Spring Data JPA

浙江大学城市学院
彭 彬
pengb@zucc.edu.cn

Spring Data JPA

JDBC：Java操作数据库（关系数据库）的接口

Hibernate：基于OR-Mapping思想开发的库

JPA：Java官方为OM-Mapping操作提供的对象存储规范（从Java EE 5开始）

https://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

Spring Data JPA：是Spring Data项目的组成部分，为基于JPA快速方便的实现数据库访问提供支持。

https://spring.io/projects/spring-data-jpa

Spring Data JPA官方介绍解析

Spring Data JPA, part of the larger Spring Data family, makes it easy to **easily implement JPA based repositories**. This module deals with enhanced support for JPA based data access layers. It makes it easier to build Spring-powered applications that use data access technologies.

Implementing a data access layer of an application has been cumbersome for quite a while. **Too much boilerplate code has to be written to execute simple queries** as well as perform pagination, and auditing. Spring Data JPA aims **to significantly improve the implementation of data access layers by reducing** the effort to the amount that' s actually needed. As a developer you write your repository interfaces, including custom finder methods, and Spring will **provide the implementation automatically**.

Features(节选)

- Sophisticated **support to build repositories** based on Spring and JPA
- Support for **Querydsl predicates** and thus type-safe JPA queries
- **Pagination** support, **dynamic query execution**, ability to integrate custom data access code

Spring Data项目

Spring Data项目尝试为各种数据访问方案提供统一的解决方案，并且通过最核心的Repository对象来完成核心的数据库操作。比如下面分别是访问图数据库Neo4j、非关系数据库MongoDB和关系数据库MySQL的方法（下面的查询都是自动实现，需要写的代码就是下面这些，而且模式非常相似）：

Neo4j

```
@Component
public interface SchoolRepository extends Neo4jRepository<SchoolNode, Long> {
    SchoolNode findByCode(String code);

    @Query("MATCH (s:School)-[:USE_MODEL]->(m:ModelRoot),p=(m:ModelRoot)<-[*]-(l:ModelLeaf)
            "WHERE id(s) = $0 AND id(m) = $1 " +
            "RETURN id(l) as id, l.title as title, l.code as code, reduce(v = \"\", x IN no
            "ORDER BY l.code")
    List<ModelLeafResult> queryModelLeafOfSchool(Long schoolId, Long modelId);
```

MongoDB

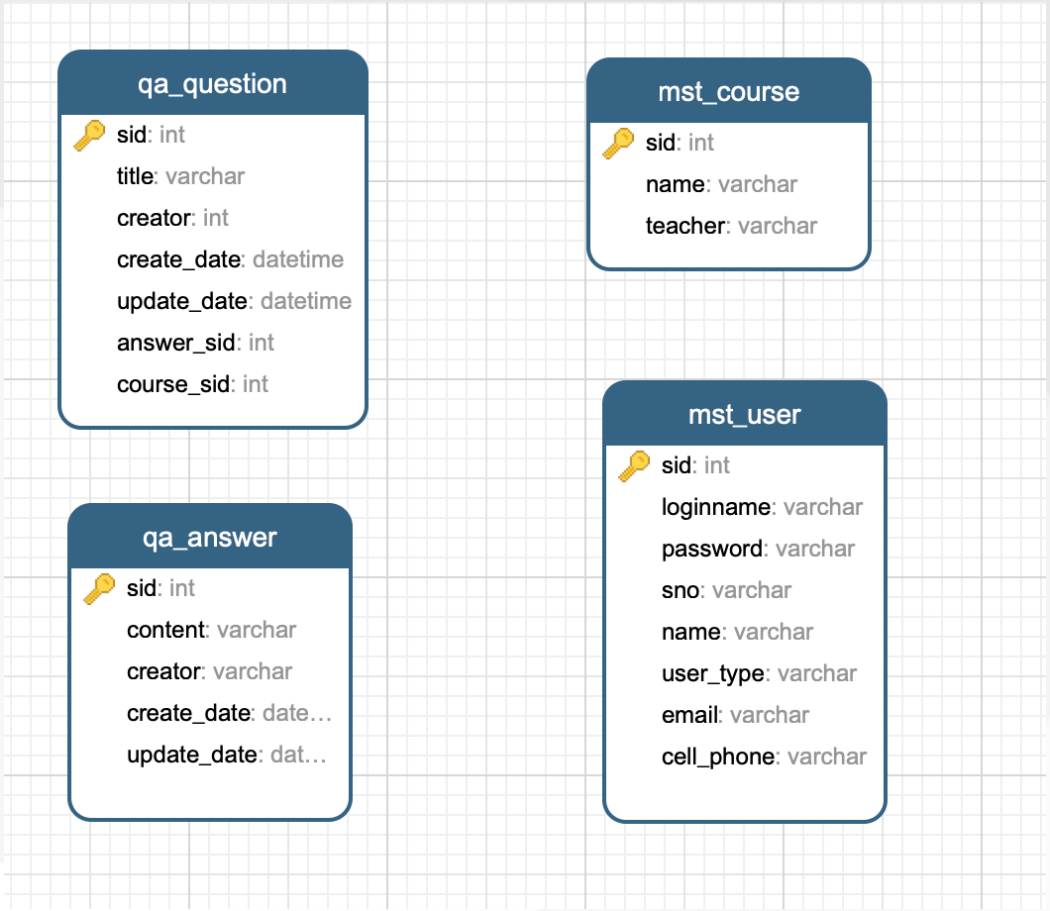
```
public interface PersonRepository extends MongoRepository<Person, String>

  @Query("{ 'firstname' : ?0 }")
  List<Person> findByThePersonsFirstname(String firstname);

}
```

MySQL

```
public interface UserRepository extends JpaRepository<User, Long> {

  @Query("select u from User u where u.firstname like %?1")
  List<User> findByFirstnameEndsWith(String firstname);
}
```

