


[개정판] Spring Boot를 이용한 RESTful API 개발

{ REST API } +  Spring
Boot

```
class Book {
    def setData(self, title, price, author):
        self.title = title
        self.price = price
        self.author = author
}

var fs = require('fs');
fs.readFile('./JONE.txt', '* 1 *',
    function (err, data) {
        console.log(data); // 3
    });

@interface NextInnovationDelegate : NSObject <UIApplicationDelegate>
```

- Section 0: Web Service & Web Application
- Section 1: Spring Boot로 개발하는 RESTful API
- Section 2: User Service API 추가
- Section 3: RESTful Service 기능 확장
- Section 4: Spring Boot API 사용
- **Section 5: JPA 사용**
- Section 6: REST API 설계 가이드

5. Section

Java Persistence API 사용

- JPA 개요
- Entity 설정
- H2 Console 사용을 위한 SecurityConfig 파일 수정
- JPA Service를 위한 GET/POST/DELETE 메소드 추가
- Post Entity 추가와 User Entity와의 관계설정

■ JPA

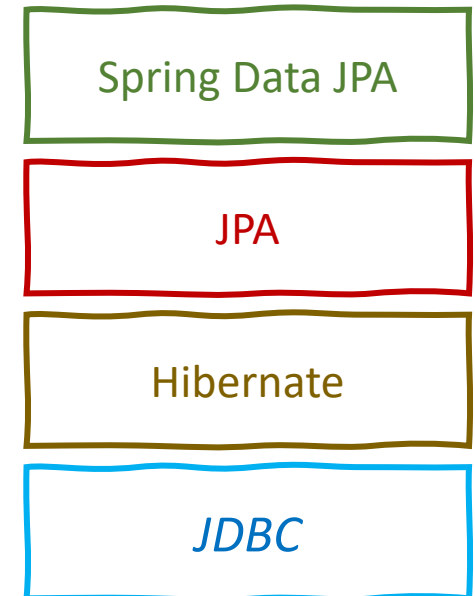
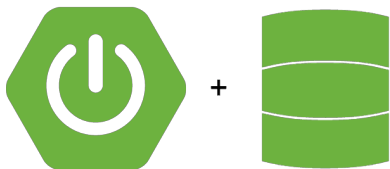
- Java Persistence API
- 자바 ORM 기술에 대한 API 표준 명세
- 자바 어플리케이션에서 관계형 데이터베이스를 사용하는 방식을 정의한 인터페이스
- EntityManager를 통해 CRUD 처리

■ Hibernate

- JPA의 구현체, 인터페이스를 직접 구현한 라이브러리
- 생산성, 유지보수, 비종속성

■ Spring Data JPA

- Spring Module
- JPA를 추상화한 Repository 인터페이스 제공



```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

pom.xml

- <http://localhost:8088/h2-console>

English

[Preferences](#) [Tools](#) [Help](#)

Login

Saved Settings:

Setting Name:

Driver Class:

JDBC URL:

User Name:

Password:

Test successful

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@JsonIgnoreProperties(value = {"password", "ssn"})
@Schema(description = "사용자 상세 정보를 위한 도메인 객체")
@Entity
public class User {
    @Schema(title = "사용자 ID", description = "사용자 ID는 자동 생성됩니다.")
    @Id
    @GeneratedValue
    private Integer id;
```

User.java

Table structure

```
create table user (
    id integer not null,
    join_date timestamp,
    name varchar(255),
    password varchar(255),
    ssn varchar(20),
    primary key (id)
)
```

→ Project 실행 시 데이터베이스 자동 생성

■ 테이블 생성 시 오류 발생

Syntax error in SQL statement "create table [*]user (id integer not null, join_date timestamp, name varchar(255), pwd varchar(255), ssn varchar(255), primary key (id))"; expected "identifier"; SQL statement:

create table user (id integer not null, join_date timestamp, name varchar(255), pwd varchar(255), ssn varchar(255), primary key (id)) [42001-214] 42001/42001 ([Help](#))

