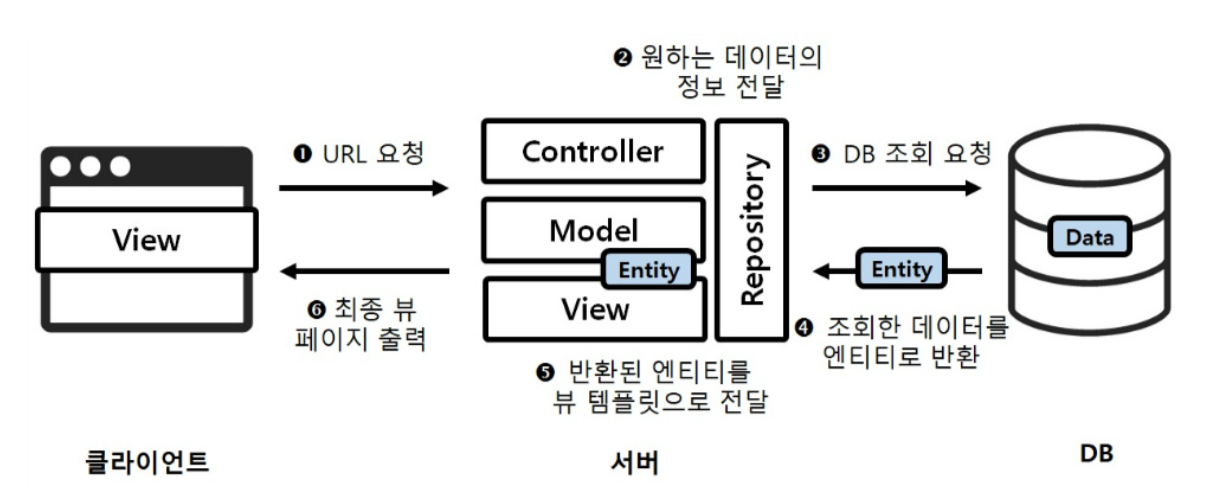
게시글 읽기 : Read

데이터 조회 과정

웹 페이지에서 게시글을 등록하게 되면 서버를 통해 DB에 저장이 된다. 이렇게 DB에 저장된 데이터를 웹페이지에 출력하는 과정은 앞서 배운 다양한 요소 간의 상호작용으로 이루어진다.

1. 사용자가 데이터를 조회해 달라고 웹 페이지에서 url 요청을 보낸다.
2. 서버의 컨트롤러가 요청을 받아 해당 url에서 찾고자 하는 데이터 정보를 repository에 전달한다.
3. Repository는 정보를 가지고 DB에 데이터 조회를 요청한다.
4. DB는 해당 데이터를 찾아 이를 엔티티로 반환한다.
5. 반환된 엔티티는 모델을 통해 뷰 템플릿으로 전달된다.
6. 최종적으로 결과 뷰 페이지가 완성되어 사용자의 화면에 출력이 된다!