基于我们简单设计的数据库来探索JPA

作为示例,我们沿用上次课程设计了几个简单的表格（注意：这是示例！）



基于JPA我们可以很快完成以下操作：

1）基本的增删改查询

2）处理分页

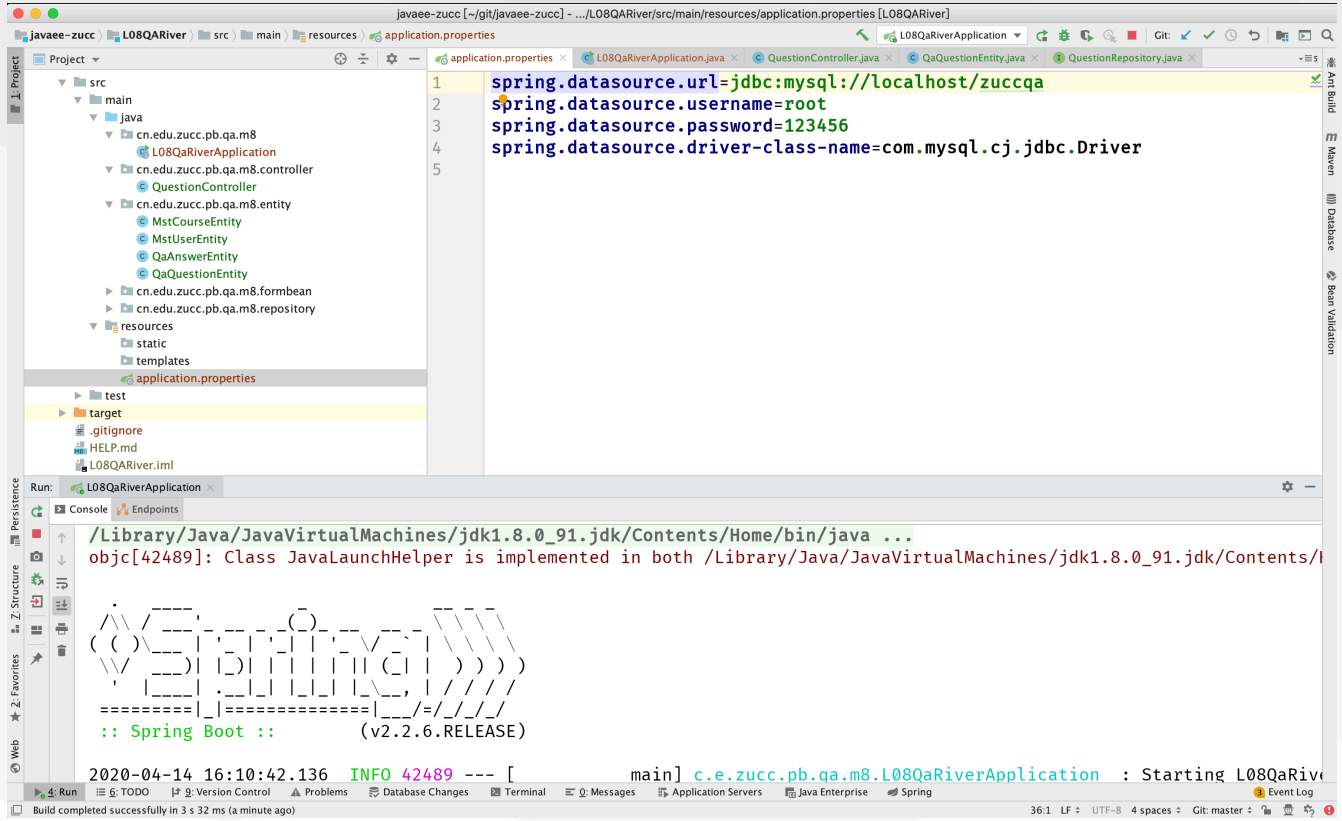
3）处理多表查询

4）基于SQL语句的各种复杂查询统计

再次重申：我们为什么不设计外键？

配置SpringBoot数据源参数

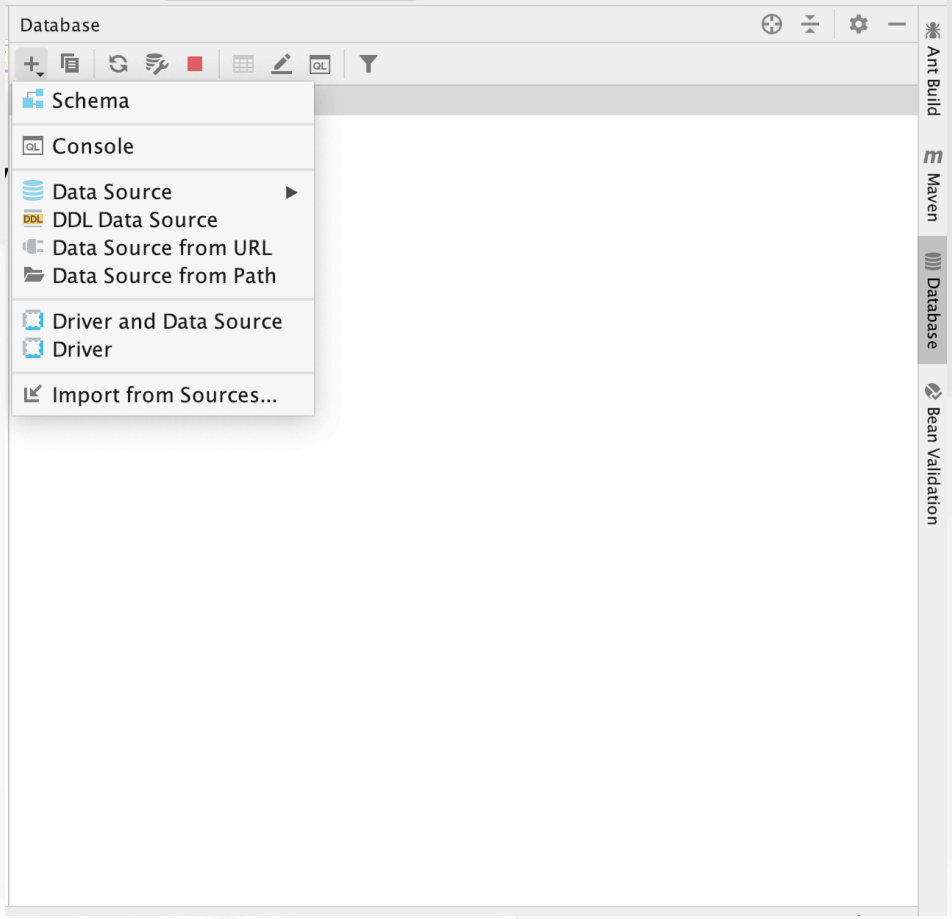
我们使用spring boot配置数据源，特别简单，在application.properties文件中输入下面的信息即可。（沿用上一课的方式）

