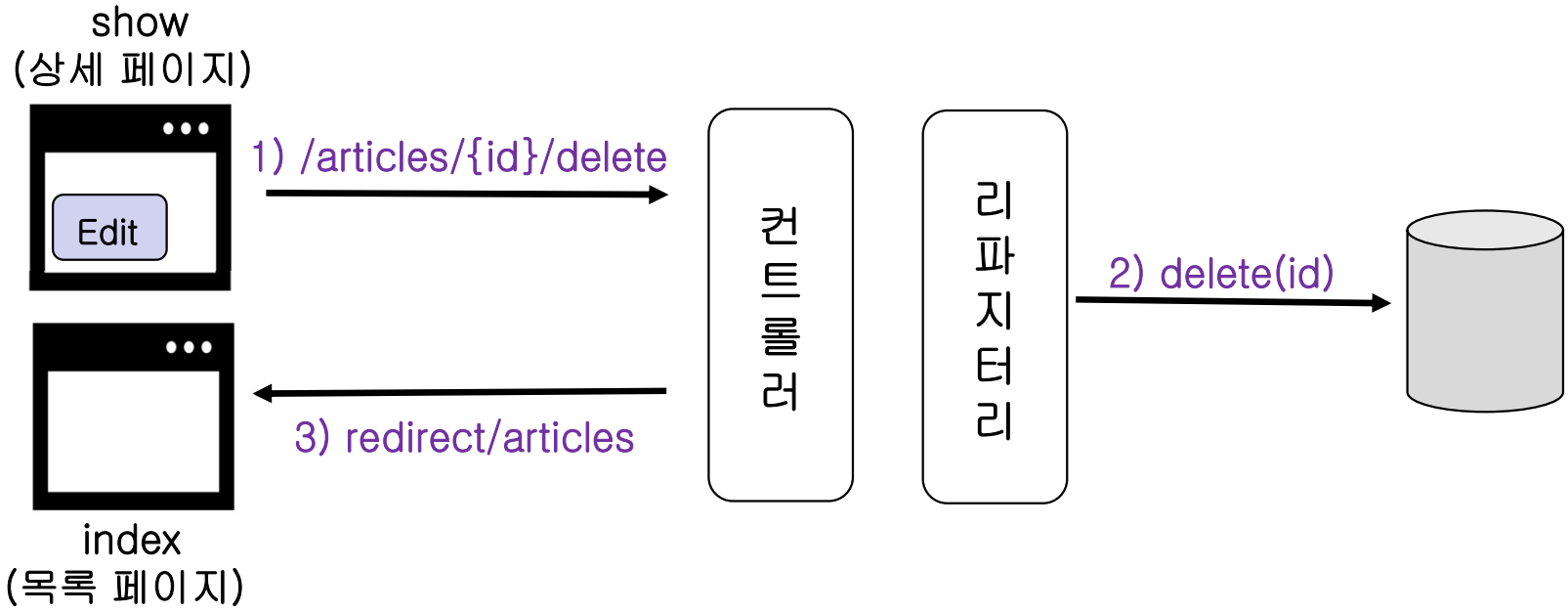
8장 게시글 삭제하기: Delete

출처: 코딩 자율학습 스프링부트3 자바 백엔드 개발 입문, 홍팍, 길벗, 2023

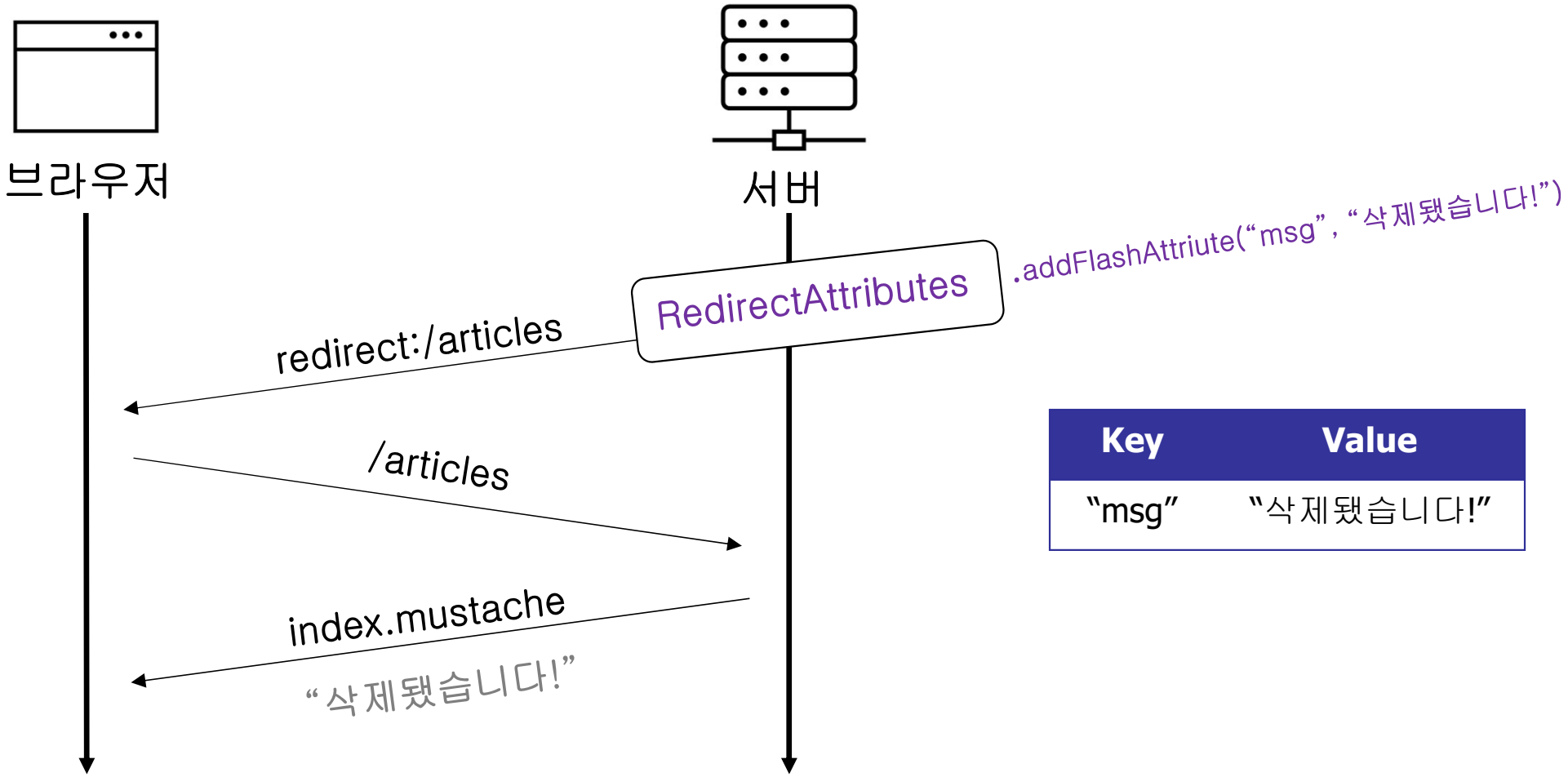
데이터 삭제 과정

- 클라이언트가 HTTP 메서드로 특정 게시글의 삭제를 요청
- 삭제 요청을 받은 컨트롤러는 리파지터리를 통해 DB에 저장된 데이터를 찾아 삭제, 기존 데이터가 있는 경우에만 수행함.
- 삭제가 완료됐다면, 클라이언트를 결과 페이지로 리다이렉트



리다이렉트된 페이지에 삭제 메시지 띄우기

- **RedirectAttributes의 addFlashAttribute() 사용**
  - 리다이렉트된 페이지에서 일회성으로 사용할 데이터 등록해서 보내기



Key	Value
"msg"	"삭제됐습니다!"

Delete 버튼 추가하기

- 상세 페이지, show.mustache 파일 수정

```
<a href="/articles/{{article.id}}/edit" class="btn btn-primary">Edit</a>
<a href="/articles/{{article.id}}/delete" class="btn btn-danger">Delete</a>
<a href="/articles">Go to Article List</a>
```

