RESTful API

Resstful API

Spring Boot

RESTful API

- **RESTful API**
  - RESTful API는 웹에서 서버와 클라이언트가 데이터를 주고받는 방식을 설계하는 아키텍처 프레임이다.
  - REST (Representational State Transfer) 는 2000년 로이 필딩(Roy Fielding) 이 논문에서 제안한 웹 기반의 설계 원칙으로
  - "**리소스(Resource)를 URL로 표현하고, 그 리소스에 대한 행위는 HTTP 메서드로 구분한다**" 개념이다.
  - 행위는 **HTTP 메서드(GET, POST, PUT, DELETE 등)** 로 구분한다.

RESTful API

- **주요 HTTP 메서드와 의미**

메서드	의미	URL
GET	데이터 조회(Read)	/users – 모든 정보 조회 /users/{id} – 특정 id 조회
POST	데이터 생성(Create)	/users
PUT	데이터 수정(Update)	/users/{id}
DELETE	데이터 삭제(Delete)	/users/{id}

RESTful API

- **RequestBody & ResponseBody**
  - 웹에서 화면전환(새로고침) 없이 이루어지는 동작들은 대부분 ***비동기 통신***으로 이루어진다.
  - 비동기통신을 하기 위해서는 클라이언트에서 서버로 요청 메세지를 보낼 때, 본문에 데이터를 담아서 보내야 하고, 서버에서 클라이언트로 응답을 보낼 때에도 본문에 데이터를 담아서 보내야 한다. 이 본문이 바로 body 이다.

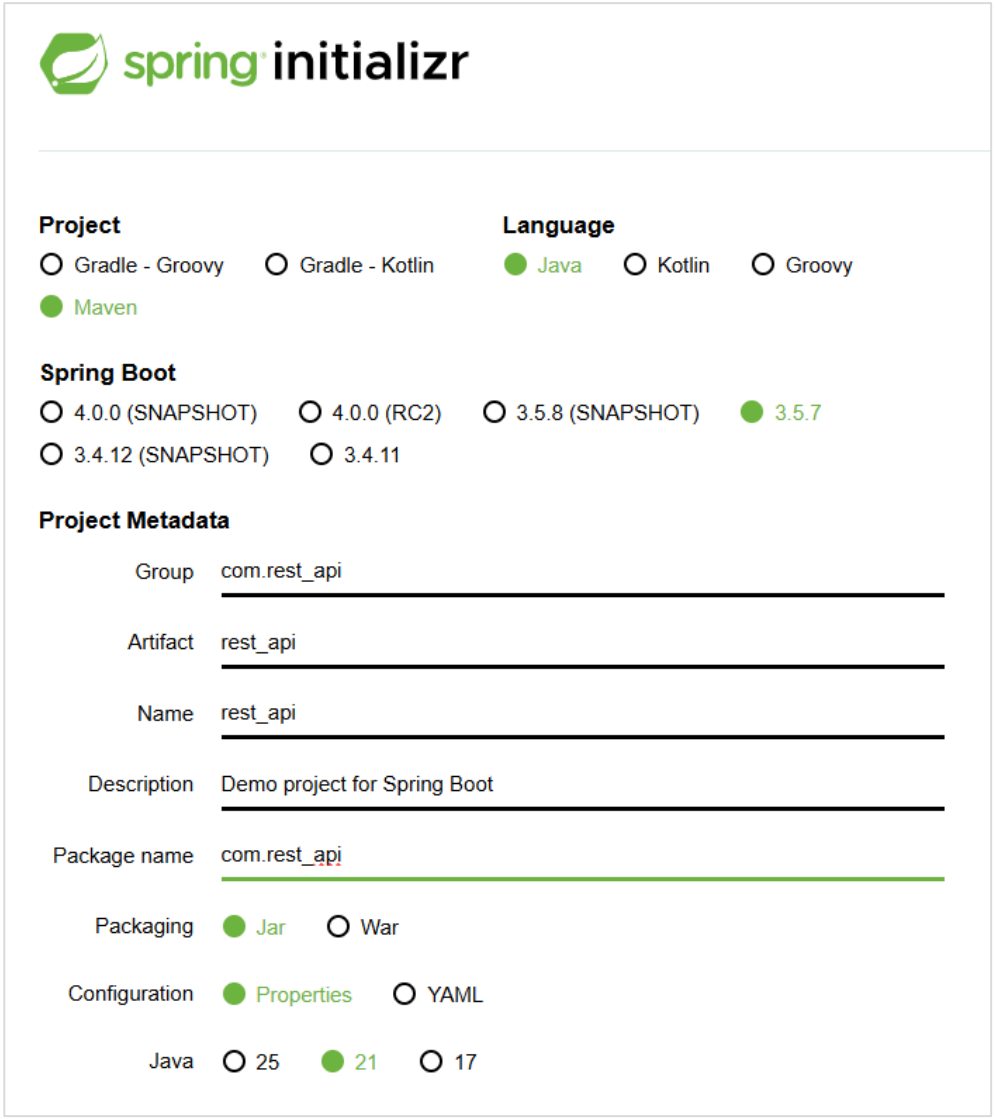
    즉, 요청 본문 **requestBody**, 응답 본문 **responseBody** 을 담아서 보내야 한다.

