JPA： java persistence API

Spring boot JPA:只需要关注 CRUD 操作，无需关注具体代码实现，自动根据方法名生成实现。

JPA 基本步骤:

课程实例:

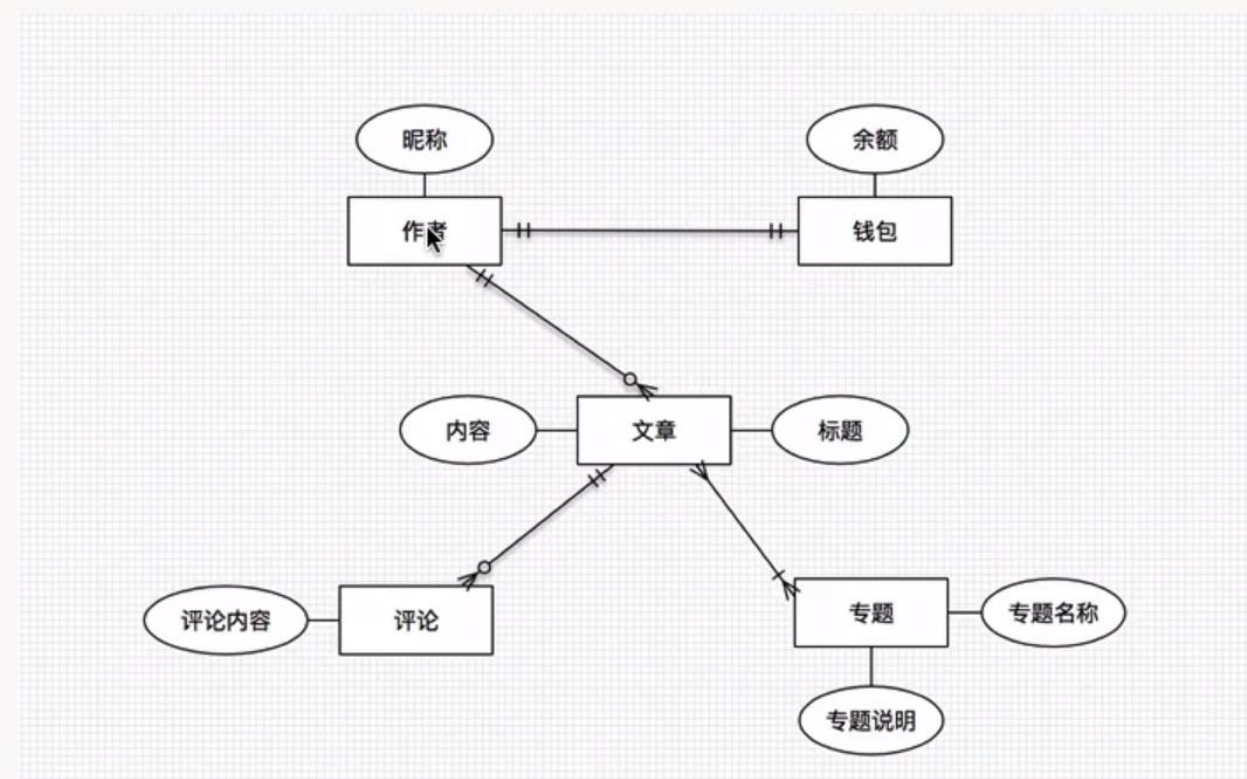
作者注册更新删除。

文章新增，更新，删除

评论文章

专题新增，更新，删除

专题收录



JPA 使用步骤:

1. Spring boot 框架构建
2. 引入 JPA 模块 （initializr 引入，或者 pom 中加入）
3. 在 properties,yml 文件里配置 JPA
4. 模型类？Domain 包下，定义实体类
5. 实现 Repository 接口

