3️⃣ 게시판 만들고 새 글 작성하기: Create

3.1 폼 데이터란

폼 데이터(form data) : HTML 요소인 <form> 태그에 실려 전송되는 데이터

- <form> 태그는 웹 브라우저에서 서버로 데이터를 전송할 때 사용
- **DTO**(Data Transfer Object) : <form> 태그에 실어 보낸 데이터를 서버의 컨트롤러가 담는 객체
  - DTO로 받은 데이터는 최종적으로 데이터베이스에 저장됨

3.2 폼 데이터를 DTO로 받기

1) 입력 폼 만들기

- src > mian > resources > templates 디렉터리에 articles 디렉터리 생성 → new.mustache 파일을 생성해 뷰 페이지 생성

2) 컨트롤러 만들기

- ArticleController 생성

3) 폼 데이터 전송하기

- <form> 태그에 2가지 정보(데이터를 어디로, 어떻게 보낼지에 관한 정보)를 주어야 함
- action 속성 ⇒ 어디로 보낼지 설정
  - URL 연결 주소를 적음
  - ex) action="/articles/create" ⇒ localhost:8080/articles/create 페이지로 폼 데이터를 보냄
- method 속성 ⇒ 어떻게 보낼지 설정
  - 속성 값으로 get과 post 설정 가능
  - ex) method="post"

4) 폼 데이터 받기

설정한 action, method 속성 정보를 조합해 서버의 컨트롤러가 사용자가 전송한 폼 데이터를 받도록 하기

- 컨트롤러에 createArticle() 메서드를 추가하고 return 값에는 빈문자를 적음
- URL 요청을 받아 오기 위해 `@PostMapping()` 사용(뷰 페이지에서 폼 데이터를 post 방식으로 전송했으므로)
  - 괄호 안에 받는 URL 주소를 넣음
  - ex) @PostMapping("/articles/create")

5) DTO 만들기

- com.example.firstproject에 새 패키지를 생성
- 폼 데이터를 받아올 DTO인 ArticleForm 클래스를 생성
- 제목과 내용을 받아올 title, content 필드 선언
- 전송받은 제목과 내용을 필드에 저장하는 생성자 추가
- 데이터를 잘 받았는지 확인할 toString() 메서드 추가

6) 폼 데이터를 DTO에 담기

- ArticleController에서 코드 수정
- 폼에서 전송한 데이터를 createArticle() 메서드의 매개변수로 받아옴
- 폼에서 전송한 데이터가 DTO에 잘 담겼는지 확인하기 위해 출력문 추가

7) 입력 폼과 DTO 필드 연결하기

- title, content 필드에 값이 들어가게 하기 위해 new.mustache 입력 폼에 필드명 지정 ⇒ 해당 입력 폼이 DTO의 필드와 연결됨
- new.mustache에 속성 추가
  - 제목 입력하는 <input> 태그에 name=”title” 속성 추가
  - 내용 입력하는 <textarea> 태그에 name=”content” 속성 추가

3.3 DTO를 데이터베이스에 저장하기

1) 데이터베이스와 JPA

- 데이터베이스(DB) : 데이터를 관리하는 창고
  - 모든 데이터는 행과 열로 구성된 테이블에 저장해 관리
- JPA(Java Persistence API) : 자바 언어로 DB에 명령을 내리는 도구
  - 핵심 도구로 엔티티(entity)와 리파지터리(repository)가 있음
    - 엔티티 : 자바 객체를 DB가 이해할 수 있게 만든 것
      - 이를 기반으로 테이블이 만들어짐
    - 리파지터리 : 엔티티가 DB 속 테이블에 저장 및 관리될 수 있게 하는 인터페이스
    - DTO로 받은 폼 데이터를 DB로 저장하는 순서
      1. DTO를 엔티티로 변환하기
      2. 리파지터리를 이용해 엔티티를 DB에 저장하기

2) DTO를 엔티티로 변환하기

form 객체의 toEntity() 메서드를 호출, 반환 값을 Article 타입의 article 엔티티에 저장

- Article 클래스 만들기 : 기본 패키지 아래 entity 패키지에 클래스 생성
  - `@Entity` 어노테이션 : JPA에서 제공, 어노테이션이 붙은 클래스를 기반으로 DB에 테이블이 생성됨
  - title, content 필드 선언

        - `@Column` 어노테이션 : 필드가 DB 테이블의 각 열(column)과 연결됨
    - 엔티티의 대푯값 넣기
        - 대푯값을 id로 선언
        - `@Id` 어노테이션 : 엔티티의 대푯값 지정
        - `@GeneratedValue` 어노테이션 : 자동 생성 기능 추가(숫자가 자동으로 매겨짐)
    - Article 생성자 추가 및 toString() 메서드 추가
- toEntity() 메서드(DTO인 form 객체를 엔티티 객체로 변환하는 역할을 수행) 추가하기
    - ArticleForm(DTO 클래스)에 toEntity() 메서드 추가
    - toEntity() 메서드에서는 폼 데이터를 담은 DTO 객체를 엔티티로 반환
        - Article 클래스의 생성자 형식에 맞게 전달값 작성
        - ex) return new Article(null, title, content);

3) 리파지터리로 엔티티를 DB에 저장하기

- articleRepository.save() 메서드를 호출해 article 엔티티를 저장
    - save() 메서드 : 저장된 엔티티를 반환
- 필드 선언부에 ArticleRepository 타입의 articleRepository 객체를 선언
- 리파지터리(ArticleRepository) 만들기
    - 기본 패키지 아래 repository 패키지 생성
    - ArticleRepository 인터페이스 생성
    - CrudRepository 인터페이스 상속받기
        - extends Crud 입력해 CrudRepository<T, ID> 선택
        - CrudRepository를 상속하면 엔티티를 관리(생성, 조회, 수정, 삭제)가 가능
    - CrudRepository의 <>에 Article, Long 요소 받기
        - Article : 관리 대상 엔티티의 클래스 타입
        - Long : 관리 대상 엔티티의 대푯값 타입
- **객체 주입하기**
    - `@AutoWired` 어노테이션 : 스프링 부트에서 제공, 컨트롤러의 필드에 붙이면 스프링 부트가 만들어 놓은 객체를 가져와 주입
        - 의존성 주입(DI, Dependency Injection)
- 데이터 저장 확인하기
    - DTO가 엔티티로 잘 변환되는지 article.toString() 메서드를 호출해 확인
    - article이 DB에 잘 저장되는지 saved.toString() 메서드를 호출해 확인

3.4 DB 데이터 조회하기

1) H2 DB 접속하기

- src > main > resources 의 application.properties 파일을 오픈
- spring.j2.console.enabled=true 작성 : H2 DB에 웹 콘솔로 접근할 수 있도록 허용하는 설정
- 서버 재시작
- 'jdbc:h2:mem: ...' 문구를 복사해 웹 브라우저의 JDBC URL 에 붙여넣어 DB에 접속
    - JDBC H2 DB가 메모리에서 동작하는 데 그 주소가 ...임을 의미

2) 데이터 조회하기

- 테이블 속성 조회
    - `SELECT 속성명 FROM 테이블명;`

  * 레코드(record) : 데이터가 테이블에 저장되는 단위인 행 하나하나
- 레코드 삽입
    - `INSERT INTO 테이블명(속성명1, 속성명2, 속성명3, ...) VALUES(값1, 값2, 값3, ...);`