@RequestBody

이 어노테이션이 붙은 파라미터에는 http요청의 본문(body)이 그대로 전달된다.

json 기반의 메시지를 사용하는 요청의 경우에 이 방법이 매우 유용하다.

회원 관리 API


spring initializr
Project
Gradle - Groovy
Gradle - Kotlin
Maven
Language
Java
Kotlin
Groovy
Spring Boot
4.0.0 (SNAPSHOT)
4.0.0 (RC2)
3.5.8 (SNAPSHOT)
3.5.7
3.4.12 (SNAPSHOT)
3.4.11
Project Metadata
Group
com.rest_api
Artifact
rest_api
Name
rest_api
Description
Demo project for Spring Boot
Package name
com.rest_api
Packaging
Jar
War
Configuration
Properties
YAML
Java
25
21
17

회원 관리 API

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

회원 관리 API

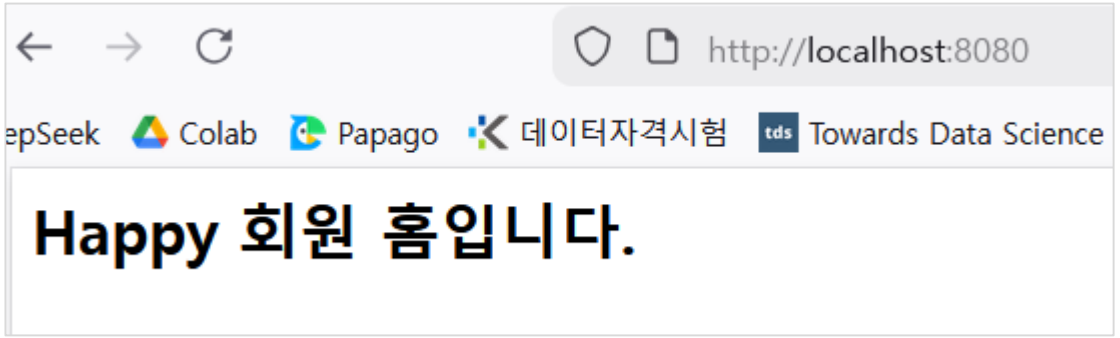
- **application.property**

```
server.port=8080

# DataSource 설정 - mysql
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/bootdb?serverTime=Asia/Seoul
spring.datasource.username=bootuser
spring.datasource.hikari.password=pwboot

# JPA 설정
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

회원 관리 API


http://localhost:8080
epSeek
Colab
Papago
데이터자격시험
tds
Towards Data Science
Happy 회원 홈입니다.

회원 관리 API

- **HomeController**

```java
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController  no usages
public class HomeController {

    @GetMapping("/")  no usages
    public String home(){
        return "<h2>Happy 회원 홈입니다.</h2>";
    }
}
```

회원 관리 API

- User 엔티티

```java
@Data
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;  //번호

    @Column(unique=true)
    private String username; //아이디

    @Column(nullable=false)
    private String password; //비밀번호

    @Column(nullable=false)
    private String email;  //이메일

    @CreationTimestamp
    @Column(updatable=false)
    private Timestamp regDate; //가입일

    @UpdateTimestamp
    @Column(insertable=false)
    private Timestamp updateDate; //수정일
}
```

회원 관리 API

- User 저장소

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Integer>{

}
```

회원 관리 API

- User 컨트롤러

```java
@RequestMapping("/users")
@RequiredArgsConstructor
@RestController
public class UserController {

    private final UserService service;

    //회원 가입
    //@RequestBody - json 데이터 요청
    @PostMapping
    public String saveUser(@RequestBody User user) {
        service.save(user);
        return "회원 가입 성공!";
    }

    //회원 목록
    @GetMapping
    public List<User> getAllUsers(){
        List<User> userList = service.findAll();
        return userList;
    }
```