Jpa.hibernate.ddl-auto 参数说明?
Update:第一次加载 hibernate 时,会根据我们的 model 类在对应数据库里创建新表,以后再加载 hibernate 的时候，会根据 model 类来自动更新表结构。注意:即使实体模型发生了变化，表结构不会删除以前的列或者行，只会新增一个列。
还有 create，create-drop，validate 参数。
Create:第一次加载 hibernate 时，根据 model 类创建新表，每次加载 hibernate 都会创建新表。表数据会丢失。慎用此属性。
Create-drop:加载 hibernate 时，会根据 model 类创建新表，当 sessionFactory 销毁的时候，把对应的表删除掉。相当于临时表。
Validate：每次加载 hibernate 时候，根据 model 类来验证我们的表结构，不会创建新表，或者是更新表。

@Basic(fetch=LAZY)返回的数据库表对象默认不含该列属性，当调用 class.getter 时候，才会去数据库查找此属性。

@Transient 不需要持久化，短暂使用。数据库表里不会映射此属性
@Temporal(Temporal.DATE)只存日期，不存小时 秒。。
@Lob 映射成 BLOB 或者 CLOB，BLOB 是图片，CLOB 是文本
@Lob 一般与延迟加载配合使用。

实体类内必须要加空的构造函数：
JPA 对象时 hibernate 创建的，hibernate 内部通过反射机制，反射机制需要用到无参数构造函数。

Repository 接口？

5、Repository接口

Spring Data JPA简化了持久层的操作，开发者只需声明持久层接口，而不需要实现该接口。Spring Data JPA内部会根据不同的接口方法，采用不同的策略自动生成实现。

而开发者声明持久层接口，需要直接或者间接的方式继承Repository接口，从而使自定义的持久层拥有了持久层的操作能力。

Repository接口：

- **Repository** 是Spring Data的核心接口，最顶层接口，不包括任何方法，他的目的是为了统一所有Repository的类型，且让组件扫描的时候自动识别。

扩展的Repository接口：

- **CrudRepository** 继承了Repository，提供了曾删改查方法，可以直接调用。
- **PagingAndSortingRepository** 继承了CrudRepository，提供了分页和排序两个方法。
- **JpaRepository** 继承了PagingAndSortingRepository，针对于JPA技术的接口，提供了flush(), saveFlush(), deleteInBatch()等方法

方法名创建查询?
Find+全局修饰+By+实体的属性名称+限定词+连接词+。。。(其他实体属性）+OrderBy+排序属性+排序方向

Distinct 是全局修饰，非必须
Nickname 和 phone 是实体属性名
And 是连接词
Ignorecase 是限定词
Signdate 是排序属性
Desc 是排序方向
例如

```
例如：findDistinctByNickNameIgnoreCaseAndPhoneOrderBySignDateDesc(String nickname,String phone)
```

JPQL 语句自定义查询？？？

SQL 查询，JPQL 更新

使用@Query 标记 JPQL 语句

```
@Query("select a from Author a where
a.phone=?1")
List<Author> findByPhone(String phone);
```

```
 @Query("select
a.nickName,length(a.nickName) from Author a
where a.nickName like %?1%")
List<Object[]> findArry(String nickName);
```

开启 SQL 语句查询?

```
@Query(value = "select * from author where
nickname like %?1%", nativeQuery = true)
List<Author> findbySql(String nickName);
```

