7️⃣

게시글 수정하기: Update

7.1 데이터 수정 과정

데이터 수정 2단계

1. <수정 페이지> 만들고 기존 데이터 불러오기
    - <상세 페이지>에서 [Edit] 버튼 클릭
    - 요청을 받은 컨트롤러는 해당 글을 id로 DB에서 데이터를 찾아 가져옴
    - 컨트롤러는 가져온 데이터를 뷰에서 사용할 수 있도록 모델에 등록
    - 모델에 등록된 데이터를 <수정 페이지>에서 보여줌 ⇒ 사용자가 내용 수정 가능한 상태가 됨
2. 데이터를 수정해 DB에 반영한 후 결과를 볼 수 있게 <상세 페이지>로 리다이렉트하기
    - 폼 데이터(수정 요청 데이터)를 DTO에 담아 컨트롤러에서 받음
    - DTO를 엔티티로 변환
    - DB에서 기존 데이터를 수정 데이터로 갱신
    - 수정 데이터를 <상세 페이지>로 리다이렉트

7.2 <수정 페이지> 만들기

1) <상세 페이지>에 Edit 버튼 만들기

- show.mustache
- </table> 아래에 <a> 태그 추가, href 속성 값으로 연결하려는 URL 작성, 링크 걸 텍스트는 Edit으로 설정
    - ex) Edit

      💫 article 사용 범위를 {{#article}}{/article}} 형식으로 지정한 경우 {{id}}로 사용, 범위를 지정하지 않았다면 {{article.id}}라고 표시해야 함
- 버튼 디자인 수정 : 부트스트랩 CSS를 적용해 class="btn btn-primary" 속성을 추가
    - ex) Edit

2) Edit 요청을 받아 데이터 가져오기

- edit() 메서드 기본 틀 만들기
    - ArticleController.java의 indexx() 메서드 아래에 edit() 메서드 추가
        - 수정 요청을 받아 처리할 edit() 메서드 작성, 수정 페이지를 articles 디렉터리 안에 edit.mustache로 설정
        - @GetMapping("/articles/{id}/edit")

💫뷰 페이지에서 변수를 사용할 때는 중괄호 2개, 컨트롤러에서 URL 변수를 사용할 때는 중괄호 1개임을 유의

- 수정할 데이터 가져오기
  - articleRepository의 findById(id) 메서드로 데이터 가져오기
    - 데이터를 찾지 못하면 null을 반환, 찾았다면 Article 타입의 articleEntity로 저장
  - id를 메서드의 매개변수로 받아와 자료형은 Long으로 작성, 데이터 타입 앞에 @PathVariable 어노테이션 추가
- 모델에 데이터 등록하기
  - 메서드의 매개변수로 model 객체 받아오기
  - addAttribute() 메서드로 모델에 데이터 등록
    - article이라는 이름으로 articleEntity를 등록

      ⇒ DB에서 가져온 데이터를 article이라는 이름으로 뷰 페이지에서 사용 가능

3) 수정 폼 만들기

- src > main > resources > templates > articles에 edit.mustache 생성
- 헤더-푸터 구조 만듦
- new.mustache의 <form class="container" ...>...</form> 복사해 붙여넣기
  - <form> 태그의 action 속성 값을 공백으로 만듦
  - Back 링크를 누르면 상세 페이지로 돌아가도록 href 속성 값을 "/articles/{{article.id}}"로 수정
- 수정 페이지에 내용이 보이도록 글의 제목은 <input> 태그의 value="{{article.title}}" 속성 추가, 글의 내용은 <textarea> 태그의 콘텐츠 영역에 {{article.content}} 추가
  - ex) <input type="text" class="form-control" name="title" value="{{aritlce.title}}">
  - ex) <textarea class="form-control" rows="3" name="content">{{article.content}} </textarea>

7.3 수정 데이터를 DB에 갱신하기

데이터를 수정해 DB에 반영한 후 결과를 볼 수 있게 상세 페이지로 리다이렉트하기

- 폼 데이터 전달 → DTO를 엔티티로 변환 → DB 갱신 → 리다이렉트
- 클라이언트와 서버 간 데이터 처리를 위한 4가지 기술
  - MVC(Model-View-Controller) : 서버 역할을 분담해 처리하는 기법
  - JPA(Java Persistentce API) : 서버와 DB 간 소통에 관여하는 기술
  - SQL(Structured Query Language) : DB 데이터를 관리하는 언어
  - HTTP(HyperText Transfer Protocol) : 데이터를 주고받기 위한 통신 규약