회원 관리 API

- **User 컨트롤러**

```
    //회원 상세보기(정보)
    @GetMapping("/{id}")
    public User getUser(@PathVariable Integer id) {
        User user = service.findById(id);
        return user;
    }

    //회원 삭제
    @DeleteMapping("/{id}")
    public String deleteUser(@PathVariable Integer id) {
        service.delete(id);
        return "회원 삭제 완료!";
    }

    //회원 수정
    @PutMapping("/{id}")
    public String updateUser(@PathVariable Integer id,
            @RequestBody User user) {
        service.update(id, user);
        return "회원 수정 완료!";
    }
}
```

회원 관리 API

- User 서비스

```java
@RequiredArgsConstructor
@Service
public class UserService {
    //저장소 객체 생성
    private final UserRepository userRepo;

    //회원 가입
    public void save(User user) {
        userRepo.save(user);
    }

    //회원 목록 보기
    public List<User> findAll() {
        return userRepo.findAll();
    }
```

회원 관리 API

- User 서비스

```
    //회원 삭제
    public void delete(Integer id) {
        userRepo.deleteById(id);
    }

    //회원 수정
    public void update(Integer id, User updateUser) {
        User user = findById(id); //수정할 회원 가져옴
        //수정 처리
        user.setUsername(updateUser.getUsername());
        user.setPassword(updateUser.getPassword());
        user.setEmail(updateUser.getEmail());
        userRepo.save(user);
    }
}
```

회원 관리 API

- **포스트 맨(Postman) 툴 사용**
  - Postman은 API(특히 RESTful API)를 테스트하고 디버깅하기 위한 강력한 도구이다.
  - 서버에 HTTP 요청을 보내고, 응답(결과)을 바로 확인할 수 있는 API 클라이언트 프로그램이다.
  - 보통 웹 브라우저는 **GET 요청**밖에 직접 보낼 수 없지만, API 테스트에서는 **POST, PUT, DELETE** 등 다양한 HTTP 메서드가 필요하다.