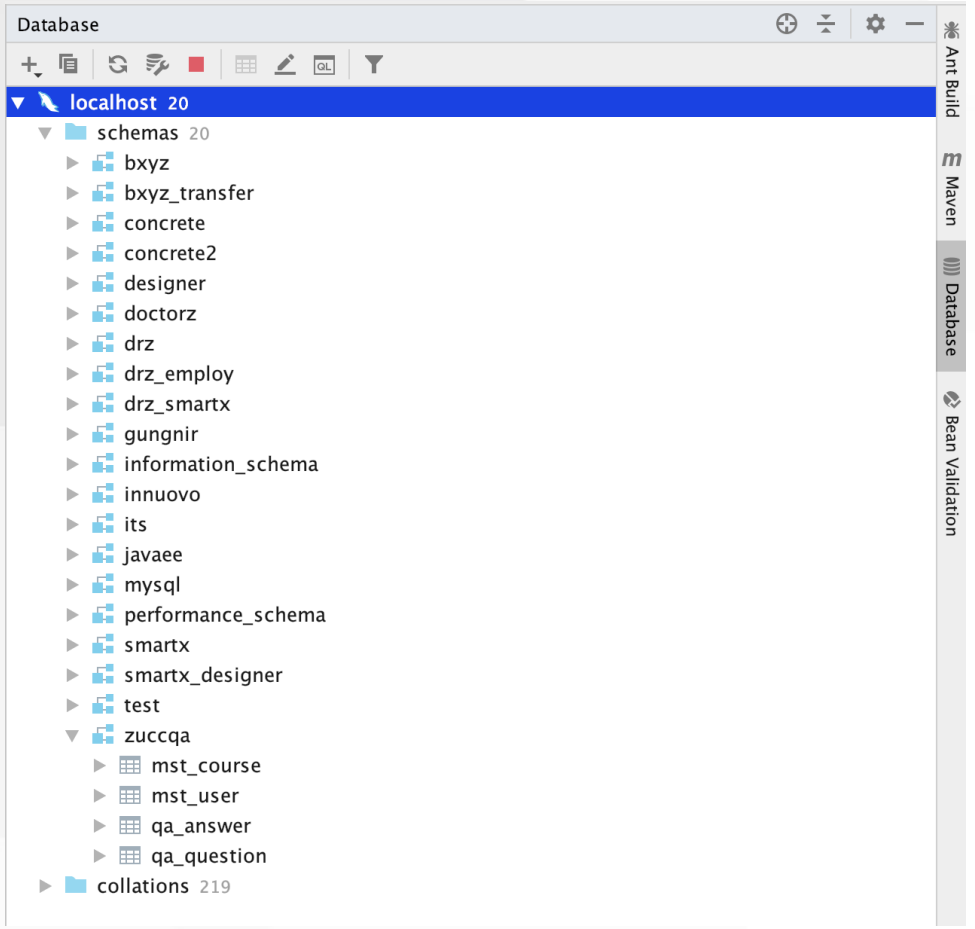


https://spring.io/guides/gs/accessing-data-mysql/

在Idea中配置数据源

为了方便操作和后续生成各种Entity类，可以通过Idea配置数据源

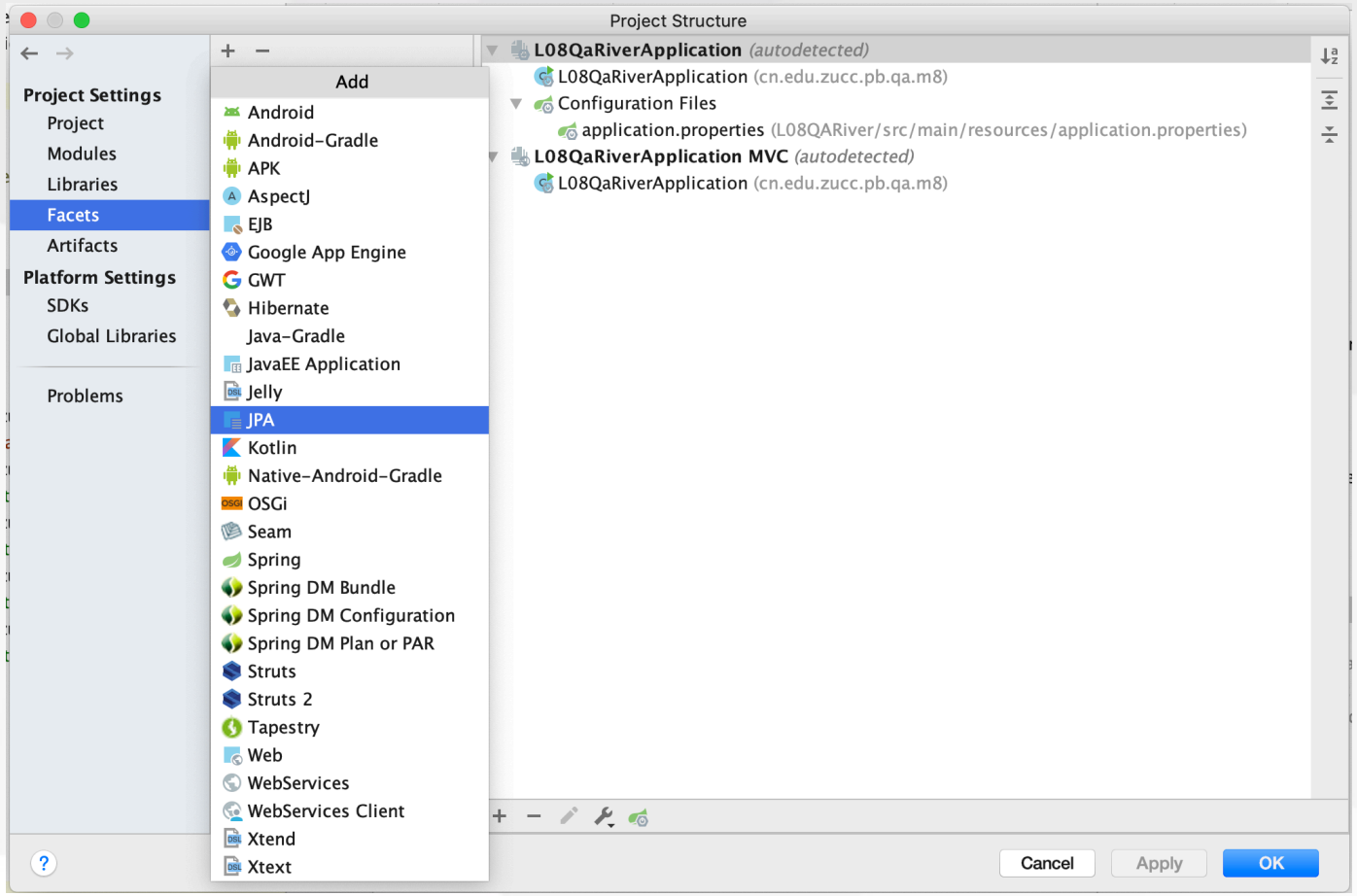




在右侧打开Database对话框，然后增加一个Datasource连接到你的Mysql数据库，连接成功后可以打开库和表格

为项目增加JPA和Hibernate的支持

在Idea中增加JPA和Hibernate支持后，可以使用插件工具自动创建Entity类



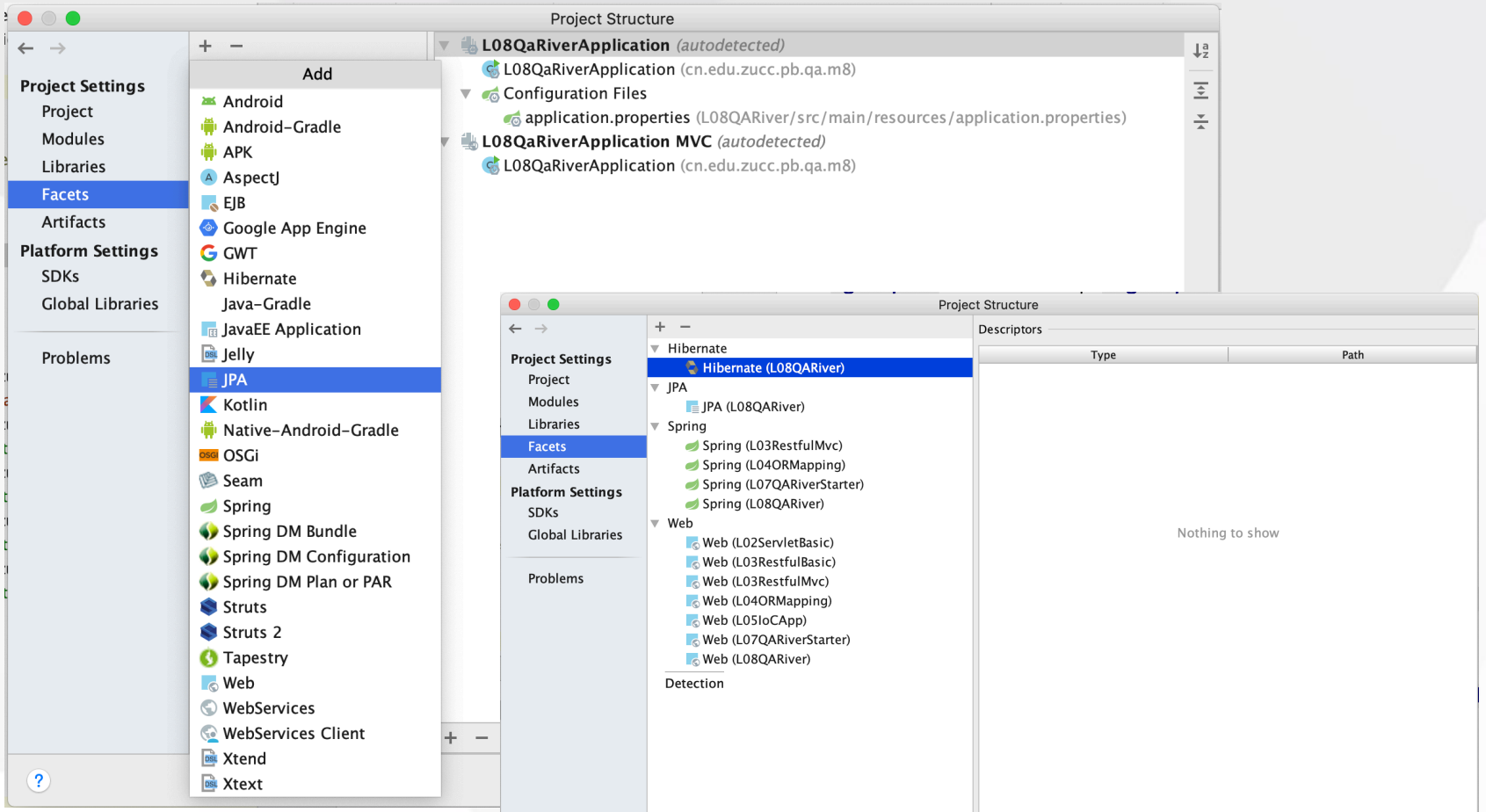
在项目配置中，增加Facets的支持。

增加的Facets包括

- JPA
- Hibernate

为项目增加JPA和Hibernate的支持

在Idea中增加JPA和Hibernate支持后，可以使用插件工具自动创建Entity类