데이터 조회 과정



단일 데이터 조회하기 : 로깅 기능을 통해 받은 기록을 웹페이지에서 확인할 수 있도록 하는 것이 이번 목표

url 요청 받기 : DB에 저장된 데이터를 웹 페이지에서 보기 위해선 해당 출력 페이지에 접속이 필요 -> url 요청 필요

1번 id 조회시 1, 2번 조회시 2 … 1000번 조회시 localhost:8080/articles/1000 이런식으로 url요청받기

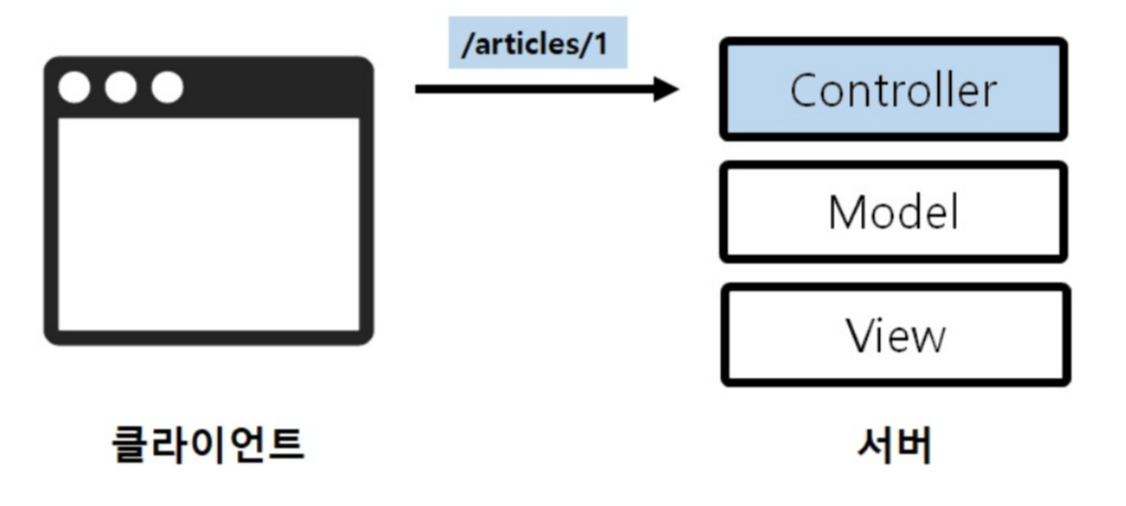
url요청 컨트롤러 제작하기

1. 조회할 데이터가 Article이기에 ArticleController에 코드를 추가하는 방식으로 만들기
2. 기존 코드 맨 밑에 @GetMapping() 어노테이션 작성 후 괄호 안에는 “/article/{id}”로 작성하기 (중괄호 id는 변수취급)
3. url요청을 받아 수행할 show() 메소드 생성 매개변수로는 url 안의 id를 가져오는데, 이때 @PathVariable 어노테이션을 붙이기 (이는 url요청으로 들어온 전달값을 컨트롤러의 매개변수로 가져오는 어노테이션을 의미)
4. 컨트롤러가 id를 잘 받았는지 확인하기 위해 log.info(“id = ” + id); 를 추가



현재 로그에 id가 잘나오는 것을 확인할 수 있다.

컨트롤러에 url 요청하기

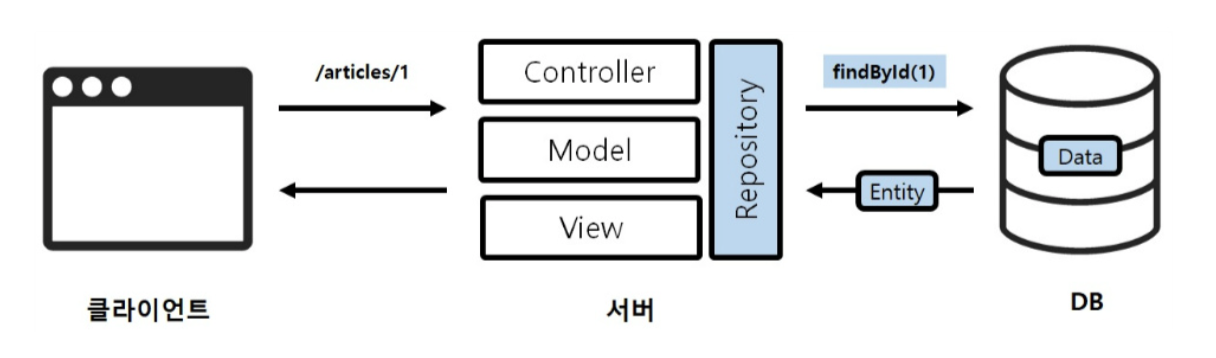


데이터를 조회하여 출력하기

1. Id를 이용하여 DB에서 해당 데이터를 가져온다
2. 가져온 데이터를 모델에 등록하기
3. 조회한 데이터를 사용자에게 보여주기 위한 뷰 페이지를 만들고 반환하기

Id를 조회하여 데이터를 가져오기 : DB에서 데이터를 가져오는 주체는 repository이다. Repository의 구현 객체는 @AutoWired 어노테이션으로 주입받는다.