- 프로젝트 빌드
- localhost:8080/articles/3
  - Delete 버튼 생성 확인

Delete 요청을 받아 데이터 삭제하기

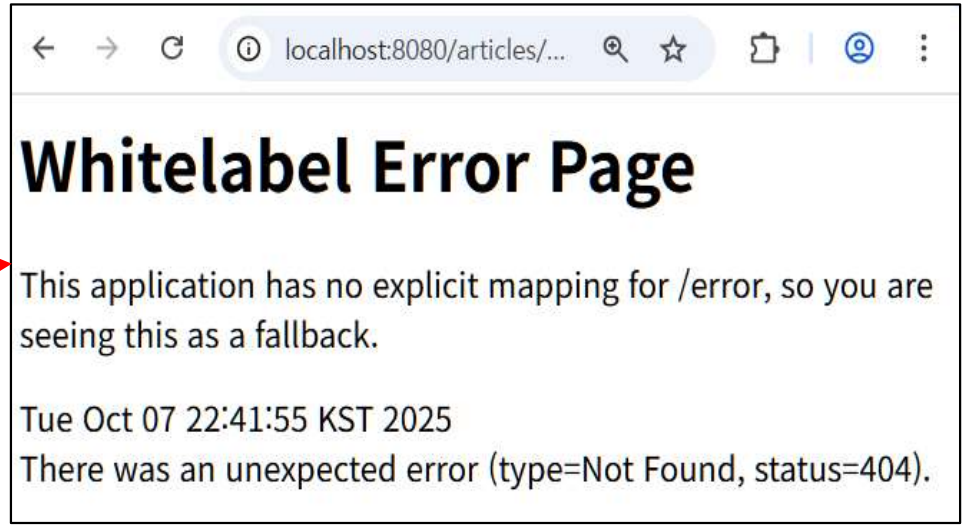
- **delete() 메서드 기본 틀 만들기**
  - ArticleController.java에 만들기
  - 삭제 요청 URL: “/articles/{id}/delete”
    - ✓ Id 1번 글 삭제 요청 URL: “/articles/1/delete”
    - ✓ Id 2번 글 ” : “/articles/2/delete”
  - 데이터를 삭제할 때는 HTTP DELETE를 사용해야 하나, HTML의 〈form〉에서 DELETE를 제공하지 않으므로 Get 방식 사용
  - delete() 메서드가 잘 동작하는지 확인하기 위해 로그 추가

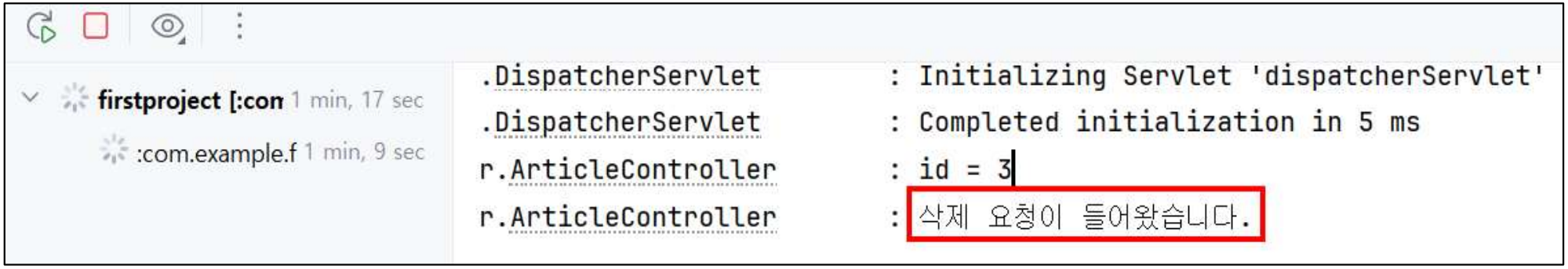
ArticleController.java

```
@GetMapping("/articles/{id}/delete") URL 요청 접수
public String delete(){
    log.info("삭제 요청이 들어왔습니다.");
    return null;
}
```

Delete 요청을 받아 데이터 삭제하기

- 서버 재시작
- localhost:8080/articles/3 접속, [Delete] 클릭
- 로그 확인





Delete 요청을 받아 데이터 삭제하기

- **DB에서 데이터 삭제 후, 결과 페이지로 리다이렉트 하는 과정**
  - 삭제할 대상 가져오기
  - 대상 엔티티 삭제하기
  - 결과 페이지로 리다이렉트 하기

ArticleController.java

```
@GetMapping("/articles/{id}/delete")
public String delete(){
    log.info("삭제 요청이 들어왔습니다.");
    // 1. 삭제할 대상 가져오기
    // 2. 대상 엔티티 삭제하기
    // 3. 결과 페이지로 리다이렉트하기
    return null;
}
```

Delete 요청을 받아 데이터 삭제하기

- **삭제할 대상 가져오기**
  - DB에 해당 id를 가진 데이터가 있는지 찾아 가져오기(없으면, null 리턴)
  - @GetMapping("/articles/{id}/delete")의 URL 주소에서 id를 delete()의 매개변수로 가져옴
  - target에 데이터가 있는지 확인하는 로그 찍기