分页与排序和事务处理

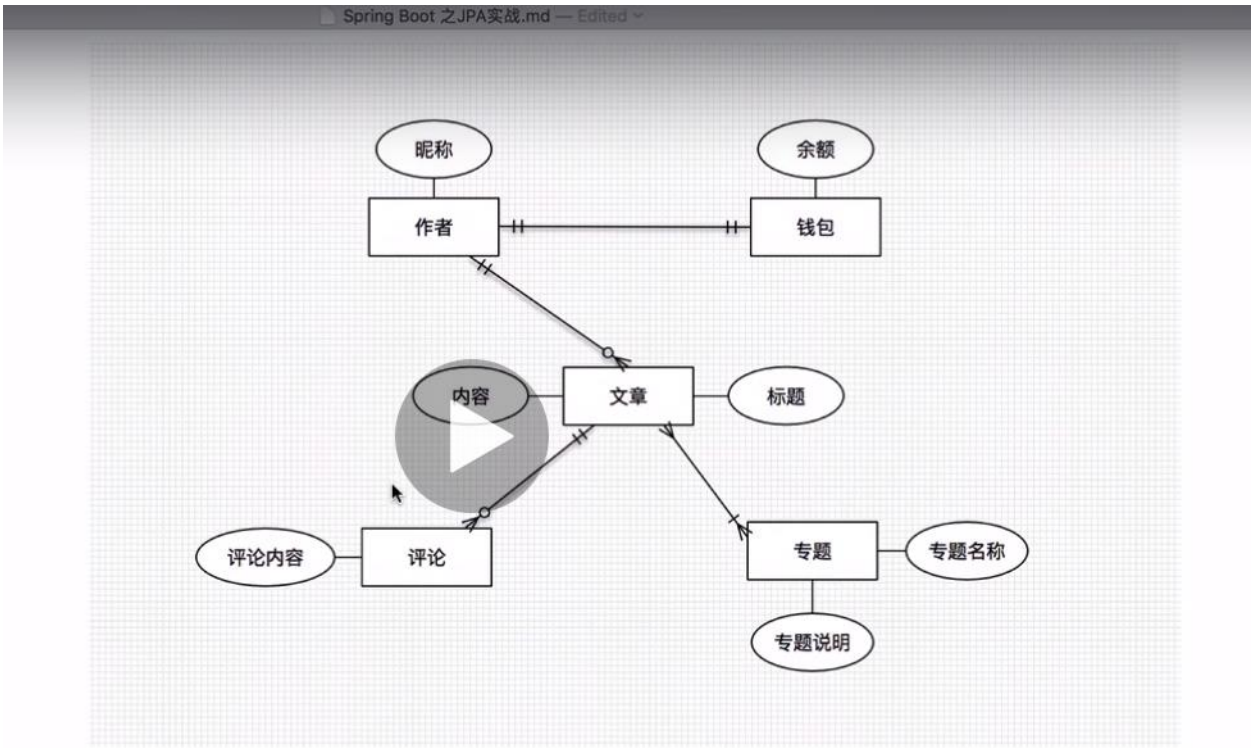
分页排序:

```
@RestController
@RequestMapping("/authors")
public class TestController {
    @Autowired
    private AuthorRepository authorRepository;
    @GetMapping
    public Object findAuthorForPage(@PageableDefault(page=0, size=5, sort = {"Id"}, direction = Sort.Direction.DESC) Pageable pageable){
        return authorRepository.findAll(pageable);
}
}
```

事务处理:

@Transactional

一对一关系？作者和钱包一对一




```
public class Author {
    @Id
    @GeneratedValue
    private Long id;
    private String nickName;
    private String phone;
    @Temporal(TemporalType.DATE)
    private Date signDate;
```