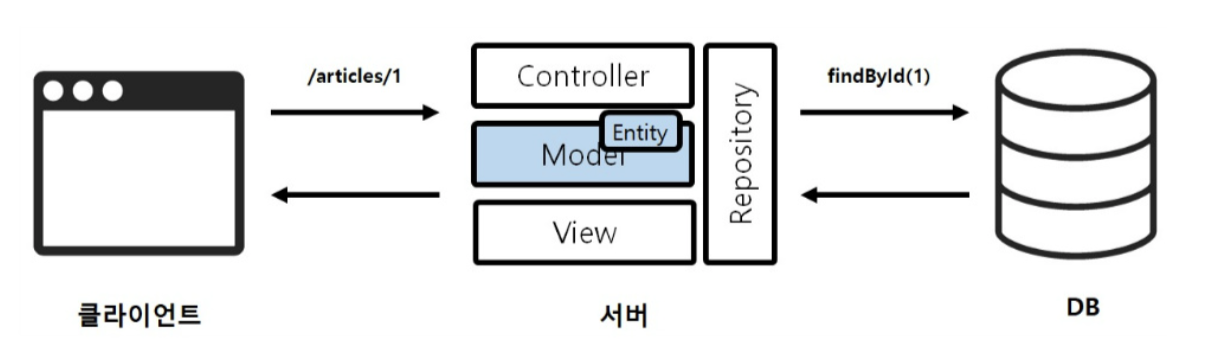
Repository로 데이터를 가져오기



1. articleRepository까지 입력하고 점을 찍으면 사용할 수 있는 메소드 목록이 나온다. 이 목록에서 findById(Long id)를 선택한다. 이는 JPA의 CrudRepository가 제공하는 메소드로 특정 엔티티의 id 값을 기준으로 데이터를 찾아 Optional 타입으로 반환합니다. 일단 반환 타입을 무시하고 찾은 데이터를 Article 타입의 articleEntity 변수에 저장한다.
2. 이때 문제가 생기는데, 찾은 값이 반환타입이 Article 이 아니라서 생기는 오류이다. 반환타입은 Optional<Article> 타입이기에 이를 변경하기

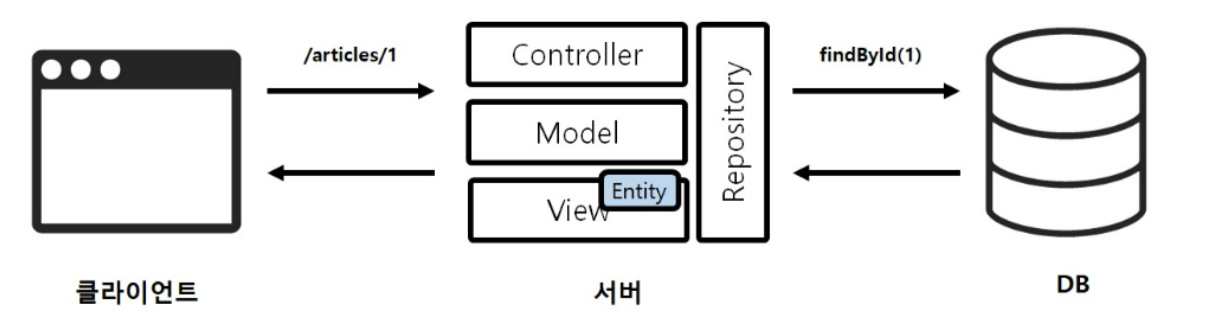
모델에 데이터 등록하기

articleEntity에 담긴 데이터를 모델에 등록해보기. 이는 MVC 패턴에 따라 조회한 데이터를 뷰 페이지에서 사용하기 위해서이다.



1. 모델을 사용하기 위해서는 show() 메소드의 매개변수로 model 객체를 받아온다.
2. 모델에 데이터를 등록할 때는 addAttribute() 메소드를 사용한다.

뷰 페이지 반환하기

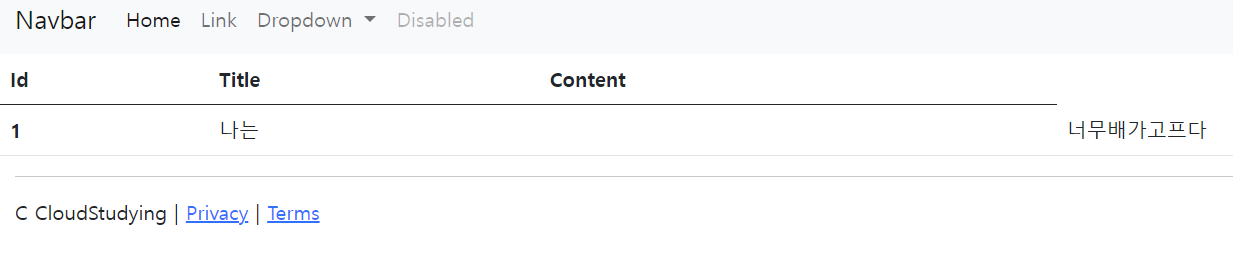


1. 뷰 페이지는 articles 라는 디렉토리 안에 show라는 파일이 있다라고 가정하고 다음과 같이 반환합니다. (return “aricles/show”)
2. show.mustache 라는 템플릿을 제작하기
3. 뷰 템플릿 기본 웹 템플릿 작성

템플릿 내에 적정 범위에 {{#article}} , {{/article}} 머스테치 문법을 사용하여 변수를 사용하도록 함

기본 생성자 추가하기

1. 기본생성자를 추가하기 기본생성자는 매개변수가 아무것도 없는 생성자를 의미한다.
2. 이를 위해 @NoArgsConstructort 어노테이션을 추가한다.