회원 관리 API

- **포스트 맨(Postman) 기능**

  ✓ 모든 HTTP 메서드 테스트 가능

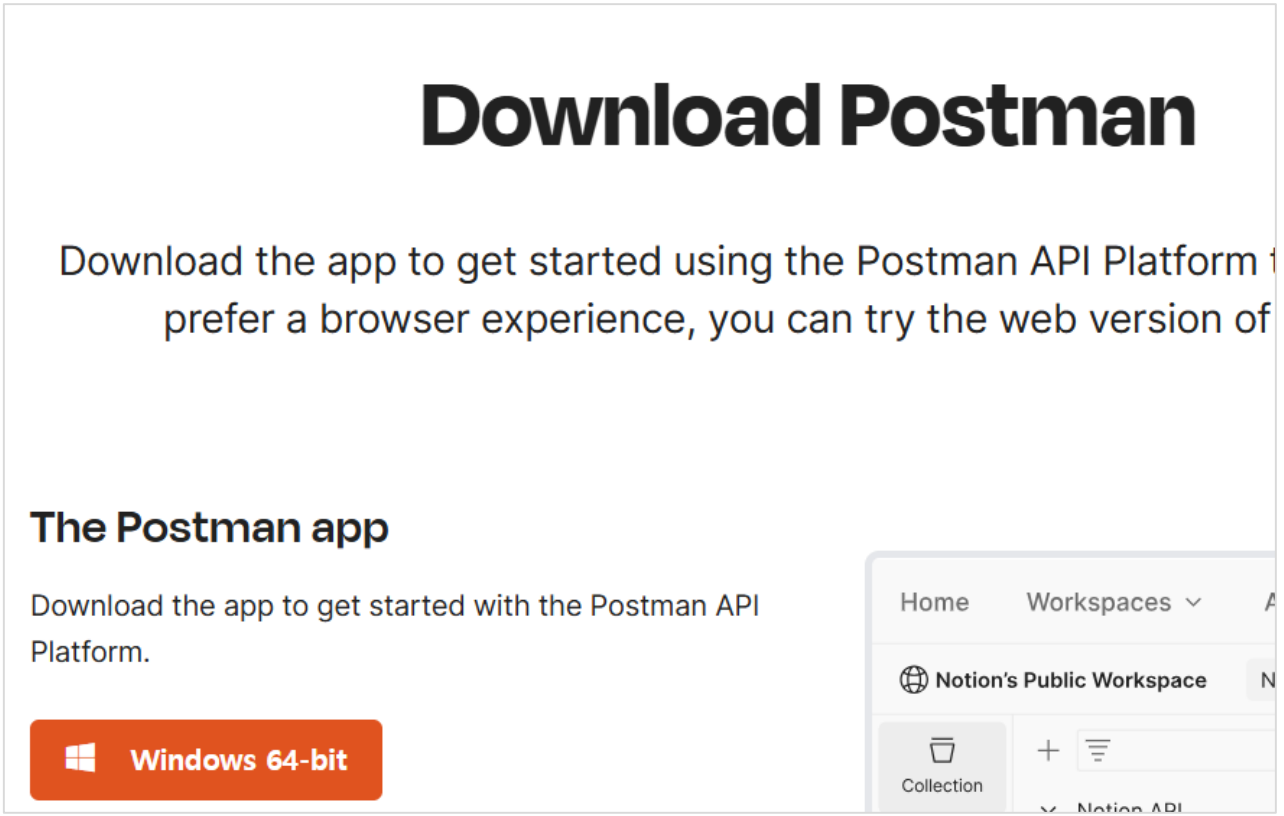
  ✓ 요청 헤더, 파라미터, 바디를 자유롭게 설정 가능

  ✓ 서버의 응답(JSON, XML 등)을 보기 쉽게 출력

  ✓ API 테스트 자동화도 가능

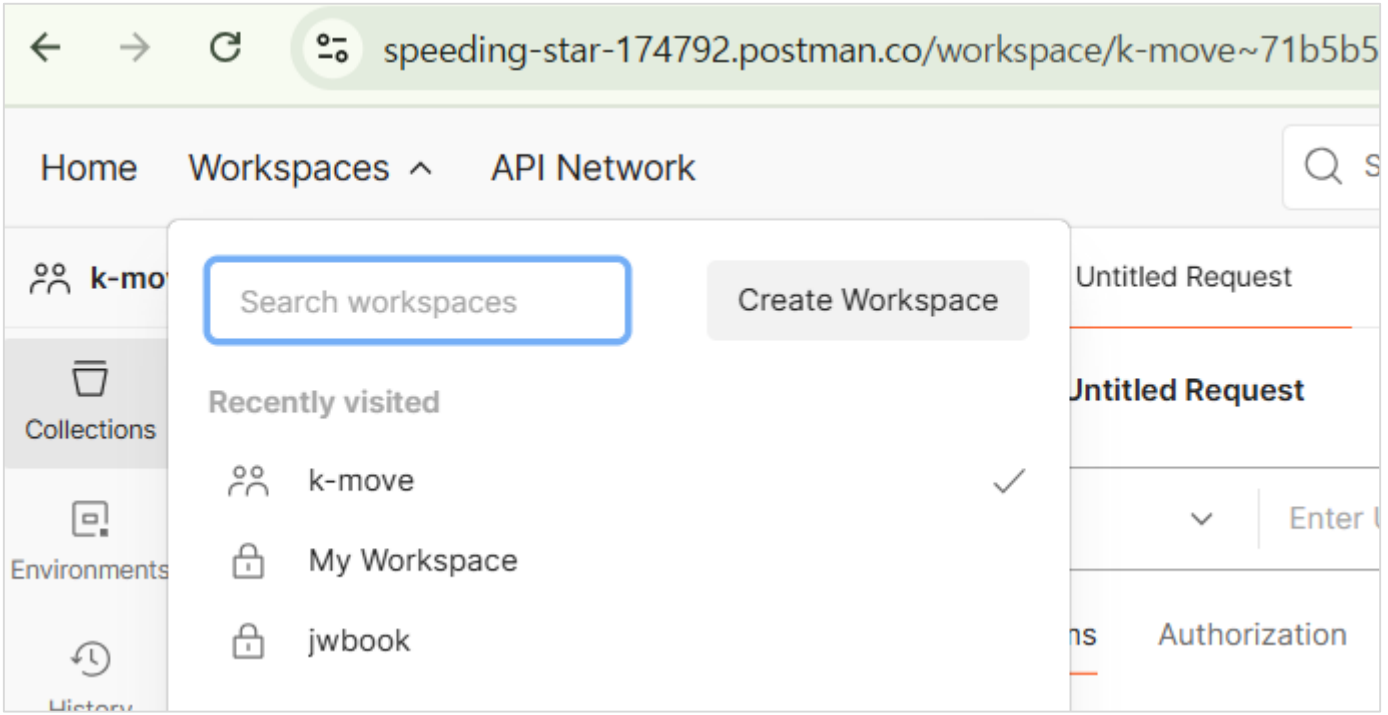
회원 관리 API