JPA - Entity 설정과 초기 데이터 생성 2/5

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@JsonIgnoreProperties(value = {"password", "ssn"})
@Schema(description = "사용자 상세 정보를 위한 도메인 객체")
@Entity
@Table(name = "users")
public class User {
    @Schema(title = "사용자 ID", description = "사용자 ID는 자동 생성됩니다.")
    @Id
    @GeneratedValue
    private Integer id;
```

User.java

jdbc:h2:mem:testdb
+ USERS
+ INFORMATION_SCHEMA
+ Sequences
+ Users
i H2 2.1.214 (2022-06-13)

```
Hibernate: drop table if exists users cascade
Hibernate: drop sequence if exists users_seq
Hibernate: create sequence users_seq start with 1 increment by 50
Hibernate: create table users (id integer not null, join_date timestamp(6), name varchar(255), password varchar(255), ssn varchar(255), primary key (id))
```

JPA - Entity 설정과 초기 데이터 생성 3/5

```
insert into users(id, join_date, name, password, ssn) values(90001, now(), 'User1', 'test111', '701010-1111111');  
insert into users(id, join_date, name, password, ssn) values(90002, now(), 'User2', 'test222', '801111-2222222');  
insert into users(id, join_date, name, password, ssn) values(90003, now(), 'User3', 'test333', '901111-1222222');
```

resources/data.sql

```
Caused by: org.h2.jdbc.JdbcSQLSyntaxErrorException: Table "USERS" not found (this database is empty); SQL statement:  
insert into users(id, join_date, name, password, ssn) values(90001, now(), 'User1', 'test111', '701010-1111111') [42104-214]  
    at org.h2.message.DbException.getJdbcSQLException(DbException.java:502) ~[h2-2.1.214.jar:2.1.214]  
    at org.h2.message.DbException.getJdbcSQLException(DbException.java:477) ~[h2-2.1.214.jar:2.1.214]
```

```
jpa:  
  hibernate:  
    ddl-auto: create-drop  
  show-sql: true  
  defer-datasource-initialization: true
```

application.yml

JPA - Entity 설정과 초기 데이터 생성 4/5

← → ↻ ⓘ localhost:8088/h2-console/login.jsp?jsessionid=e307a92f4af2e2d35a0a5df7ebe7ac5c

English ▾ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▾

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:testdb jdbc:h2:mem:

User Name: sa

Password:

Connect Test Connection

jdbc:h2:mem:testdb

- + USERS
- + INFORMATION_SCHEMA
- + Sequences
- + Users
- ⓘ H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM USERS

SELECT * FROM USERS;

ID	JOIN_DATE	NAME	PASSWORD	SSN
90001	2023-11-21 15:36:20.683344	User1	test111	701010-1111111
90002	2023-11-21 15:36:20.683883	User2	test222	801111-2222222
90003	2023-11-21 15:36:20.684024	User3	test333	901111-1222222

(3 rows, 1 ms)

- spring security 문제

```
@Configuration
public class SecurityConfig {

    @Bean
    UserDetailsService userDetailsService() {...}

    @Bean
    BCryptPasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Bean
    public WebSecurityCustomizer webSecurityCustomizer() {
        return (web) → web.ignoring().
            requestMatchers(new AntPathRequestMatcher( pattern: "/h2-console/**"));
    }
}
```

SecurityConfig.java

JPA – Controller, Repository 생성 1/5

```
@Repository
public interface UserRepository extends JpaRepository<User, Integer> {
}
```

UserRepository.java

```
@RestController
@RequestMapping("/jpa")
public class UserJPAController {

    private UserRepository userRepository;

    public UserJPAController(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @GetMapping("/users")
    public List<User> retrieveAllUsers() {
        return userRepository.findAll();
    }
}
```

UserJPAController.java

GET 127.0.0.1:8088/jpa/users

Params Auth Headers (8) Body Pre-req. Tests Settings

Body Cookies (1) Headers (14) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
[
  {
    "id": 90001,
    "name": "User1",
    "joinDate": "2023-11-21T07:44:56.850+00:00"
  },
  {
    "id": 90002,
    "name": "User2",
    "joinDate": "2023-11-21T07:44:56.850+00:00"
  },
  {
    "id": 90003,
    "name": "User3",
    "joinDate": "2023-11-21T07:44:56.851+00:00"
  }
]
```