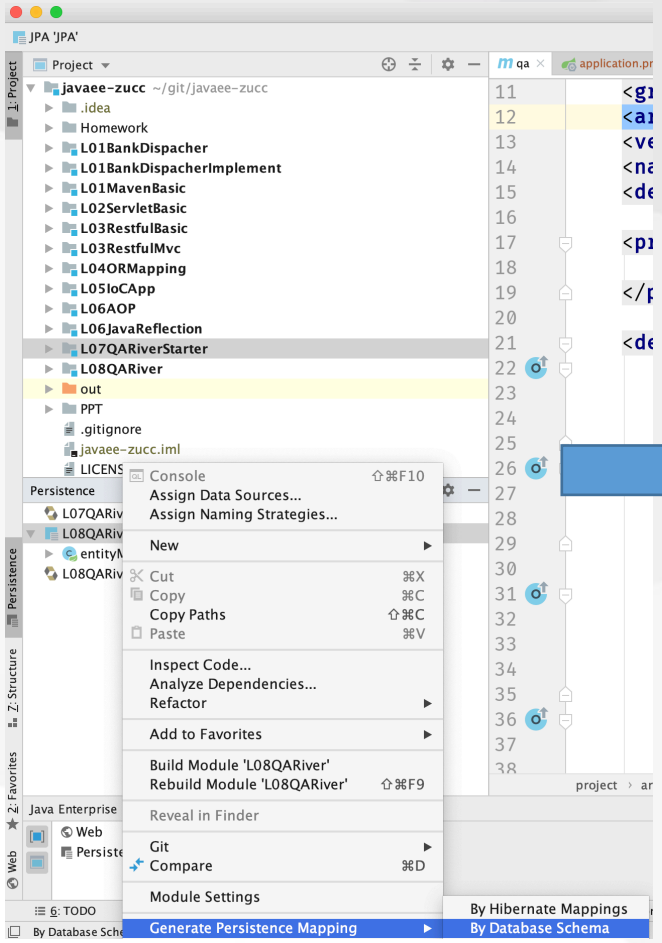


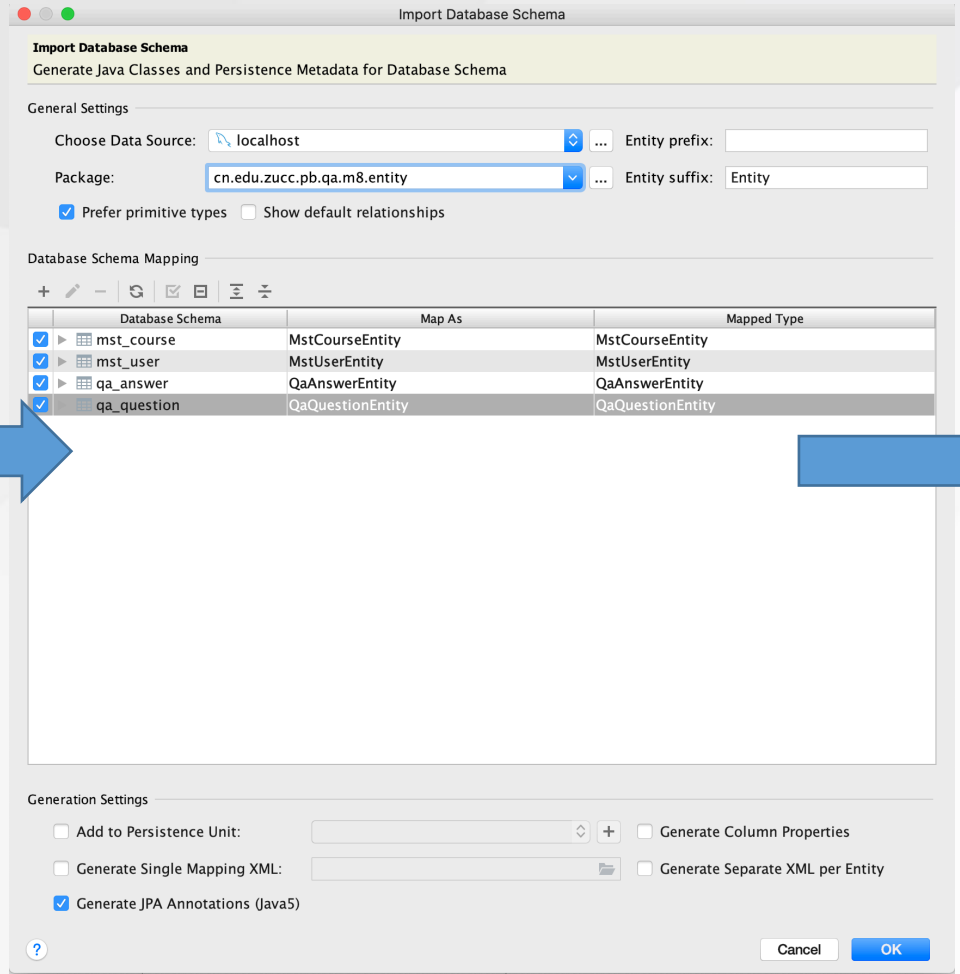
在项目配置中，增加Facets的支持。增加的Facets包括

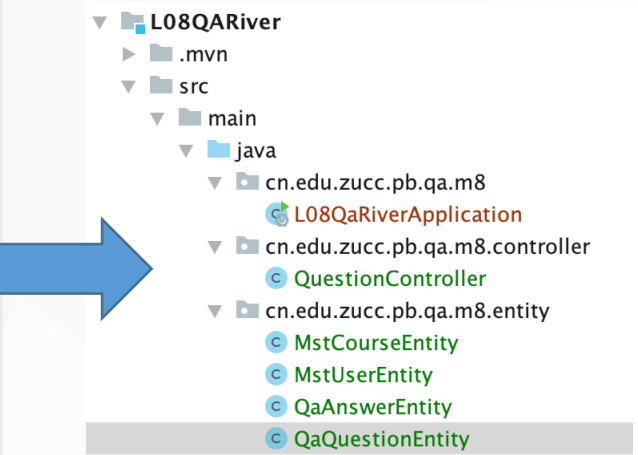
- JPA
- Hibernate

可以自动从数据库表生成Entity类

上次课中我们是自己手工创建了Entity类（针对Question表），本次我们使用工具一次性为示例的四张表增加对应的Entity类

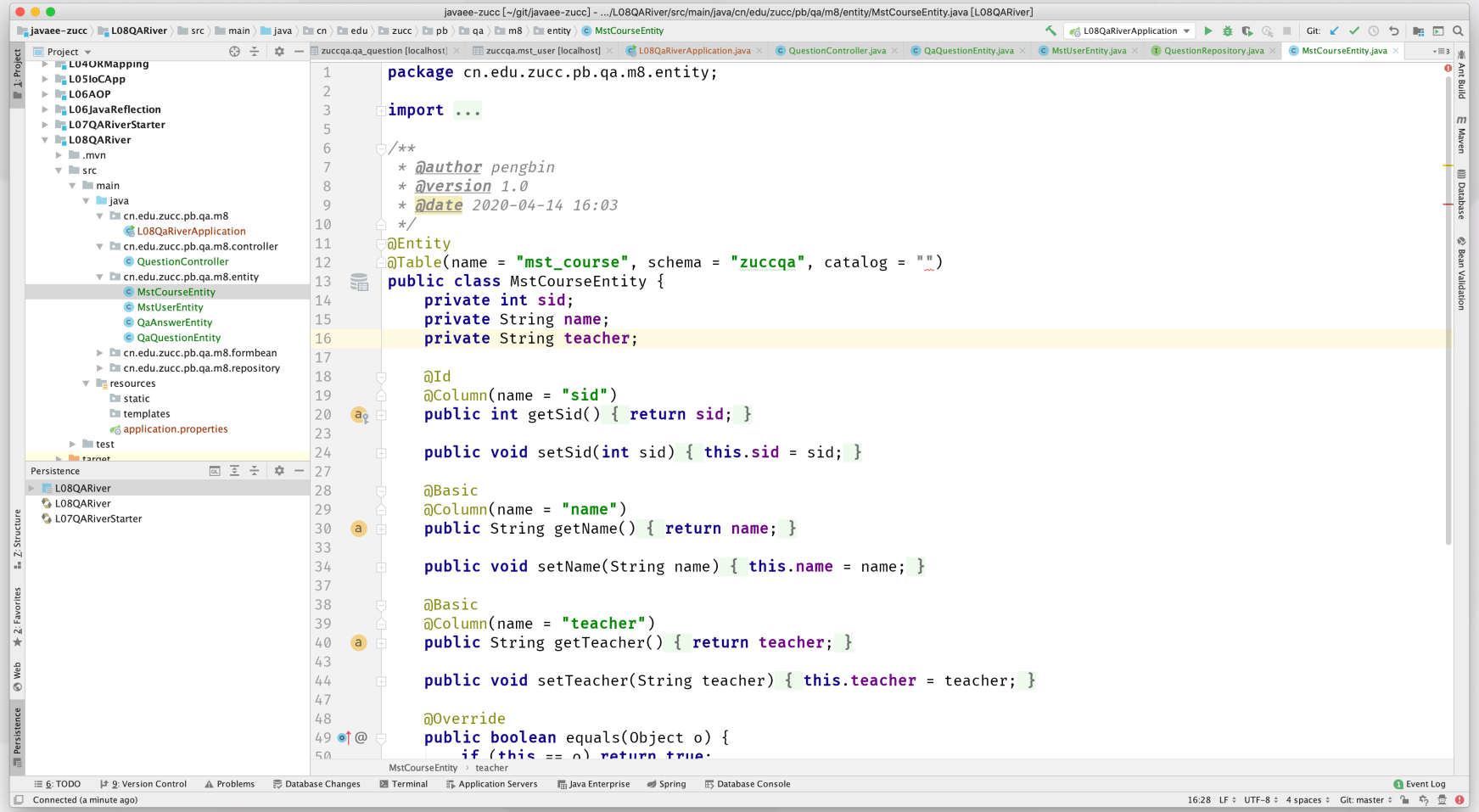






JPA Entity类

JPA的Entity类是普通的Java Class。但是这个Java Class有一些特定的标注



@Entity

@Table

@Id

@Column

@CreationTimestamp

@UpdateTimestamp

JPA定义了非常多的标注，参考下面的官方列表

https://docs.oracle.com/javaee/7/api/javax/persistence/package-summary.html

Entity类

当然我们可以修改Entity类，以支持我们的更多特性，比如增加timestamp标注

```
@Basic
@Column(name = "create_date")
@CreationTimestamp
public Timestamp getCreateDate() { return createDate; }

public void setCreateDate(Timestamp createDate) { this.createDate = createDate; }

@Basic
@Column(name = "update_date")
@UpdateTimestamp
public Timestamp getUpdateDate() { return updateDate; }
```

