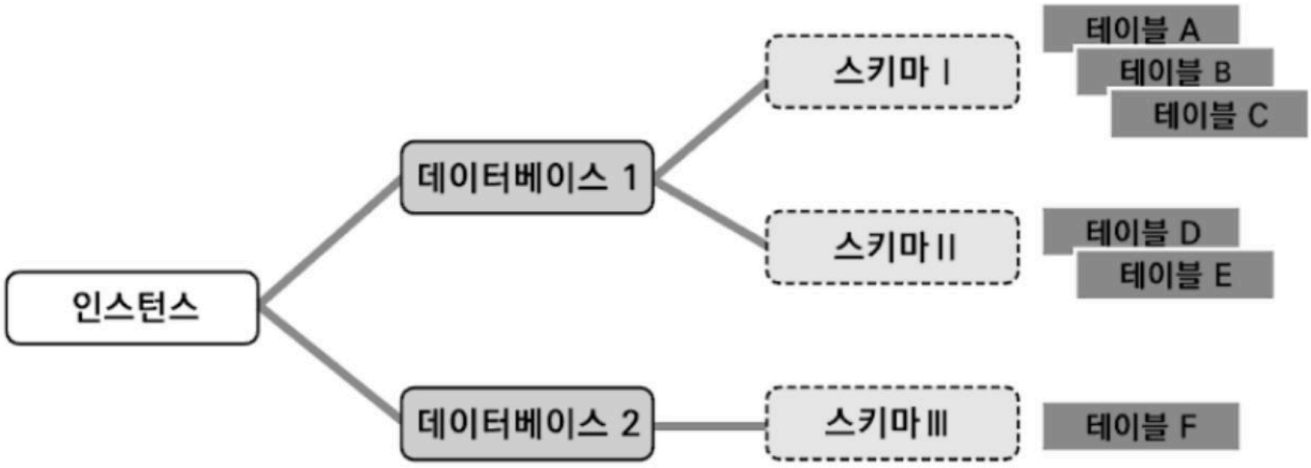
멋사 4주차 온라인 강의

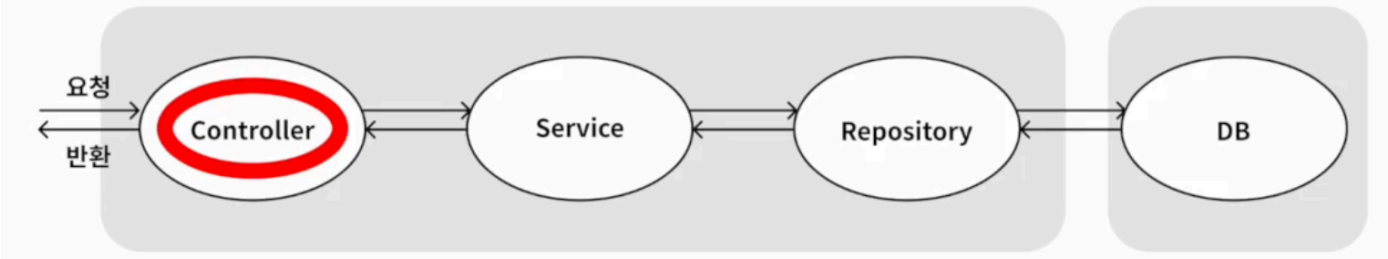
- resouce 폴더
  - 리소르 파일 관리
- application.properties
  - 설정 파일
  - 애플리케이션 설정, 메시지 리소스, DB 연결 정보 등 저장
- .gitignore
  - Git이 버전 관리에서 제외할 파일 또는 디렉토리를 지정하는 설정 파일
- build.gradle
  - Gradle 빌드 도구를 사용하여 프로젝트를 설정하고 빌드하는 데 사용하는 파일
- https://www.toptal.com/developers/gitignore
- application.properties 에서 설정 작성

```
spring.application.name=likelion

#DB(local)
//스키마명 spring.datasource.url=jdbc:mysql://localhost:3306/likelion
spring.datasource.username=root
//MySQL 계정정보
spring.datasource.password=1234

//sql예쁘게보여줄거에요
#showSQL
spring.jpa.properties.hibernate.format_sql=true

//기존스키마가있으면 삭제하고 다시 생성
springjpa.hibernate.ddl-auto=create
```

- 스키마와 테이블



2. Controller 와 CostomApiResponse



Get http://localhost:8080/api/simpleText/success

- HttpStatus: 200(OK)
- ResponseBody(Plain text): 안녕

1. TestController
    - @RestController: 해당 클래스가 RESTful 웹 서비스의 Controller 임을 나타냄
    - @RequestMapping("/경로"): 해당 메소드가 지정된 경로에 요청을 처리하는데 사용됨을 명시
    - @GetMapping("/경로"): GET 메소드로 들어온 요청을 받음
        - (PostMapping, @PutMapping, @DeleteMapping 등 존재)

2. ResponseEntity 이해하기
    - HTTP 응답을 나타내는 객체
    - 응답의 상태 코드, 헤더 및 본문 데이터를 포함할 수 있음
    - 클라이언트에게 보낼 응답을 생성하고 정의하는 데 유용
    - 사용 예시

- **ResponseEntity.status(HttpStatus.BAD_REQUEST).body("실패");**
  응답 상태 코드 지정 응답 body에 들어갈 데이터