JPA – Controller, Repository 생성 2/5

```
@GetMapping("/users/{id}")
public ResponseEntity retrieveUser(@PathVariable int id) {
    Optional<User> user = userRepository.findById(id);

    if (!user.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    EntityModel entityModel = EntityModel.of(user.get());
    WebMvcLinkBuilder linkTo = linkTo(methodOn(this.getClass()).retrieveUser(id));
    entityModel.add(linkTo.withRel("all-users"));

    return ResponseEntity.ok(entityModel);
}
```

UserJPAController.java

GET 127.0.0.1:8088/jpa/users/90001

Params Auth Headers (8) Body Pre-req. Tests Settings

Body Cookies (1) Headers (14) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "name": "User1",
3   "joinDate": "2023-11-21T07:50:16.467+00:00",
4   "_links": {
5     "all-users": {
6       "href": "http://127.0.0.1:8088/jpa/users"
7     }
8   }
9 }
```

JPA – Controller, Repository 생성 3/5

```
@DeleteMapping("/{id}")
public void deleteUser(@PathVariable int id) {
    userRepository.deleteById(id);
}
```

UserRJPAController.java

```
@Bean
protected SecurityFilterChain filterChain(HttpSecurity http, HandlerMappingIntrospector introspector) throws Exception {
    http.csrf(AbstractHttpConfigurer::disable);

    return http.build();
}
```

SecurityConfig.java

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** 127.0.0.1:8088/jpa/users/90001
- Authorization:** Basic Auth
- Username:** user
- Password:** passw0rd
- Status:** 200 OK

The interface also includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The bottom section shows tabs for Body, Cookies (1), Headers (10), and Test Results, with a Pretty/Raw/Preview/Visualize view selector and a Text dropdown.

JPA - Controller, Repository 생성 4/5

```
@PostMapping("/users")
public ResponseEntity<User> createUser(@Valid @RequestBody User user) {
    User savedUser = userRepository.save(user);
    // CREATED
    // /users/4
    URI location = ServletUriComponentsBuilder
        .fromCurrentRequest()
        .path("/{id}")
        .buildAndExpand(savedUser.getId())
        .toUri();

    return ResponseEntity.created(location).build();
}
```


UserJPAController.java


POST 127.0.0.1:8088/jpa/users

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

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

```
1 {
2   ... "name": "Dowon",
3   ... "joinDate": "2023-11-20T03:21:32.896+0000",
4   ... "password": "test4",
5   ... "ssn": "701010-1234567"
6 }
```

Body Cookies (1) Headers (11) Test Results  Status: 201 Created

Pretty Raw Preview Visualize Text ▾ 

Quiz) 사용자 전체 목록보기에서 사용자 수를 함께 반환하도록 API 수정

예시)

```
{
  "count": 4,
  "users": [
    {
      "name": "Dowon",
      "joinDate": "2023-11-20T03:21:32.896+00:00",
      "password": "test4",
      "ssn": "701010-1234567"
    },
    {
      "name": "User1",
      "joinDate": "2023-11-21T08:18:10.428+00:00",
      "password": "test111",
      "ssn": "701010-1111111"
    },
    {
      "name": "User2",
      "joinDate": "2023-11-21T08:18:10.429+00:00",
      "password": "test222",
      "ssn": "801111-2222222"
    },
  ]
}
```


JPA - 게시물 관리

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Post {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String description;

    @ManyToOne(fetch = FetchType.LAZY)
    @JsonIgnore
    private User user;
}
```

Post.java

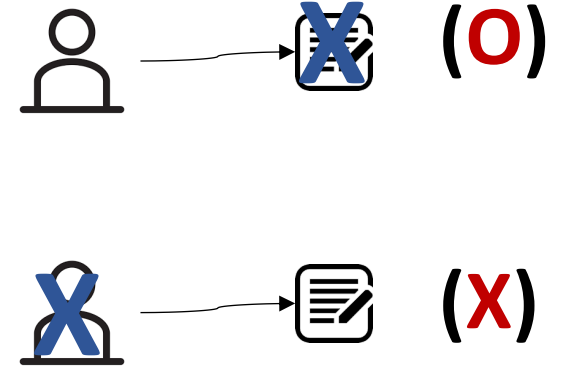
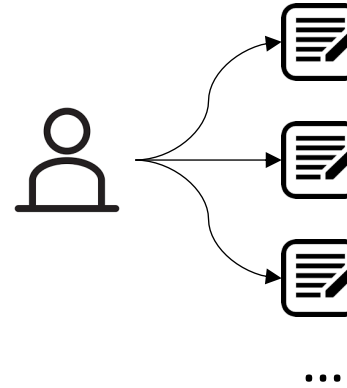
User:Posts → 1:N

```
@OneToMany(mappedBy = "user")
```

```
private List<Post> posts;
```

```
public User(Integer id, @Size(min = 2, message = "Name은 2글자 이상 입력해 주세요.") String name, @Past Date joinDate,
             String password, String ssn) {
    this.id = id;
    this.name = name;
    this.joinDate = joinDate;
    this.password = password;
    this.ssn = ssn;
}
```