数据库操作的核心Repository

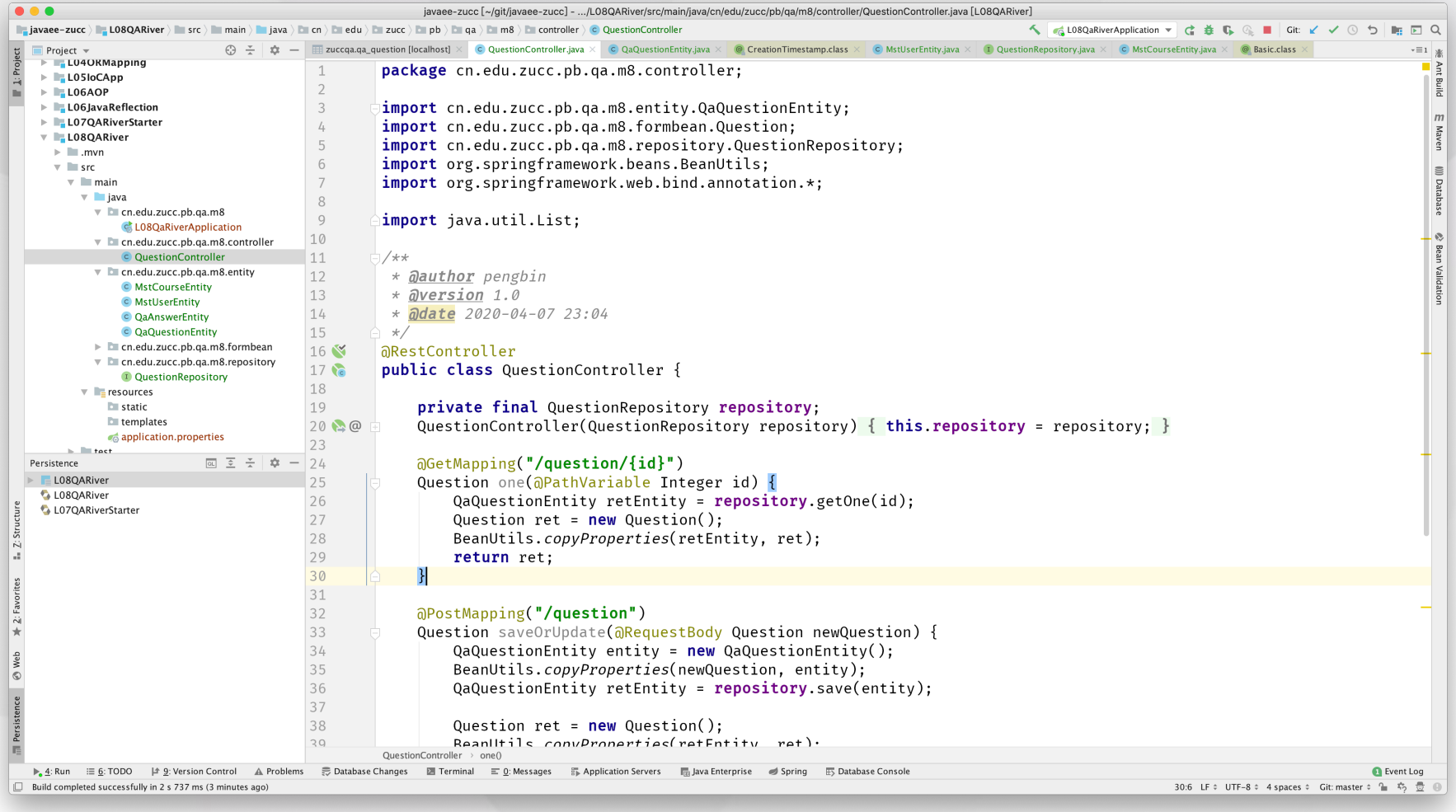
我们通过接口继承JpaRepository就具备的主要的增删改查接口，不用写任何SQL语句。

```java
package cn.edu.zucc.pb.qa.repositories;

import cn.edu.zucc.pb.qa.entity.QuestionEntity;
import org.springframework.data.jpa.repository.JpaRepository;

/**
 * @author pengbin
 * @version 1.0
 * @date 2020-04-07 23:25
 */
public interface QuestionRepository extends JpaRepository<QuestionEntity, Integer> {
}
```

*具体的SQL操作实现由spring data框架帮我们完成了，所以之前说的自动产生查询语句就体现在这些地方

Spring MVC

我们先使用Spring MVC框架创建一个Controller，以构造WebApi



@RestController

@GetMapping

@PostMapping

依赖注入

Controller中需要repository，这个时候我们就使用了Spring的DI功能注入一个Repository。构造函数注入方式：

```
@RestController
public class QuestionController {

    private final QuestionRepository repository;
    QuestionController(QuestionRepository repository){
        this.repository = repository;
    }
```

Spring会自动发现依赖关系，然后进行注入。

实现一个使用ID查询数据

我们在Controller中增加一个接口，使用Question的id来获取其实例数据

```
@RestController
public class QuestionController {

    private final QuestionRepository repository;
    QuestionController(QuestionRepository repository) { this.repository = repository; }

    @GetMapping("/question/{id}")
    Question one(@PathVariable Integer id) {
        QaQuestionEntity retEntity = repository.getOne(id);
        Question ret = new Question();
        BeanUtils.copyProperties(retEntity, ret);
        return ret;
    }
```

我们获取id参数后，直接调用repository的getOne方法就可以获取对应的Entity对象，其需要的select语句及其执行代码全部由spring data框架代劳了。

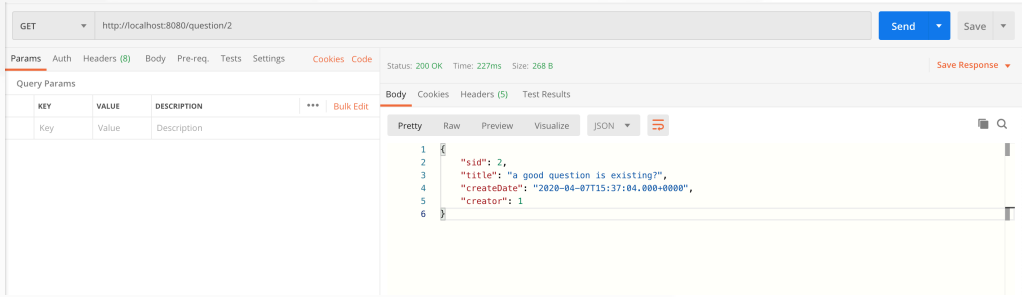
再次强调为啥我们要创建FormBean对象和Entity对象，然后复制呢？

实现一个使用ID查询数据

我们在Controller中增加一个接口，使用Question的id来获取其实例数据

```java
@RestController
public class QuestionController {

    private final QuestionRepository repository;
    QuestionController(QuestionRepository repository) { this.repository = repository; }

    @GetMapping("/question/{id}")
    Question one(@PathVariable Integer id) {
        QaQuestionEntity retEntity = repository.getOne(id);
        Question ret = new Question();
        BeanUtils.copyProperties(retEntity, ret);
        return ret;
    }
```

我们获取id参数后，直接调用repository的getOne方法就可以获取对应的Entity对象，其需要的select语句及其执行代码全部由spring data框架代劳了。