- **Postman DeskTop Tool 다운로드**



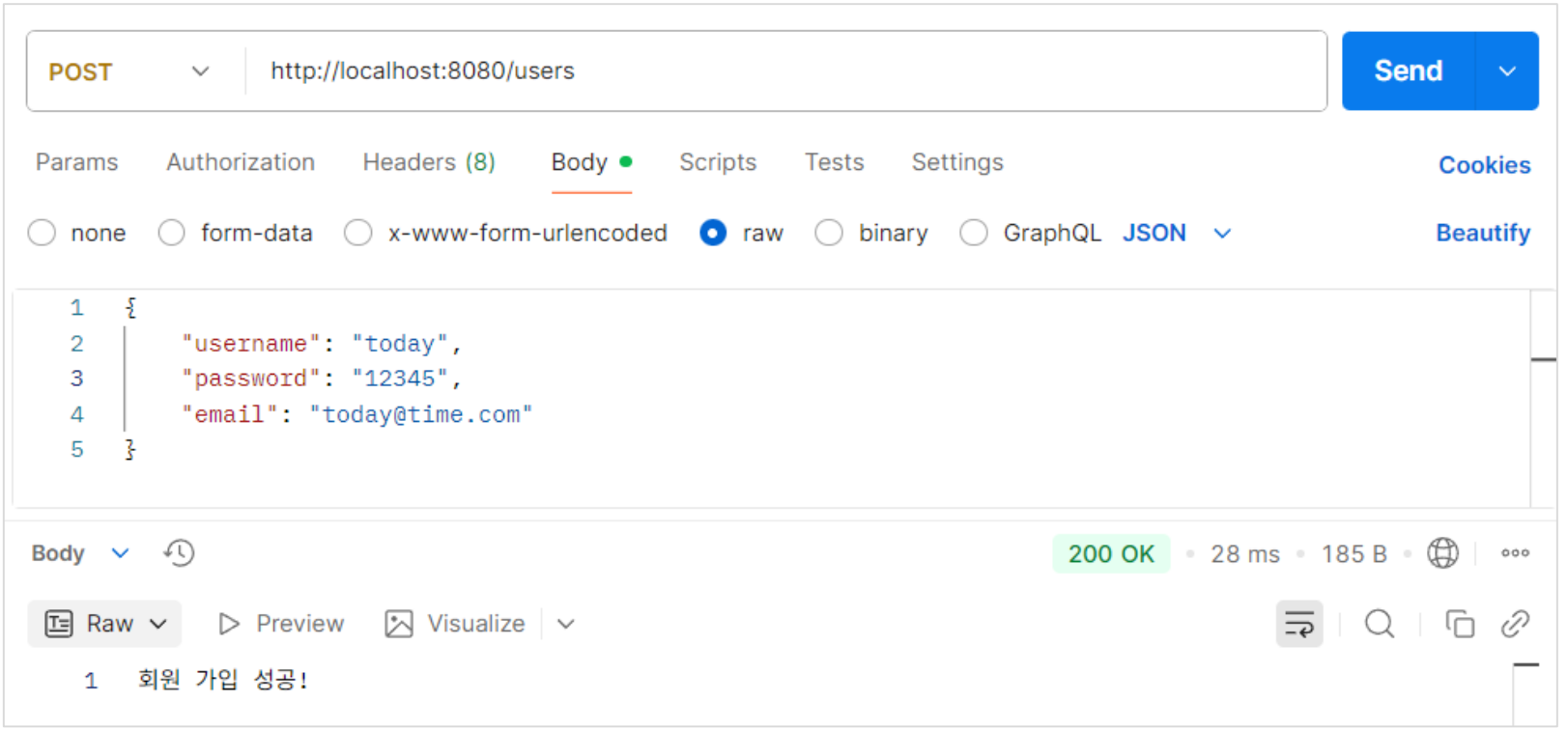
회원 관리 API

- **Workspaces 생성**



회원 관리 API

- 회원 가입 – 2명



회원 관리 API

- MySQL WorkBench

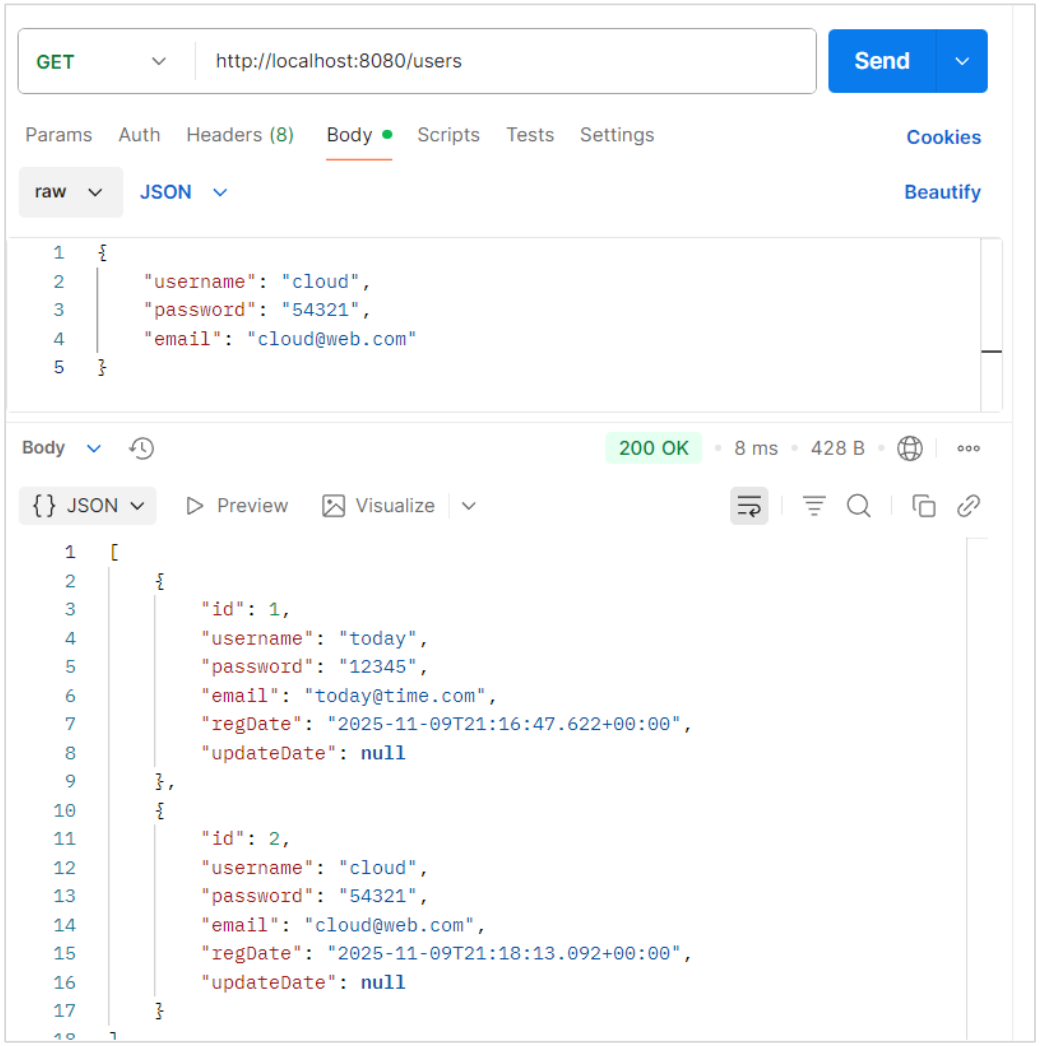
```
1 •  use bootdb;
2
3 •  select * from user;
4
```

Result Grid | Filter Rows: | Edit: | Export/Import:

	id	reg_date	update_date	email	password	username
▶	1	2025-11-10 06:16:47.622924	NULL	today@time.com	12345	today
	2	2025-11-10 06:18:13.092035	NULL	cloud@web.com	54321	cloud
*	NULL	NULL	NULL	NULL	NULL	NULL