User.java



JPA - 게시물 관리

```
insert into post(description, user_id) values('My first post', 90001);  
insert into post(description, user_id) values('My second post', 90001);
```

data.sql

```
Hibernate: drop table if exists post cascade  
Hibernate: drop table if exists users cascade  
Hibernate: drop sequence if exists users_seq  
Hibernate: create sequence users_seq start with 1 increment by 50  
Hibernate: create table post (id integer generated by default as identity, user_id integer, description varchar(255), primary key (id))  
Hibernate: create table users (id integer not null, join_date timestamp(6), name varchar(255), password varchar(255), ssn varchar(255), primary key (id))  
Hibernate: alter table if exists post add constraint FK7ky67sg17k0ayf22652f7763r foreign key (user_id) references users
```

jdbc:h2:mem:testdb

+

 POST

+

 USERS

+

 INFORMATION_SCHEMA

+

 Sequences

+

 Users

i

 H2 2.1.214 (2022-06-13)

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT * FROM POST|

SELECT * FROM POST;

ID	USER_ID	DESCRIPTION
1	90001	My first post
2	90001	My second post

(2 rows, 2 ms)

JPA - 게시물 목록조회

```
@GetMapping("/users/{id}/posts")
public List<Post> retrieveAllPostsByUser(@PathVariable int id) {
    Optional<User> user = userRepository.findById(id);
    if (!user.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    return user.get().getPosts();
}
```

UserJPAController.java

GET http://127.0.0.1:8088/jpa/users/90002/posts

Body Cookies (1) Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 []
```

GET http://127.0.0.1:8088/jpa/users/90001/posts

Body Cookies (1) Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 10001,
4     "description": "My first post"
5   },
6   {
7     "id": 10002,
8     "description": "My second post"
9   }
10 ]
```

GET http://127.0.0.1:8088/jpa/users/90009/posts

Body Cookies (1) Headers (8) Test Results Status: 404 Not Found

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2020-03-23T07:26:31.820+0000",
3   "message": "id-90009",
4   "details": "uri=/jpa/users/90009/posts"
5 }
```

JPA - 게시물 추가

```
@Repository
public interface PostRepository extends JpaRepository<Post, Integer> {
}
```

Post.java

```
@PostMapping("/users/{id}/posts")
public ResponseEntity<Post> createPost(@PathVariable int id, @RequestBody Post post) {
    Optional<User> userOptional = userRepository.findById(id);
    if (!userOptional.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    User user = userOptional.get();

    post.setUser(user);

    postRepository.save(post);

    URI location = ServletUriComponentsBuilder
        .fromCurrentRequest()
        .path("/{id}")
        .buildAndExpand(post.getId())
        .toUri();

    return ResponseEntity.created(location).build();
}
```

UserJPAController.java

JPA - 게시물 추가

POST

127.0.0.1:8088/jpa/users/90002/posts

ParamsAuthorizationHeaders (9)Body •Pre-request ScriptTestsSettings

noneform-datax-www-form-urlencoded

1{

2... "description":

3}

BodyCookies (1)Headers (11)

PrettyRawPreview

jdbc:h2:mem:testdb

POST

USERS

INFORMATION_SCHEMA

Sequences

Users

H2 2.1.214 (2022-06-13)

RunRun SelectedAuto completeClearSQL statement:

SELECT * FROM POST

SELECT * FROM POST;

ID	USER_ID	DESCRIPTION
1	90001	My first post
2	90001	My second post
3	90002	My New Blog

(3 rows, 2 ms)

2{

3" id": 3,

4" description": "My New Blog"

5}

6}

Settings

Status: 200 OK