DELETE /pet/{petId} Deletes a pet.

POST /pet/{petId}/uploadImage Uploads an image.

store

Access to Petstore orders

Find out more about our store

GET /store/inventory Returns pet inventories by status.

POST /store/order Place an order for a pet.

GET /store/order/{orderId} Find purchase order by ID.

DELETE /store/order/{orderId} Delete purchase order by identifier.

user

Operations about user

# REST API 실습: Swagger

```
1  openapi: 3.0.3
2  info:
3    title: User API Example
4    version: 1.0.0
5
6  paths:
7    /users:
8      get:
9        summary: 모든 사용자 조회
10       responses:
11         '200':
12           description: 사용자 목록
13       post:
14         summary: 새로운 사용자 생성
15         requestBody:
16           required: true
17           content:
18             application/json:
19               schema:
20                 type: object
21                 properties:
22                   name: { type: string }
23                   email: { type: string }
24         responses:
25           '201':
26             description: 생성 완료
27
```

```
28  /users/{id}:
29    parameters: # ✅ 여기서 id를 한번만 정의
30      - name: id
31        in: path
32        required: true
33        schema:
34          type: integer
35
36    get:
37      summary: 특정 사용자 조회
38      responses:
39        '200':
40          description: 사용자 정보
41    put:
42      summary: 사용자 전체 수정
43      requestBody:
44        required: true
45        content:
46          application/json:
47            schema:
48              type: object
49              properties:
50                name: { type: string }
51                email: { type: string }
52      responses:
53        '200':
54          description: 수정 완료
55    patch:
56      summary: 사용자 일부 수정
57      requestBody:
58        required: true
59        content:
60          application/json:
61            schema:
62              type: object
63              properties:
64                email: { type: string }
65      responses:
66        '200':
67          description: 일부 수정 완료
68    delete:
69      summary: 사용자 삭제
70      responses:
71        '204':
72          description: 삭제 완료
73
```



---

Q & A

---