//级联保存 persist,级联更新 merge,级联查询不需要设置，只需要设置 1 对 1 关系

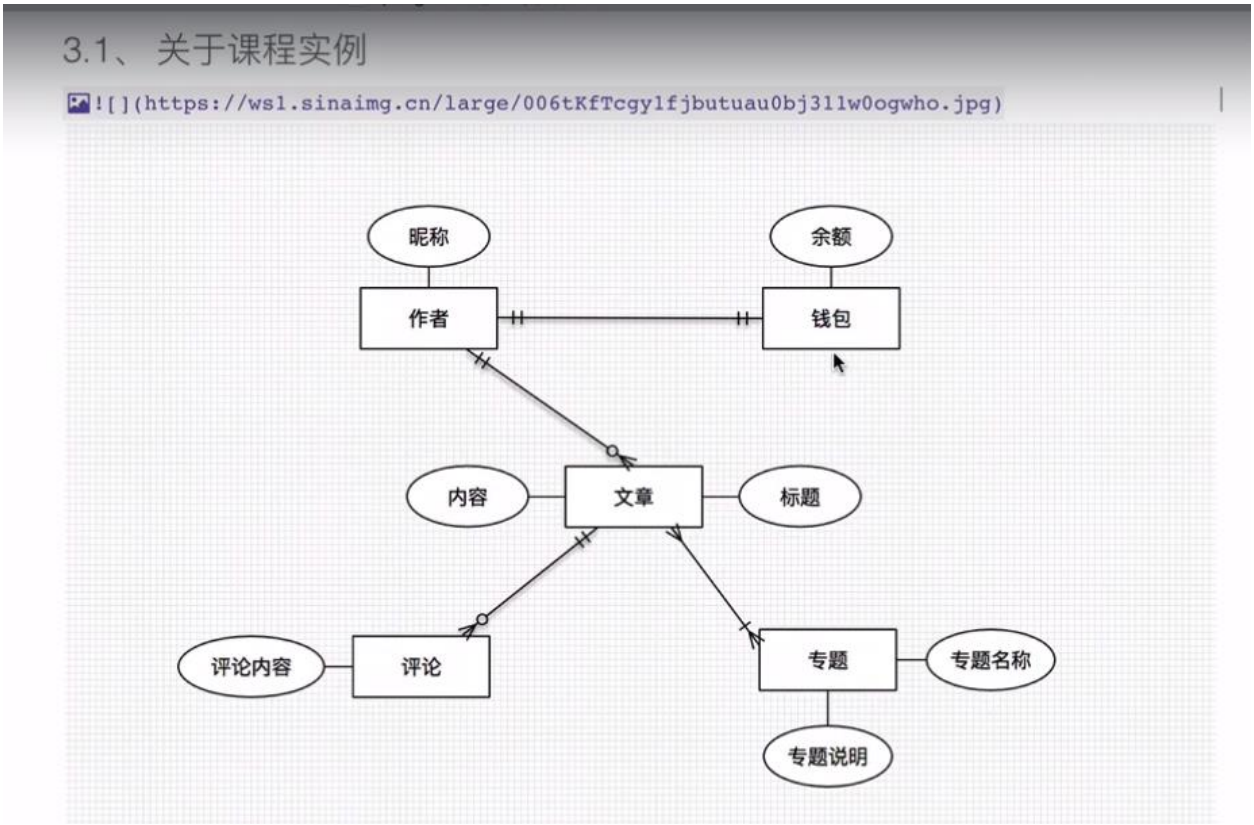
```
    @OneToOne(cascade = {CascadeType.PERSIST, CascadeType.MERGE,CascadeType.REMOVE})
    private Wallet wallet;
```

```
public Wallet getWallet() {
    return wallet;
}

public void setWallet(Wallet wallet) {
    this.wallet = wallet;
}



public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getNickName() {
    return nickName;
}

public void setNickName(String nickName) {
    this.nickName = nickName;
}

public String getPhone() {
    return phone;
}

public void setPhone(String phone) {
    this.phone = phone;
}

public Date getSignDate() {
    return signDate;
}

public void setSignDate(Date signDate) {
    this.signDate = signDate;
```