1) HTTP 메서드

- 프로토콜(protocol) : 컴퓨터 간에 원활하게 통신하기 위해 사용하는 전 세계 표준

- 기기 간 각종 신호 처리 방법, 오류 처리, 암호, 인증 방식 등 규정
- HTTP(HyperText Transfer Protocol) : 웹 서비스에 사용하는 프로토콜
  - 클라이언트의 다양한 요청을 메서드를 통해 서버로 보내는 역할
  - 메서드
    - POST : 데이터 생성 요청
    - GET : 데이터 조회 요청
    - PATCH(PUT) : 데이터 수정 요청
    - DELETE : 데이터 삭제 요청

    ⇒ CRUD(Create Read Update Delete)

데이터 관리	SQL	HTTP
데이터 생성(Create)	INSERT	POST
데이터 조회(Read)	SELECT	GET
데이터 수정(Update)	UPDATE	PATCH(PUT)
데이터 삭제(Delete)	DELETE	DELETE

2) 더미 데이터 설정하기

- 서버 재시작 마다 데이터 입력이 번거로우므로 더미 데이터 생성
- src > main > resources 에 data.sql 생성
- INSERT 문으로 데이터 삽입
- src > main > resources > application.properties에 spring.jpa.defer-datasource-initialization=true 옵션 추가

3) <수정 페이지> 변경하기

- edit.mustache
- <form> 태그의 action 속성은 URL 지정, method 속성은 보내는 방식 지정
  - action 속성 값으로 “/articles/update”, method 속성 값으로 “post” 설정

    💫 <form> 태그가 GET과 POST만 지원하므로 PATCH(수정)이 아닌 POST(새로 생성) 사용
- 수정 폼 서버로 id도 보내야 하므로 <input> 태그를 만들어 value=”{{id}}” 속성 추가
  - id는 화면에 표시할 필요 없으므로 type=”hidden” 속성 추가
  - ex) <input name=”id” type=”hidden” value=”{{id}}”>

4) 수정 데이터 받아 오기

- update() 메서드 기본 틀 만들기
  - ArticleController.java에서 edit() 메서드 아래에 작성
  - 데이터 수정 요청을 받는 메서드 이름을 update()로 하고 기본틀 작성

    - URL 요청 접수
        - @PostMapping() 사용, 괄호 안에는 “/articles/update” 작성
- 수정 데이터를 DTO에 담기
    - update() 메서드의 매개변수로 DTO 받아 오기
    - 수정 폼에 <input> 태그로 id 추가했으므로 ArticleForm에도 id 추가 필요
        - dto > ArticleForm
        - id 필드 추가
        - 생성자에도 id 작성
    - 수정 데이터 잘 받았는지 로그 찍어 확인

5) DB에 저장하고 결과 페이지로 리다이렉트하기

- DTO를 엔티티로 변환하기
    - form.toEntity() 메서드를 호출해 반환값을 Article 타입의 articleEntity로 받기
    - DTO가 엔티티로 잘 변환됐는지 확인 로그 찍기
- 엔티티를 DB에 저장하기
    - DB에서 기존 데이터 가져오기 : articleRepository.findById() 메서드 호출
        - findById() 메서드 : 리파지터리가 자동으로 제공하는 메서드, 괄호 안에는 찾는 id 값을 작성
            - 앞의 articleEntity에 getId() 메서드를 호출해 id 값 넣기
        - 반환 데이터를 Article 타입의 target에 저장
        - 데이터가 없다면 null 반환
    - 기존 데이터의 값 갱신
        - ex)

```
if (target != null) {
    articleRepository.save(articleEntity);
}
```

- 수정 결과 페이지로 리다이렉트하기
    - 컨트롤러의 마지막 return 값으로 해당 id의 상세 페이지로 이동할 수 있게 URL 작성
        - id는 엔티티에 따라 바뀌어야 하므로 articleEntity의 getId() 메서드 호출 → 앞의 URL과 + 연산자로 연결
            - ex) return “redirect:/articles/” + articleEntity.getId();

6) SQL 문으로 직접 DB 갱신하기

- UPDATE문

- 형식 : UPDATE 테이블명 SET 속셩명=변경할_값 WHERE 조건;