정상적으로 데이터가 웹페이지에 나타난 것을 볼 수 있다!

실습 정리

1. 사용자가 입력한 데이터를 조회하기 위해서는 먼저 url 요청을 진행해야 한다. 이번 실습에서는 /articles/id 형식으로 url요청을 진행
2. 컨트롤러는 @GetMapping(“/articles/{id}”) 으로 url 요청을 받는다. 그리고 이 url에 포함된 id를 show() 메소드의 매개변수로 받는데, 이때 매개변수 앞에 @PathVariable 어노테이션을 붙인다. 그래야 url의 id를 가져올 수 있다.
3. repository에서 DB에 저장된 데이터를 id로 조회할 때 findById() 메소드를 사용한다. 특별히 조회가 되지 않는 데이터도 처리해야하기에 orElse() 메소드를 통해 null 이 반환되도록 한다.
4. Id로 DB에서 조회한 데이터는 모델에 article이라는 이름으로 등록을 진행한다.
5. 최종적으로 모델에 등록된 데이터를 뷰 페이지에서 사용할 수 있게 설정을 진행한다. 이때 #과 /를 이용하여 열고 닫아 데이터를 사용할 수 있는 영역을 진행 (mustache 문법 사용)

1분 퀴즈

* @PathVariable 은 url요청으로 들어온 전달값을 컨트롤러의 매개변수로 가져오는 어노테이션을 의미한다.
* FindById()란 JPA의 CrudRepository가 제공하는 메소드로 특정 엔티티의 id 값을 기준으로 데이터를 찾아 Optional 타입으로 반환

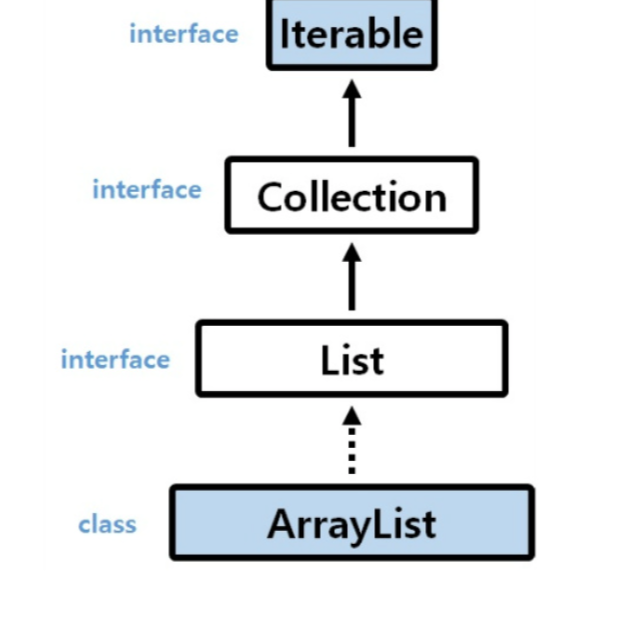
데이터를 조회하여 출력하기

1. DB에서 모든 Article 데이터를 가져오기
2. 가져온 Article묶음을 모델에 등록
3. 사용자에게 보여줄 뷰 페이지 설정하기

모든 데이터를 가져오기

* DB에서 데이터를 가져오기 위해서는 repository가 필요하다. 따라서 articleRepository를 입력후 findAll() 메소드를 사용하여 모든 데이터를 가져오도록 한다.
* 메소드 수행 결과는 ArticleEntityList라는 이름으로 받기 타입은 List<Article> 로 설정

이때, 형변환을 통해 타입을 변경하기



이후 ArticleRepository에서 findAll() 메소드를 오버라이딩을 진행한다.