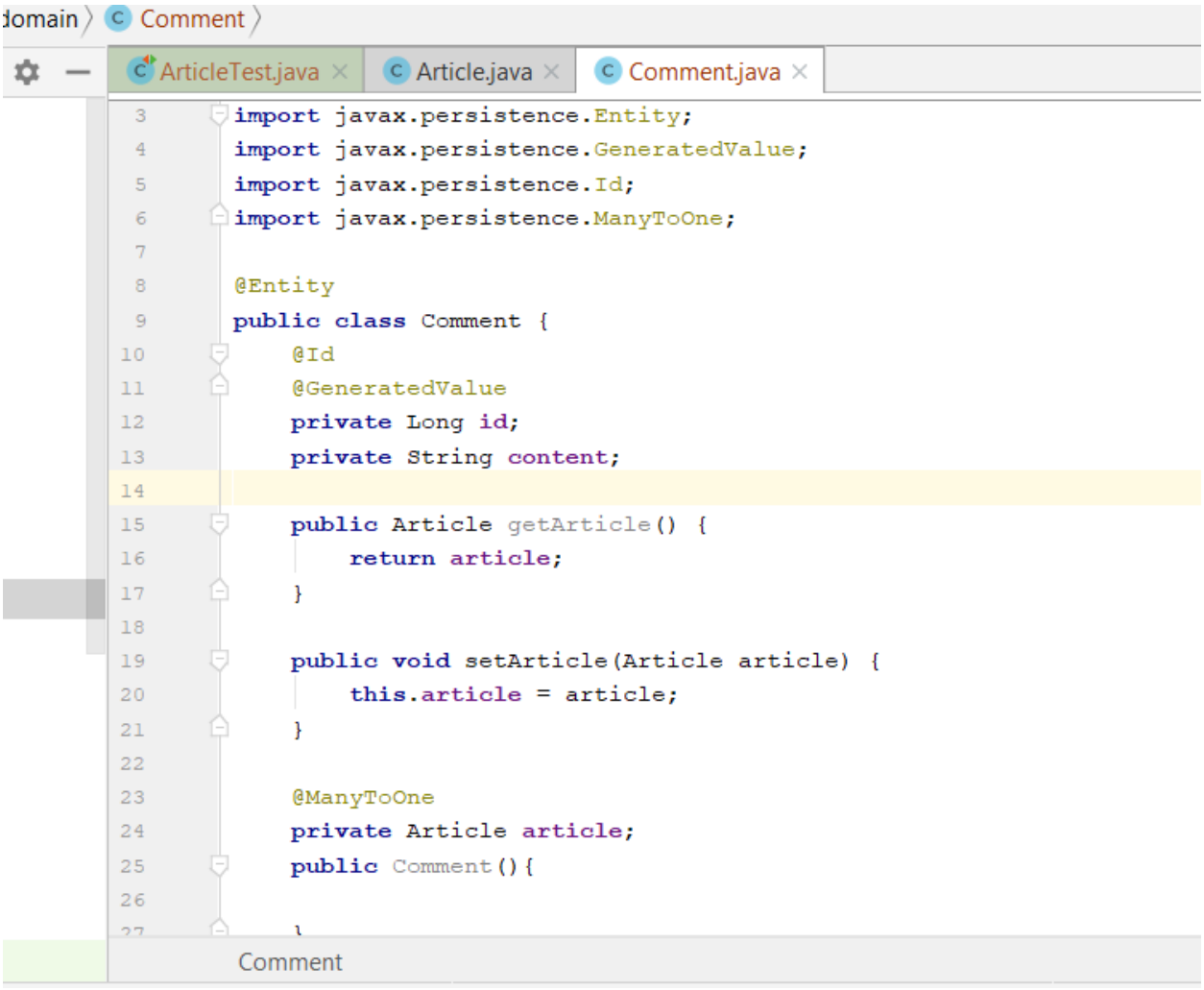
```
}


public Author(){

}
```

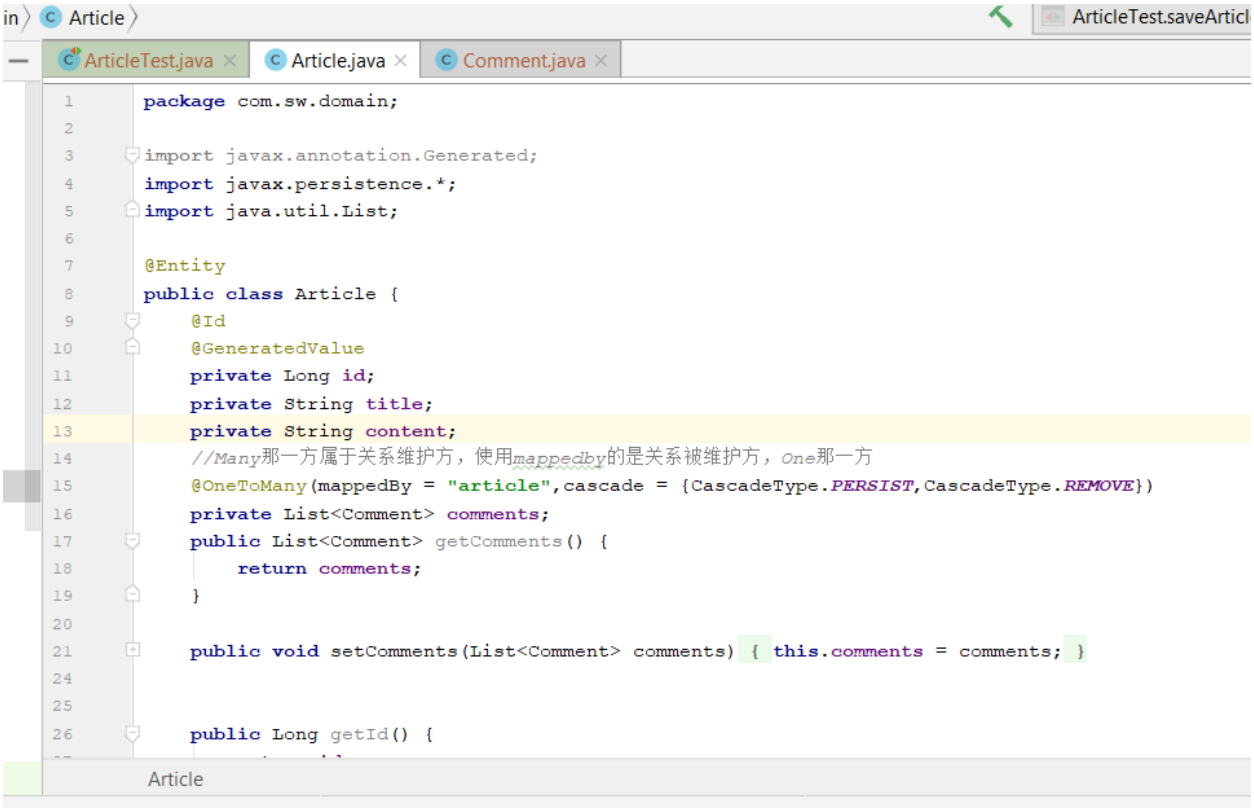
@OneToMany 注解
一对一和一对多

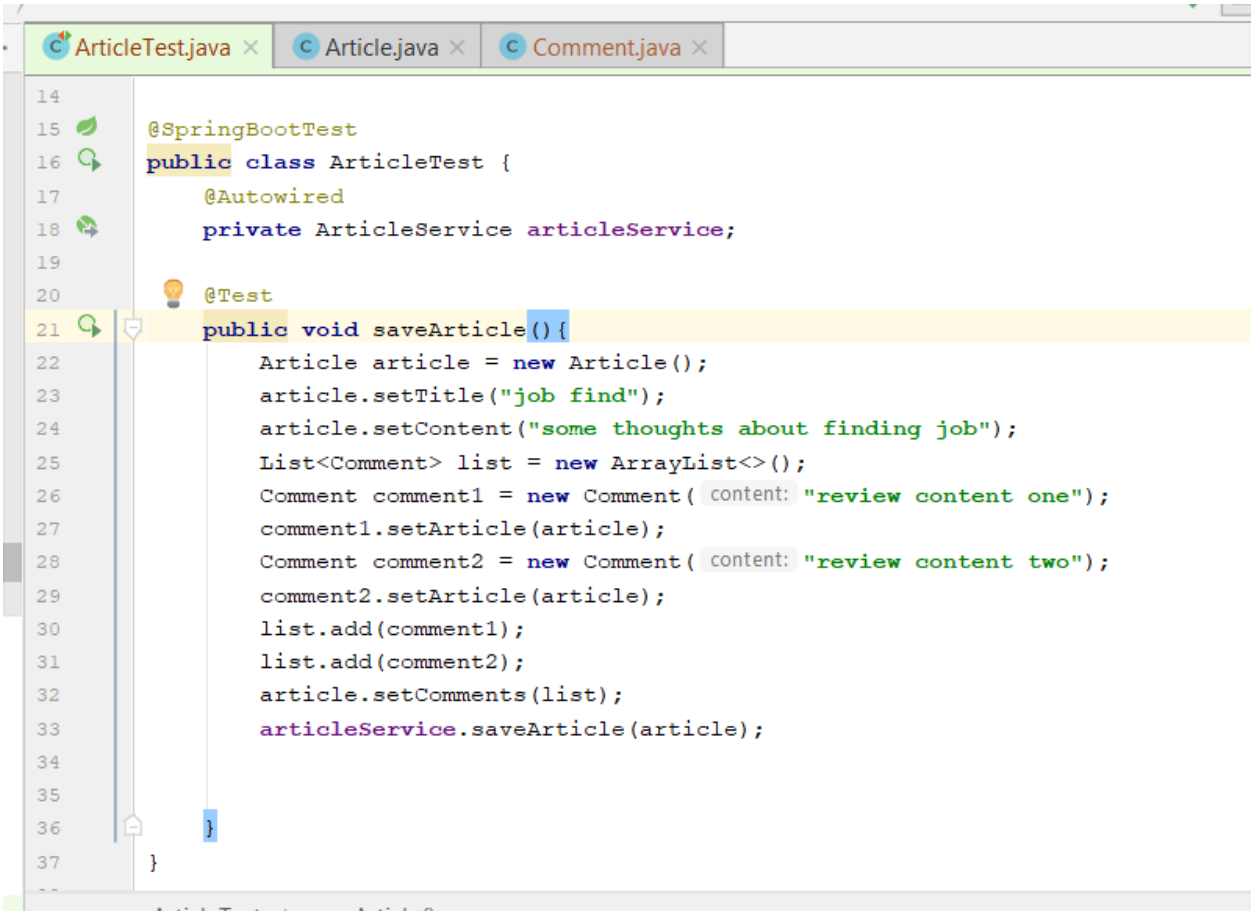
关系设置，属性设置，延迟加载？？？



文章和评论 一对多的关系
API？？
文章新增 更新 删除
评论文章，新增 删除。


domain
Comment
ArticleTest.java
Article.java
Comment.java
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.Id;
6 import javax.persistence.ManyToOne;
7
8 @Entity
9 public class Comment {
10 @Id
11 @GeneratedValue
12 private Long id;
13 private String content;
14
15 public Article getArticle() {
16 return article;
17 }
18
19 public void setArticle(Article article) {
20 this.article = article;
21 }
22
23 @ManyToOne
24 private Article article;
