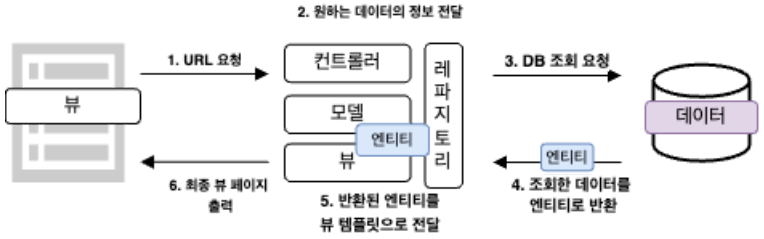
5️⃣ 게시글 읽기: Read

5.1 데이터 조회 과정



① 사용자가 데이터를 조회해 달라고 웹 페이지에서 URL 요청을 보냄

② 서버의 컨트롤러가 요청을 받아 해당 URL에서 찾으려는 데이터 정보를 리파지터리에 전달

③ 리파지터리는 정보를 가지고 DB에 데이터 조회를 요청

④ DB는 해당 데이터를 찾아 이를 엔티티로 반환

⑤ 반환된 엔티티는 모델을 통해 뷰 템플릿으로 전달됨

⑥ 최종적으로 결과 뷰 페이지가 완성돼 사용자의 화면에 출력됨

5.2 단일 데이터 조회하기

1) URL 요청받기

Article 조회를 위해 article/id로 URL 요청을 했을 때 받아 줄 컨트롤러 만들기

- com.example.firstproject > controller > ArticleController
- 클래스 안 작성된 코드 밑에 @GetMapping() 어노테이션 작성
  - ex) id에 따라 /articles/1, /articles/2 등으로 받고자 하면 “/articles/{id}”를 괄호 안에 입력
    - 중괄호 안에 id를 써주면 id는 변수로 사용됨
- URL 요청을 받아 수행할 show() 메서드 생성
  - 매개변수는 URL에 있는 id를 가져오고자 id 앞에 `@PathVariable` 어노테이션 사용
    - @PathVariable 어노테이션 : URL 요청으로 들어온 전달값을 컨트롤러의 매개변수로 가져옴
  - 컨트롤러가 id를 잘 받았는지 확인하기 위해 log.info(”id = “ + id);를 메서드에 추가

2) 데이터 조회해 출력하기

- **id를 조회해 DB에서 해당 데이터 가져오기**

    - 리파지터리로 DB에 저장된 데이터 가져오기
        - articleRepository.findById(id).orElse(null)로 데이터를 찾아 Article 타입의 articleEntity 변수에 저장
            - **findById()** : JPA의 CrudRepository가 제공하는 메서드, 특정 엔티티의 id 값을 기준으로 데이터를 찾아 Optional 타입으로 반환
            - orElse(null) : id 값으로 데이터를 찾을 때 해당 id 값이 없으면 null을 반환
- **가져온 데이터를 모델에 등록하기**
    - MVC 패턴에 따라 조회한 데이터를 뷰 페이지에서 사용하기 위해 데이터를 모델에 등록
    - show() 메서드의 매개변수로 model 객체 받아옴
    - 모델에 데이터 등록 : **addAttribute()** 메서드 사용
        - 형식 : model.addAttribute(String name, Object value)
        - ex) model.addAttribute(”article”, articleEntity);
- **조회한 데이터를 사용자에게 보여 주기 위한 뷰 페이지 만들고 반환하기**
    - articles 디렉터리 안에 show 파일이 있다고 가정하고 return “articles/show”;
    - show.mustache 파일 만들기
        - resources > templates > articles에서 show.mustache 파일 생성
        - 헤더, 푸터 삽입
        - 헤더와 푸터 사이에 부트스트랩에서 테이블 소스 코드 가져와 삽입
        - 모델에 등록한 article을 뷰 페이지에서 머스테치 문법인 이중중괄호({{}})를 이용해 출력
            - {{#article}}로 열고 {{/article}}로 닫아 데이터 사용할 범위 지정
            - article에 담긴 id, title, content도 이중중괄호를 이용해 가져옴
- 추가 단계: 기본 생성자 추가하기
    - entity > Article 파일
    - 기본생성자 생성을 위해 `@NoArgsConstructor` 어노테이션 추가
        - @NoArgsConstructor 어노테이션 : 기본 생성자를 추가

5.3 데이터 목록 조회하기

- 단일 데이터 조회와 데이터 목록 조회의 차이
    - 단일 데이터 조회 : 리파지터리가 **엔티티**를 반환
    - 데이터 목록 조회 : 리파지터리가 엔티티의 묶음인 **리스트**를 반환

1) URL 요청받기

- ArticleController의 show() 메서드 아래에 index() 메서드 추가
- index() 메서드에 @GetMapping(”/articles”) 선언 ⇒ URL 요청 받게 함

2) 데이터 조회해 출력하기

- **DB에서 모든 Article 데이터 가져오기**
  - articleRepository.findAll() 수행하고 결과를 articleEntityList로 받음
    - **findAll()** : 해당 리파지터리에 있는 모든 데이터를 가져오는 메서드
      - Iterable 데이터 타입으로 반환 ⇒ List와 불일치
        - 해결법 1) 캐스팅(형변환)을 이용 : Iterable<Article>을 List<Article>로 다운캐스팅
          - 캐스팅 : 데이터 타입을 변환하는 것
          - articleRepository.findAll() 앞에 '(List<Article>)' 붙이기
        - 해결법 2) 캐스팅(형변환)을 이용 : List<Article>을 Iterable<Article>로 업캐스팅
          - articleEntityList 앞의 List<Article>을 'Iterable<Article>'로 변경
        - 해결법 3) findAll() 메서드가 Iterable이 아닌 ArrayList를 반환하도록 수정
          - com.example.firstproject > repository > ArticleRepository
          - ArticleRepository가 상속 받는 CrudRepository를 오버라이딩
          - ArrayList<Article> findAll();로 수정
    - articleEntityList의 타입은 List<Article>로 설정
- **가져온 Article 묶음을 모델에 등록하기**
  - index() 메서드의 매개변수로 model 객체를 받아 옴
  - model.addAttribute() 메서드로 전달할 데이터 묶음인 articleEntityList를 "articleList" 이름으로 등록
- **사용자에게 보여 줄 뷰 페이지 설정하기**
  - return "articles/index";
  - index.mustache 파일 만들기
    - resources > templates > articles 에 index.mustache 파일 생성
    - 헤더, 푸터 삽입
    - 헤더와 푸터 사이에 show.mustache 파일에서 코드 가져와 삽입
    - 가져온 코드에서 {{#article}}을 {{#articleList}}로, {{/article}}을 {{/articleList}}로 수정

      * 머스테치 변수가 데이터 묶음일 경우에는 그 안쪽의 코드를 반복 실행함