ArticleController.java

```
@GetMapping("/articles/{id}/delete")
public String delete(@PathVariable Long id){  id를 매개변수로 가져오기
    log.info("삭제 요청이 들어왔습니다.");
    // 1. 삭제할 대상 가져오기
    Article target = articleRepository.findById(id).orElse( other: null);  데이터 찾기
    log.info(target.toString());  target에 데이터가 있는지 로그 찍어 확인
    // 2. 대상 엔티티 삭제하기
    // 3. 결과 페이지로 리다이렉트하기
    return null;
}
```

Delete 요청을 받아 데이터 삭제하기

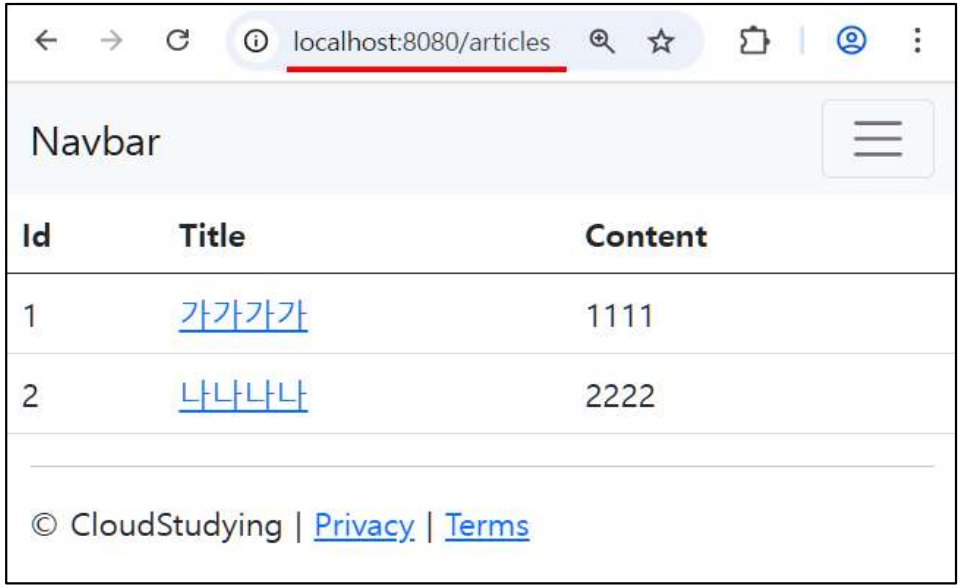
- **대상 엔티티 삭제하기**
  - CrudRepository〈T, ID〉가 제공하는 delete() 메서드 사용
    - ✓ void delete(T entity);
      - 특정 엔티티 삭제
  - if문으로 target이 null이 아닌 경우에 삭제

ArticleController.java

```
@GetMapping("/articles/{id}/delete")
public String delete(@PathVariable Long id){
    log.info("삭제 요청이 들어왔습니다.");
    // 1. 삭제할 대상 가져오기
    Article target = articleRepository.findById(id).orElse( other: null);
    log.info(target.toString());
    // 2. 대상 엔티티 삭제하기
    if(target != null) {   삭제할 대상이 있는지 확인
        articleRepository.delete(target); delete() 메서드로 대상 삭제
    }
    // 3. 결과 페이지로 리다이렉트하기
    return null;
}
```

Delete 요청을 받아 데이터 삭제하기

- 서버 재시작
- localhost:8080/articles/3 접속, [Delete] 클릭
- 로그 확인
- localhost:8080/articles 접속
  - id 3번 글 삭제 확인




```
Completed initialization in 2 ms
id = 3
삭제 요청이 들어왔습니다.
Article(id=3, title=다다다다, content=3333)
```