25 public Comment(){
26
Comment


in
Article
ArticleTest.saveArticl
ArticleTest.java
Article.java
Comment.java
1 package com.sw.domain;
2
3 import javax.annotation.Generated;
4 import javax.persistence.*;
5 import java.util.List;
6
7 @Entity
8 public class Article {
9 @Id
10 @GeneratedValue
11 private Long id;
12 private String title;
13 private String content;
14 //Many那一方属于关系维护方，使用mappedby的是关系被维护方，One那一方
15 @OneToMany(mappedBy = "article",cascade = {CascadeType.PERSIST,CascadeType.REMOVE})
16 private List<Comment> comments;
17 public List<Comment> getComments() {
18 return comments;
19 }
20
21 public void setComments(List<Comment> comments) { this.comments = comments; }
24
25
26 public Long getId() {
Article


ArticleTest.java
Article.java
Comment.java
@SpringBootTest
public class ArticleTest {
@Autowired
private ArticleService articleService;
@Test
public void saveArticle(){
Article article = new Article();
article.setTitle("job find");
article.setContent("some thoughts about finding job");
List<Comment> list = new ArrayList<>();
Comment comment1 = new Comment(content: "review content one");
comment1.setArticle(article);
Comment comment2 = new Comment(content: "review content two");
comment2.setArticle(article);
list.add(comment1);
list.add(comment2);
article.setComments(list);
articleService.saveArticle(article);
}
}

}