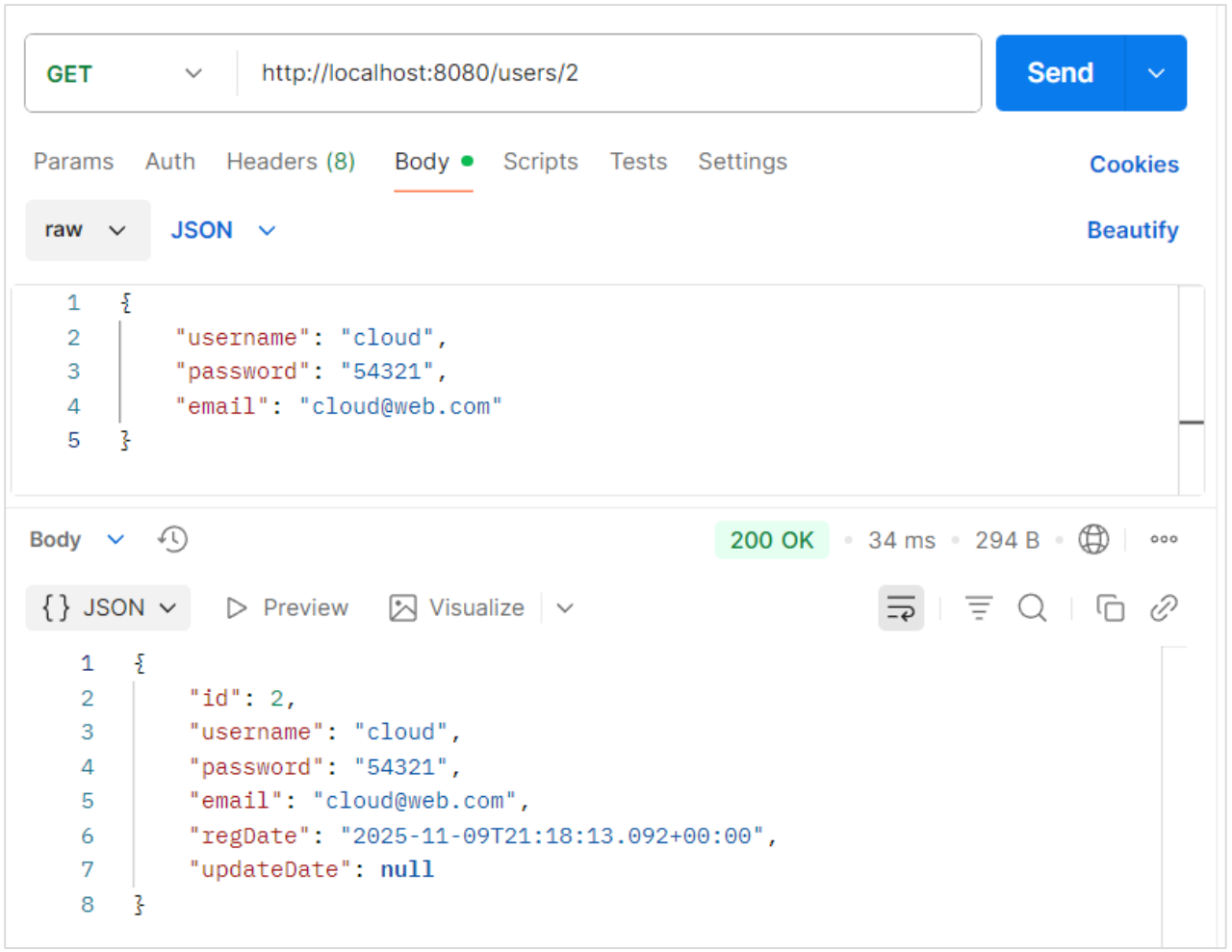
회원 관리 API

- 회원 목록



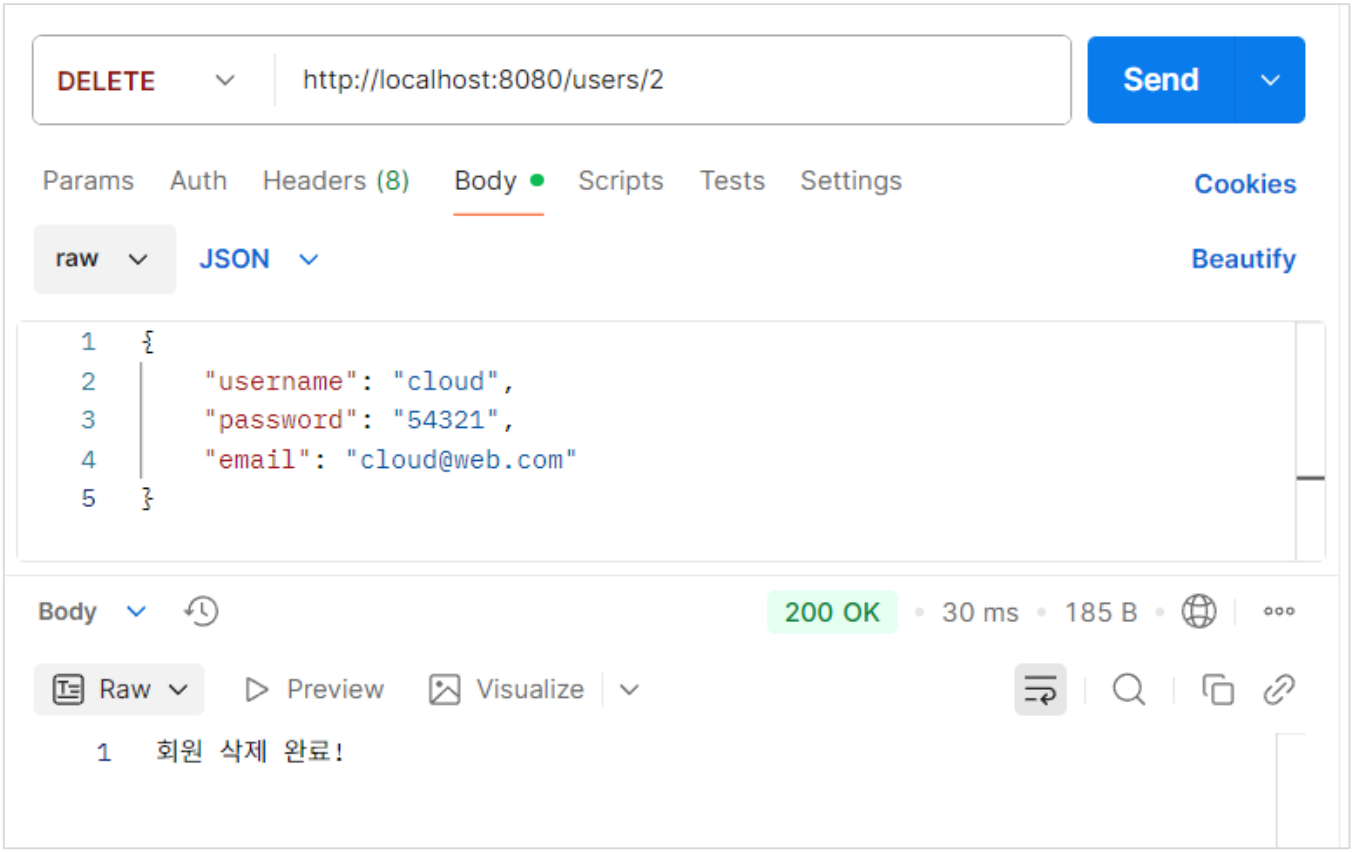
회원 관리 API

- 회원 정보(상세 보기)