결과 페이지로 리다이렉트

- **게시글을 삭제하면 목록 페이지로 돌아가기**
  - ArticleController.java delete()의 return문 작성

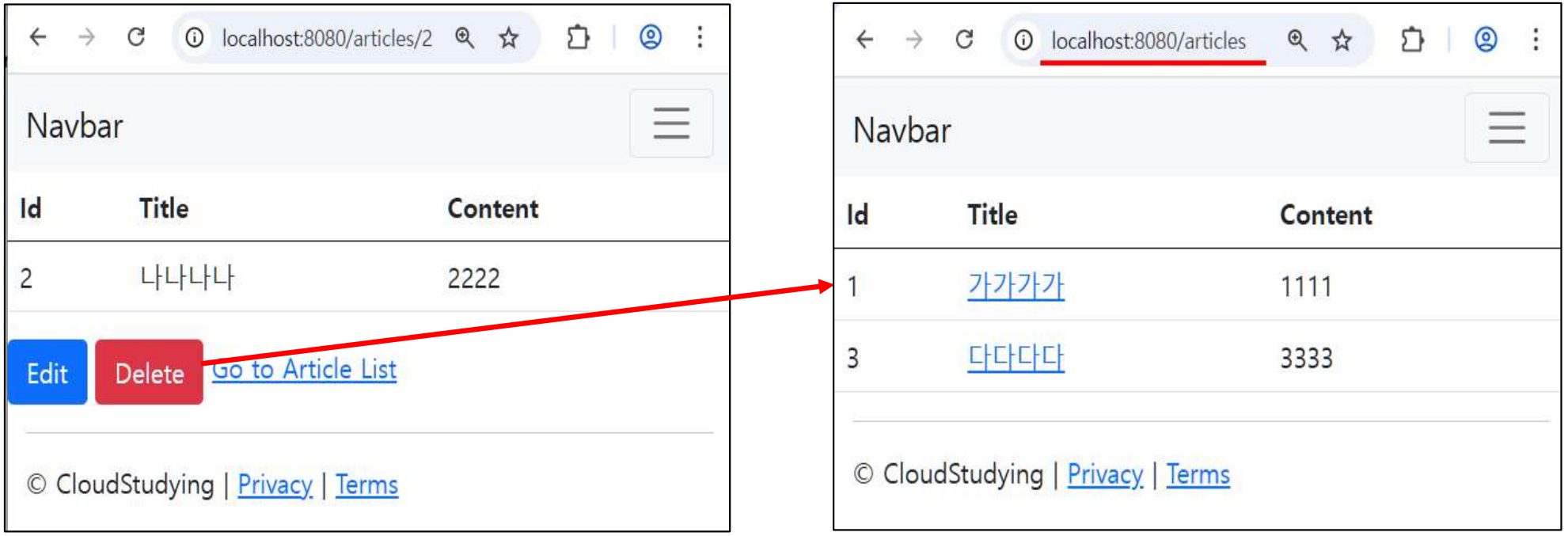
ArticleController.java

```
@GetMapping("/articles/{id}/delete")
public String delete(@PathVariable Long id){
        (중략)

    // 3. 결과 페이지로 리다이렉트하기
    return "redirect:/articles";
}
```

결과 페이지로 리다이렉트

- **서버 재시작**
- **localhost:8080/articles/2 접속**
  - [Delete] 버튼 클릭 → 목록 페이지로 이동 확인



삭제 완료 메시지 남기기

- RedirectAttributes 객체의 addFlashAttributes()를 사용하여 리다이렉트 페이지에서 사용할 1회성(휘발성) 데이터를 등록
  - 형식: `객체명.addFlashAttributes(넘겨_주려는_키_문자열, 넘겨_주려는_값_객체);`
- RedirectAttributes 객체를 활용하기 위해 delete() 메서드의 매개변수로 받기

```
@GetMapping("/articles/{id}/delete")
public String delete(@PathVariable Long id, RedirectAttributes rttr){
        (중략)
    // 2. 대상 엔티티 삭제하기
    if(target != null) {
        articleRepository.delete(target);
        rttr.addFlashAttribute( attributeName: "msg", attributeValue: "삭제됐습니다!");
    }
    // 3. 결과 페이지로 리다이렉트하기
    return "redirect:/articles";
}
```

삭제 완료 메시지 남기기

- **msg 키에 담긴 메시지 사용**
  - URL이 “/articles " 일 때, 해당 뷰 파일은 index.mustache
- **msg 키에 담긴 메시지를 출력**
  - index.html의 헤더 부분에 삭제 메시지 출력
  - [resources - templates - layouts] 디렉토리의 header.mustache 열기
  - header.mustache 편집
    - ✓ Alert 창으로 msg 내용 출력, 닫기(X) 버튼 추가

header.mustache

```
    (중략)
</nav>
{{#msg}} msg 사용 범위 설정
<div class="alert alert-primary alert-dismissible"> 메시지 창 작성
    {{msg}}
    <button type="button" class="btn-close" data-bs-dismiss="alert"
            aria-label="Close"></button>
</div>
{{/msg}}
```

삭제 완료 메시지 남기기

- 서버 재시작
- localhost:8080/articles 접속
  - 게시글 하나 선택, 상세 페이지에서 [Delete] 클릭
- 목록 페이지 상단에 “삭제됐습니다!” 메시지 창 확인, [X] 버튼 클릭