페이지 이동을 위해서 링크 사용 HTML 의 <a> <form> 태그로 작성

 링크 걸 대상

 Back

리다이렉트 정의하기 -> return "redirect:URL 주소";

return "redirect:/articles/" + saved.getId( );

어노테이션

@PostMapping("/articles/create") ("/articles/update")

@GetMapping("/articles") ("/articles/{id}") ("/articles/{id}/edit")

@Id

@GeneratedValue

@Column -> 컬럼

@NoArgsConstructor -> 기본 생성자 생성

@AllArgsConstructor -> 매개변수로 받는 생성자 생성

@Getter -> getId( ) 삭제 가능

@Entity

(@PathVariable Long id)

@RestController -> RestApi 사용 가능

@Controller

@Autowired -> private ArticleRepository articleRepository;

(@RequestBody ArticleForm dto) -> dto 있으면 RequestBody

@PatchMapping("api/articles/{id}")

@DeleteMapping("api/articles/{id}")

Public Article update(@PathVariable Long id, @RequestBody ArticleForm dto) {

POST: 데이터 생성 요청 /articles

GET: 데이터 조회 요청 /articles, /articles/id

PATCH(PUT): 데이터 수정 요청 /articles/id

DELETE: 데이터 삭제 요청 /articles/id

<form class="container" action="/articles/update" method="post">

Log.info(form.toString( )); -> lombook 필요

Article articleEntity = form.toEntity( );

200~ 정상요청 HttpStatus.OK

400~ 잘못된 요청 HttpStatus.BAD_REQUEST

403 접근 권한 없음 HttpStatus.FORBIDDEN

404 리소스 없음 HttpStatus.NOT_FOUND

500~ 서버 내부 오류 HttpStatus.INTERNAL_SERVER_ERROR

return ResponseEntity.status(HttpStatus.OK).body(null); <- 정상 요청 200

return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null); <- 잘못된 요청 400

```
//------------------------------------------------------------
// 6 & 7주차
//------------------------------------------------------------

@Slf4j
@Controller
public class ArticleController {

    @GetMapping("/articles/new") //URL(localhost:8080/articles/new) 요청 접수
    public String newArticleForm(){
        return "articles/new"; //반환값으로 뷰 페이지(articles/new.mustache)의 이름
    }

    @GetMapping("/articles/{id}")
    public String show(@PathVariable Long id, Model model) {
        log.info("id = " + id);
        //1. id를 조회해 데이터 가져오기
        Article articleEntity = articleRepository.findById(id).orElse(null);
        //2. 모델에 데이터 등록하기
        model.addAttribute("article", articleEntity);
        //3. 뷰 페이지 반환하기
        return "articles/show";
    }
```

```
    @GetMapping("/articles")
    public String index(Model model) {
        //1. 모든 데이터 가져오기
        ArrayList<Article> articleEntityList = articleRepository.findAll(); //리스트를 통해 모든 데이터 가져오기
        //2. 모델에 데이터 등록하기
        model.addAttribute("articleList", articleEntityList); //리스트 내용 모델에 등록
        //3. 뷰 페이지 설정하기
        return"articles/index";
    }



    @Autowired
    private ArticleRepository articleRepository; //articleRepository 객체 생성하기

    @PostMapping( "/articles/create" ) //PostMapping으로 URL요청 접수
    public String createArticle(ArticleForm form){ //ArticleForm == DTO임
        log.info(form.toString());
        //System.out.println(form.toString());
        //1. DTO를 엔티티로 변환
        Article article = form.toEntity();
        log.info(article.toString());
        //System.out.println(article.toString());
```

```
//2. 리파지토리로 엔티티를 DB에 저장
        Article saved = articleRepository.save(article);
        log.info(saved.toString());
        return "redirect:/articles/" + saved.getId();
        //System.out.println(saved.toString());
    }

    @GetMapping("/articles/{id}/edit")
    public String edit(@PathVariable Long id, Model model) {
        Article articleEntity = articleRepository.findById(id).orElse(null);

        model.addAttribute("article", articleEntity);

        return "articles/edit";
    }

    @PostMapping("/articles/update")
    public String update(ArticleForm form) {
        log.info(form.toString());
        //1. DTO를 엔티티로 변환
        Article articleEntity = form.toEntity();
        log.info(articleEntity.toString());
        //2. 리파지토리로 엔티티를 DB에 저장
        Article target = articleRepository.findById(articleEntity.getId()).orElse(null);
```

```
//2.2 기존 데이터값 갱신하기
            if(target != null) {
                  articleRepository.save(articleEntity);
            }
            return "redirect:/articles/" + articleEntity.getId();
      }
}
//-----------------------------------------------------------------
// 11주차
//-----------------------------------------------------------------
@Slf4j
@RestController
public class ArticleApiController {
      @Autowired
      private ArticleRepository articleRepository;

      // GET
      @GetMapping("/api/articles")
      public List<Article> index() {
            return articleRepository.findAll();
      }
```

```
    @GetMapping("/api/articles/{id}")
    public Article show(@PathVariable Long id) {
        return articleRepository.findById(id).orElse(null);
    }
    // POST
    @PostMapping("/api/articles")
    public Article create(@RequestBody ArticleForm dto) {
        Article article = dto.toEntity();
        return articleRepository.save(article);
    }
    // PATCH
    @PatchMapping("/api/articles/{id}")
    public ResponseEntity<Article> update(@PathVariable Long id, @RequestBody
ArticleForm dto) {
        // 1. DTO를 엔티티로 변환하기
        Article article = dto.toEntity();
        log.info("id: {}, article: {}", id, article.toString());
        // 2. 타깃 조회하기
        Article target = articleRepository.findById(id).orElse(null);
        // 3. 잘 못된 요청 처리하기
        if(target == null || id != article.getId()) {
            // 400, 잘 못된 요청 응답
            log.info("잘 못된 요청! id: {}, article: {}", id, article.toString());
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
        }
```

```
        // 4. 업데이트 및 정상 응답(200) 하기
        target.patch(article);
        Article updated = articleRepository.save(article);
        return ResponseEntity.status(HttpStatus.OK).body(updated);
    }
    // DELETE
    @DeleteMapping("/api/articles/{id}")
    public ResponseEntity<Article> delete(@PathVariable Long id){

        // 1. 대상 찾기
        Article target = articleRepository.findById(id).orElse(null);

        // 2. 잘못된 요청 처리하기
        if(target == null){
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
        }

        // 3. 대상 삭제하기
        articleRepository.delete(target);
        return ResponseEntity.status(HttpStatus.OK).build();
    }
}
```