- 제네릭(Generic) 이란?
    - 제네릭 타입은 클래스 또는 메소드를 작성 할 때 일반적인 데이터 타입을 지정하는 방법을 제공
    - 이를 통해 코드의 유연성과 재사용성을 높일 수 있음
    - 제네릭을 사용하면 클래스나 메소드를 선언할 때 데이터 타입을 파라미터로 받아들이고, 실제 사용될 때 그 데이터 타입을 결정

```
public interface List<E> extends Collection<E>{}

public class ResponseEntity<T> extends HttpEntity<T>{}
```

- 제네릭 예시
    - List 는 제네릭 클래스

- List 를 선얼 할 때 원하는 데이터 타입을 명시할 수 있음
- 예를 들어, List< String>은 문자열만을 저장하는 리스트를 나타냄

```
List<String> stringList = new ArrayList<>();
stringList.add("Hello");
stringList.add("World");
```

- 이렇게 하면 stringList에는 문자열만 추가할 수 있음
- 컴파일러가 타입 일치를 확인하기 때문에 타입 안정성이 보장

3. CustomApiResponse 클래스
    - 우리가 작성할 클래스
    - 모든 API에 대해 일괄적인 형태의 응답을 내려주기 위함
    - Data: 클라이언트가 원하는 데이터
        - Ex) 회원 조회 API라면 회원정보가 들어가야함
    - Message: 서버 측에서 클라이언트에게 보낼 결과 메시지
        - 회원조회 API가 성공한 경우 ex)회원 조회에 성공했습니다.
        - 회원가입 API에서 사용자 id 값이 비어있는 채로 요청이 온 경우 ex)사용자 id는 필수 값입니다.

- **회원 정보 API 응답 예시**

```
{
    "status" : 200,
    "data" : {
        "userId" : "example",
        "email" : "example@naver.com"
    }
    "message" : "회원 조회에 성공했습니다."
}
```

- **회원 가입 API 응답 예시**

```
{
    "status" : 200,
    "data" : null,
    "message" : "회원 가입에 성공했습니다."
}
```

1. 공통적으로 status, data, message 필드를 갖고 있어야함
2. JSON 형식의 데이터를 리턴해야함
3. CustomApiResponse는 ResponseEntity의 ResponseBody에 들어가게 됨
   ResponseEntity<CustomApiResponse<?>>

- DTO (Data Transfer Object)
    - 데이터를 전송하기 위한 객체
    - 순수한 데이터만 존재
    - Lombok 어노테이션 사용
        - @Getter
        - Setter
        - Builder
        - NoArgsConstructor

        - AllArgsConstructor

3. CustomErrorController와 GlobalExceptionHandler

1. CustomErrorController
    - ErrorController: 인터페이스를구현
    - @RestController: RESTful 웹 서비스의 컨트롤러임을 나타냄
    - @RequestMapping("/error"): /error 경로로 들어오는 요청을 이 클래스의 메소드로 매핑
    - handleError 메소드: HttpServletRequest를 매개변수로 받아서, 요청에 대한 에러 상태 코드를 확인 후 적절한 응답을 반환
    - HttpServletRequest: SpringMVC에서 요청에 대한 정보를 전달하는데 사용되는 클래스
    - Object status = request.getAttribute(RequestDispatcher.ERROR_STATUS_CODE); : 현재 요청에서 발생한 오류 코드를 받아올 수 있음
2. GlobalExceptionHandler
    - ex) 회원가입 API를 개발한다고 생각
        - POST http://localhost:8080/api/user/signup
            - 회원가입을 할때 , userId, email, password를 객체로 받자(@RequestBody, Dto)
            - 클라이언트로부터 받은 값에 대한 검증이 필요 (Validation, @Vaild)
            - 응답이 우리가 원하는대로 오는가? CustomErrorController에 걸림
            - Validation 메시지를 CustomApiResponse의 message 필드에 넣어보자
    - 특정 유형의 예외를 처리하는 클래스
    - @ControllerAdvice: 이 클래스가 전역 예외 처리를 담당함을 나타냄
    - @ExceptionHandler: 처리할 예외 클래스 작성

4. 회원가입 API 작성

@Entity

@Table

@Id

@GeneratedValue

@Column

@Getter

@NoArgsConstructor

@AllArgsConstructor

@Builder

- JPARespository 인터페이스
  - Spring Data JPA 프레임워크에서 제공하는 인터페이스
  - 기본적은 CRUD(Create, Read, Updata, Delete) 작업을 수행하는 메서드를 제공
  - 사용방법: extends JPARepository<T, ID>
- UserRepository 인터페이스는 JPARepository 인터페이스를 상속
  - @Repository
- Service 계층
  - 비즈니스 로직이 포함된 계층
  - @Service
  - @RequiredArgsConstructor
- Controller 계층
  - @RestController
  - @RequestMapping
  - @ReairedArgsConstructor
- BaseEntity
  - 데이터 생성 일시, 수정 일시와 같이 모든 테이블에 공통적으로 있는 필드들을 정의
  - createAt, updatedAt
  - @MappedSuperclass 사용
  - @EntityListeners(AuditingEntityListener.class)
  - @CreatedDate
  - @LastModifiedDate
  - EnableJpaAuditing (Main 클래스에)