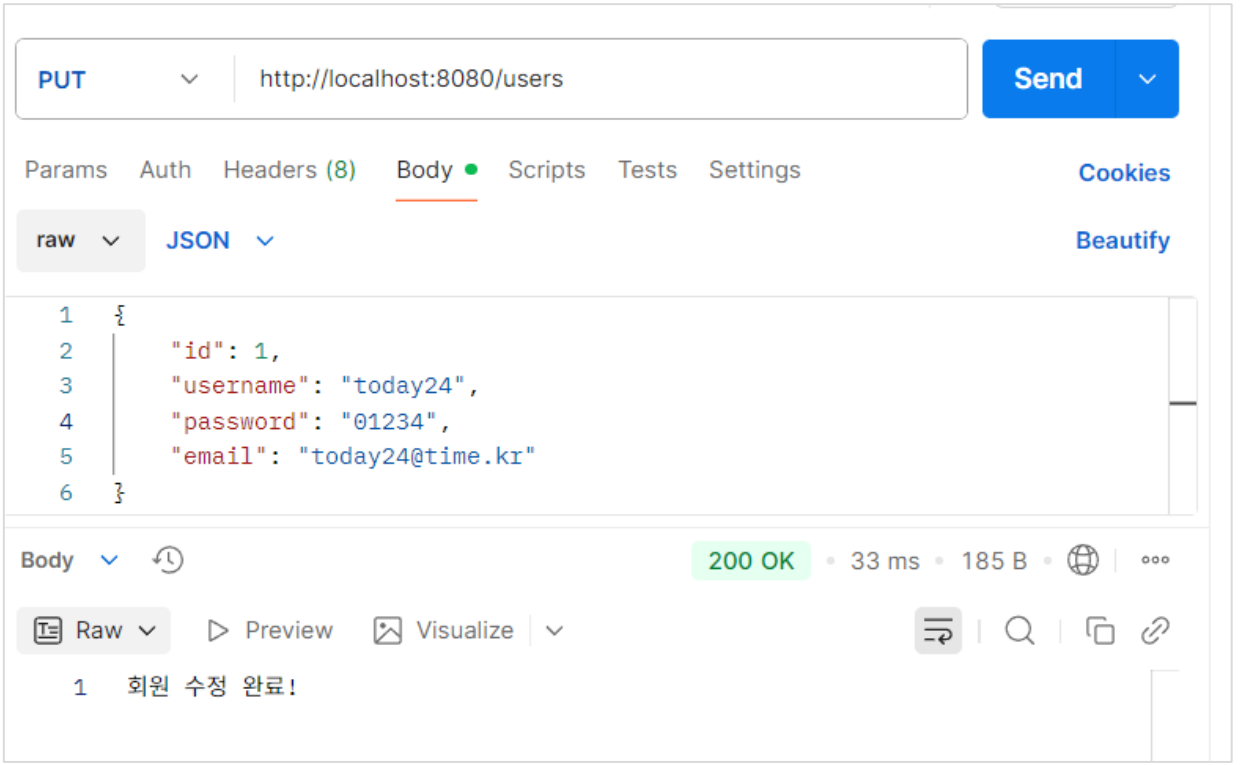
회원 관리 API

- 회원 삭제(id - 2)



회원 관리 API

- 회원 수정(id - 1)



회원 관리 API

- 회원 목록 – 웹 브라우저로 보기