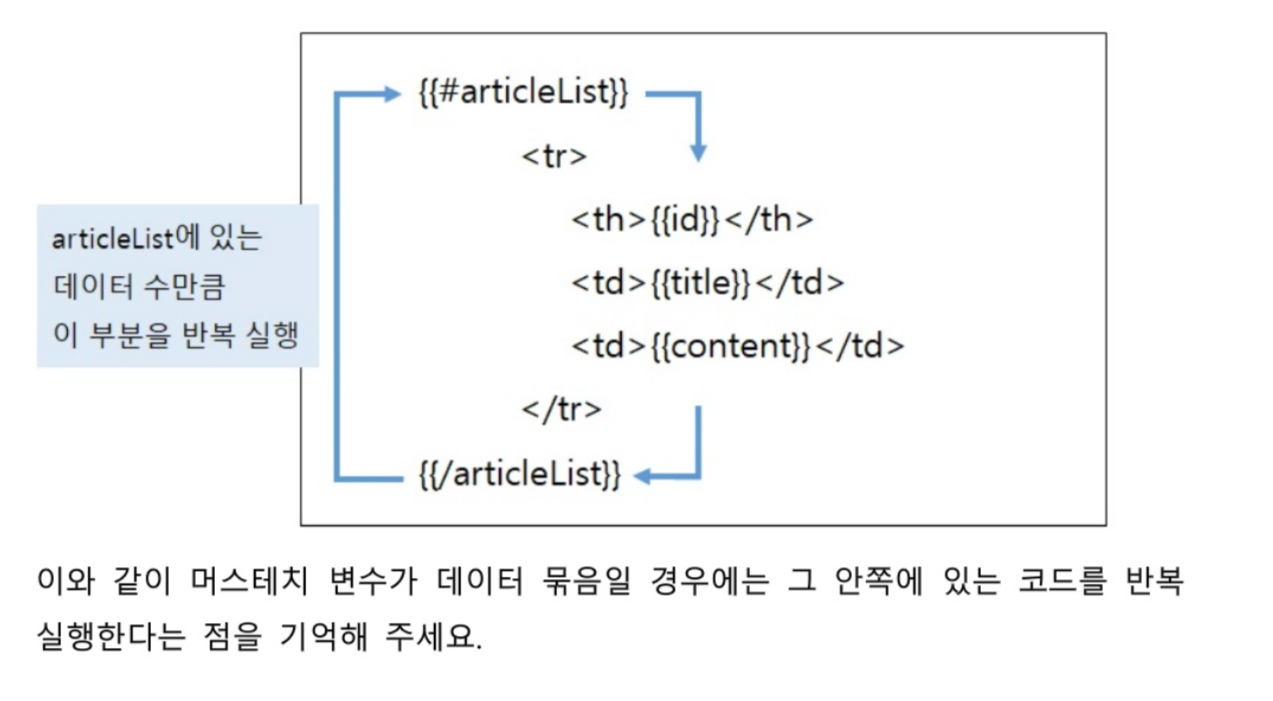
모델에 데이터를 등록하기 : 가져온 데이터를 받은 articleEntityList를 뷰 페이지로 전달할 때 모델을 사용

1. Index() 메서드의 매개변수로 model 객체를 받아온다.
2. Model.addAttribute() 메소드로 전달할 데이터 묶음인 articleEntityList를 “articleList” 라는 이름으로 등록을 진행

뷰 페이지 설정하기

1. Articles 디렉토리 안에 index.mustache 파일로 설정될 수 있도록 return 문을 작성하기
2. articles라는 templates에 index 머스테치 파일을 생성하기
3. 뷰 템플릿 안에 기본틀을 입력(header , footer)
4. 이후 {{#articleList}}, {{/articleList}} 로 설정하여 머스테치 안에 변수를 설정할 수있도록 지정

기억해야할점 : mustache 문법!



마무리

1. 데이터 조회과정 : 웹페이지에서 게시글을 등록하면 서버를 통해 DB에 저장된다. DB에 저장된 데이터는 다음과 같은 과정을 거쳐 웹 페이지에서 조회할 수 있다.

1) 사용자가 웹 페이지에서 데이터를 조회해 달라고 url 요청을 진행

2) 서버의 컨트롤러가 이 요청을 받아 해당 url에서 찾고자 하는 데이터 정보(여기서는 id)를 repository에게 전달

3) repository는 정보(id)를 가지고 DB에 데이터 조회 요청을 진행

4) DB는 해당 데이터를 찾아 이를 엔티티로 반환

5) 반환된 엔티티는 모델을 통해 뷰 템플릿으로 전달

6) 최종적으로 결과 뷰 페이지가 완성이 되어 화면에 출력