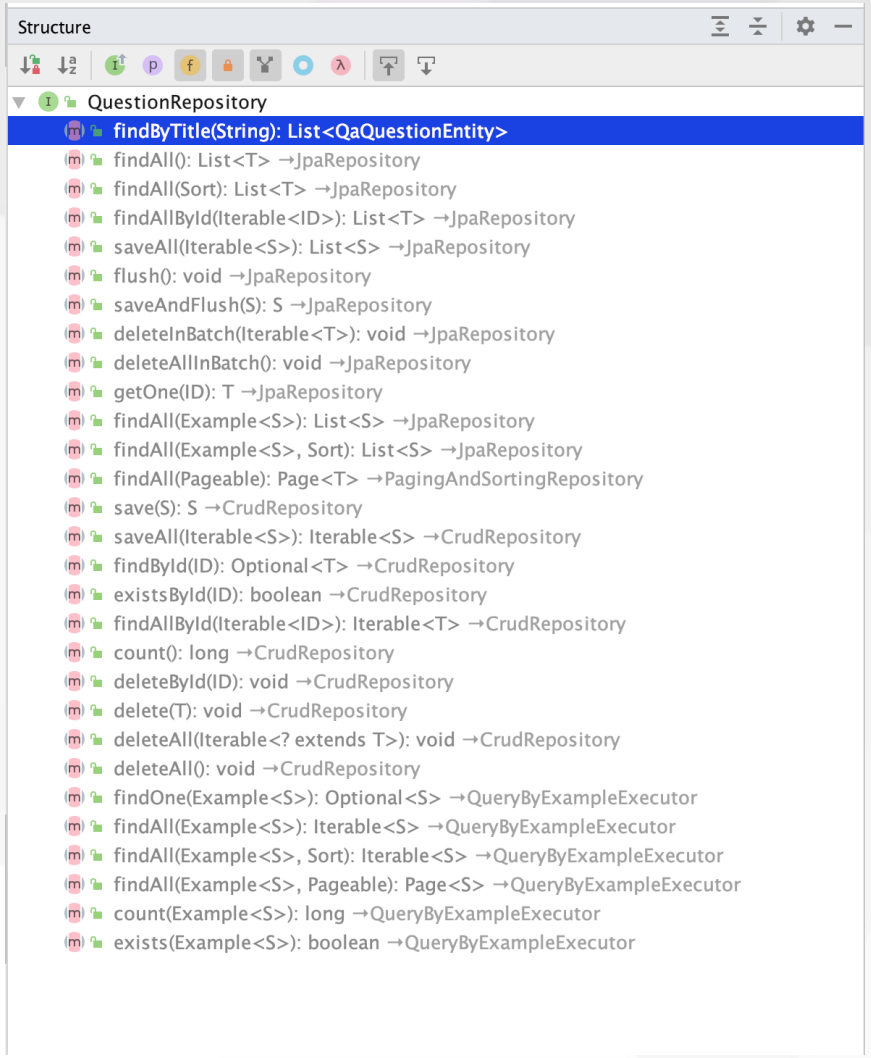


再次强调为啥我们要创建FormBean对象

和Entity对象，然后复制呢？

Repository里面有什么

通过打开Idea的Structure窗口，我们可以仔细观察到其包含了很多方法，这些方法继承自多个父接口



Findxxxx:这一组方法实现了select功能

Save：实现insert或者update功能

Delete：实现delete功能

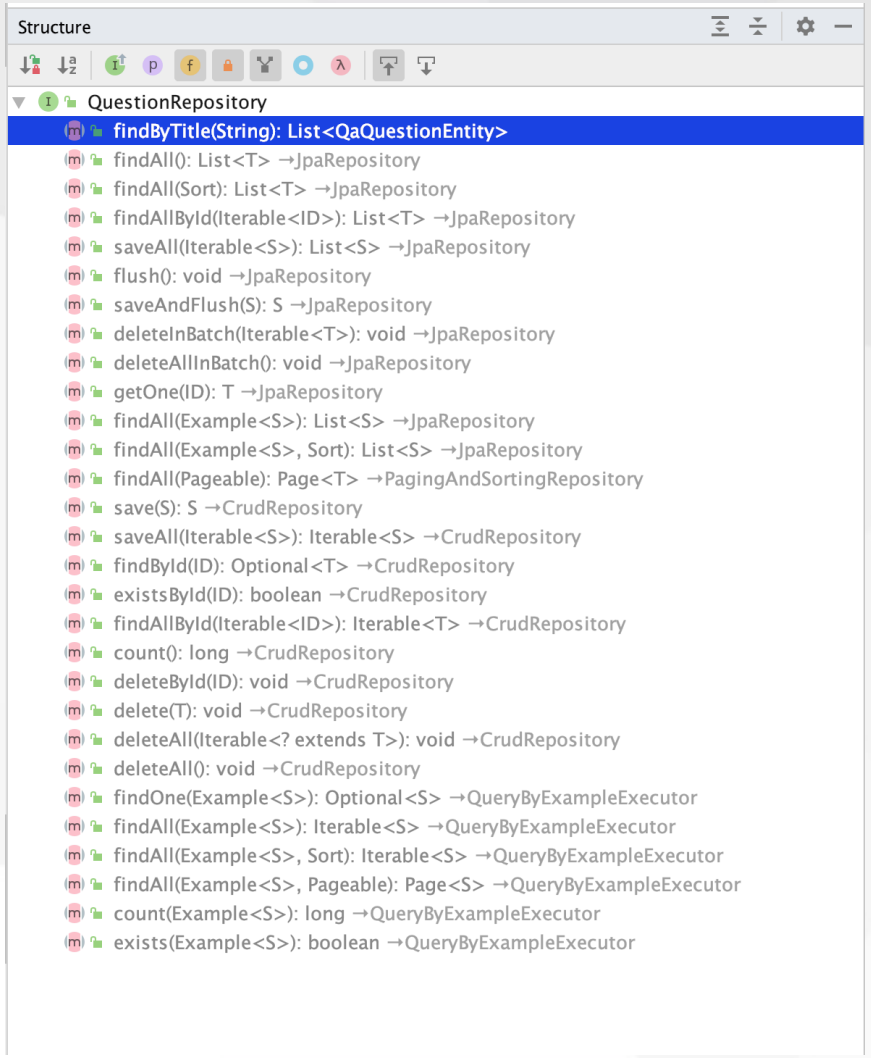
最常用的增删改查功能使用上面的功能就可以了；还有一些常用方法：

Count：计数

Existsxxxx:判断存在

Repository基本使用

我们增加一个接口来使用基本的CRUD功能



Findxxxx:这一组方法实现了select功能

Save：实现insert或者update功能

Delete：实现delete功能

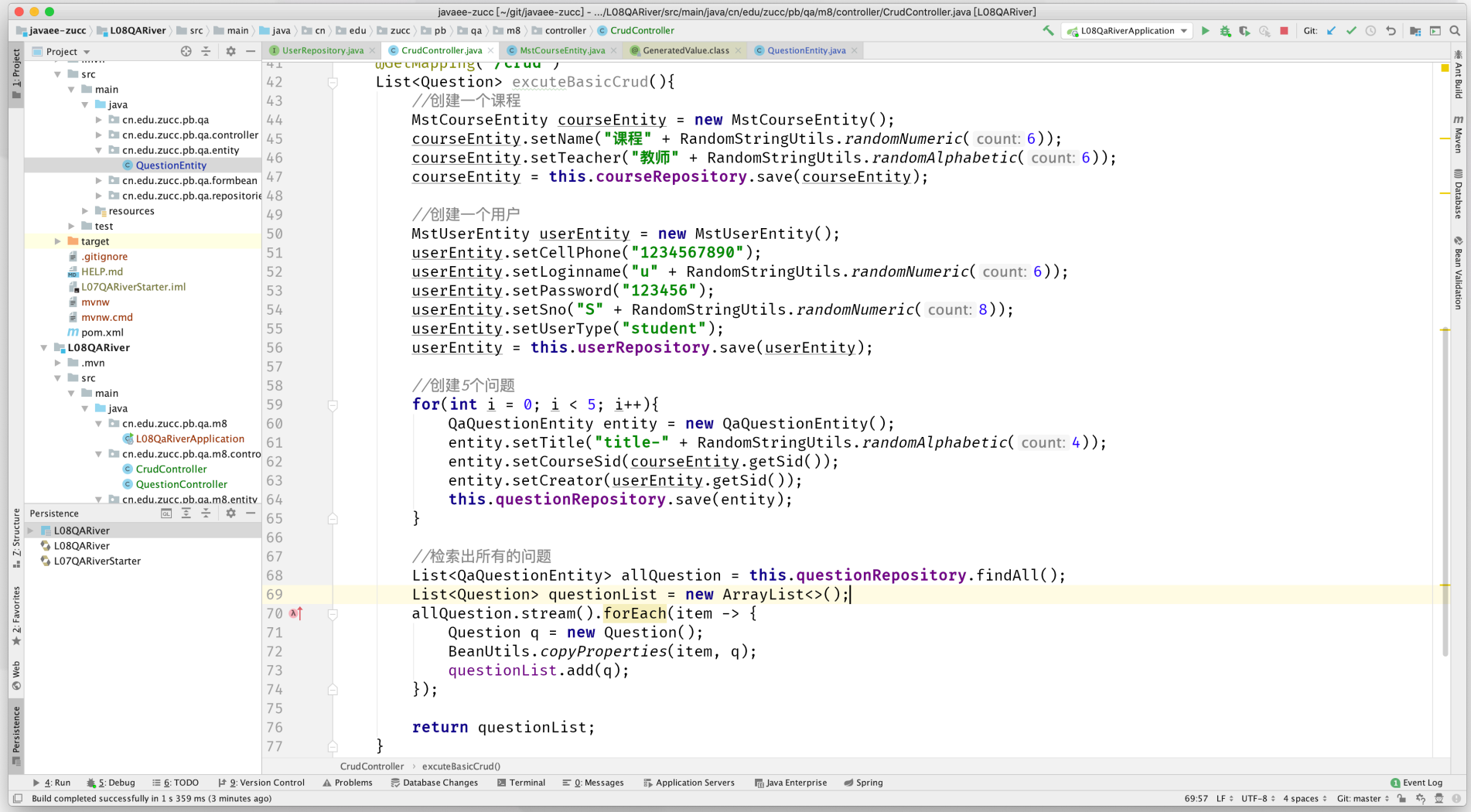
最常用的增删改查功能使用上面的功能就可以了；还有一些常用方法：

Count：计数

Existsxxxx:判断存在

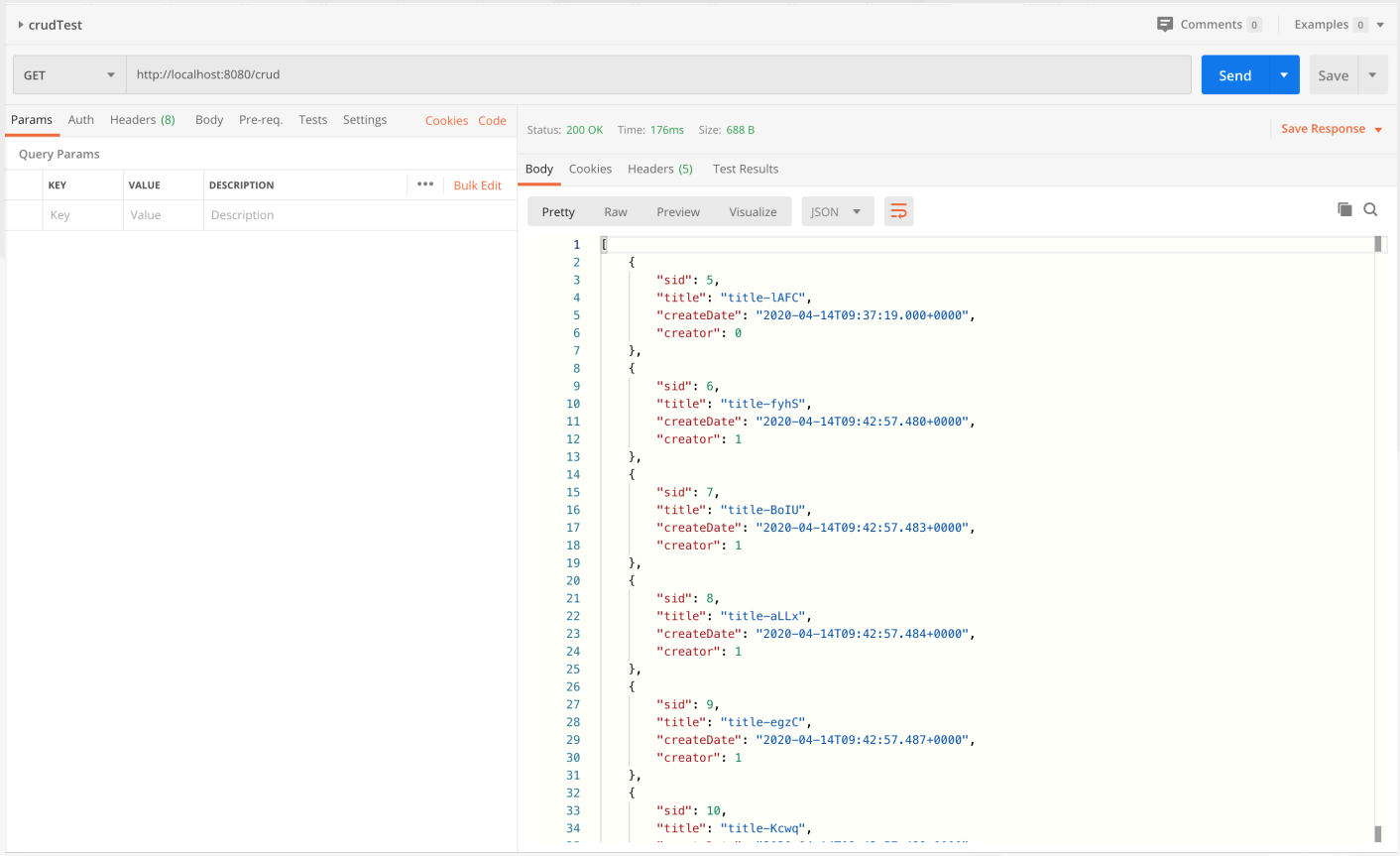
Repository基本使用

我们增加一个接口来使用基本的CRUD功能



Repository基本使用

我们增加一个接口来使用基本的CRUD功能



Objects | qa_question @zucc... | mst_course @zucc... | mst_user @zuccqa... | mst_course @zucc... | mst_user @zuccqa... | qa_answer @zuccq...

sid	title	creator	create_date	update_date	answer_sid	course_sid
6	title-fyhS	1	2020-04-14 04:42:57	2020-04-14 04:42:57	[Null]	1
7	title-BoIU	1	2020-04-14 04:42:57	2020-04-14 04:42:57	[Null]	1
8	title-aLLx	1	2020-04-14 04:42:57	2020-04-14 04:42:57	[Null]	1
9	title-egzC	1	2020-04-14 04:42:57	2020-04-14 04:42:57	[Null]	1
10	title-Kcwq	1	2020-04-14 04:42:57	2020-04-14 04:42:57	[Null]	1

使用“约定”查询

我们通过Spring Data JPA的“名称约定”来完成一个查询接口，同样我们不需要编写任何查询的实现代码，依赖Spring自动帮我们完成。我们只需做“合适”的方法命名就可以了

```
/**
 * @author pengbin
 * @version 1.0
 * @date 2020-04-07 23:25
 */
public interface QuestionRepository extends JpaRepository<QaQuestionEntity, Integer> {
    List<QaQuestionEntity> findByTitleLike(String title);
    List<QaQuestionEntity> findByTitleLikeOrderByTitleDesc(String title);
}
```

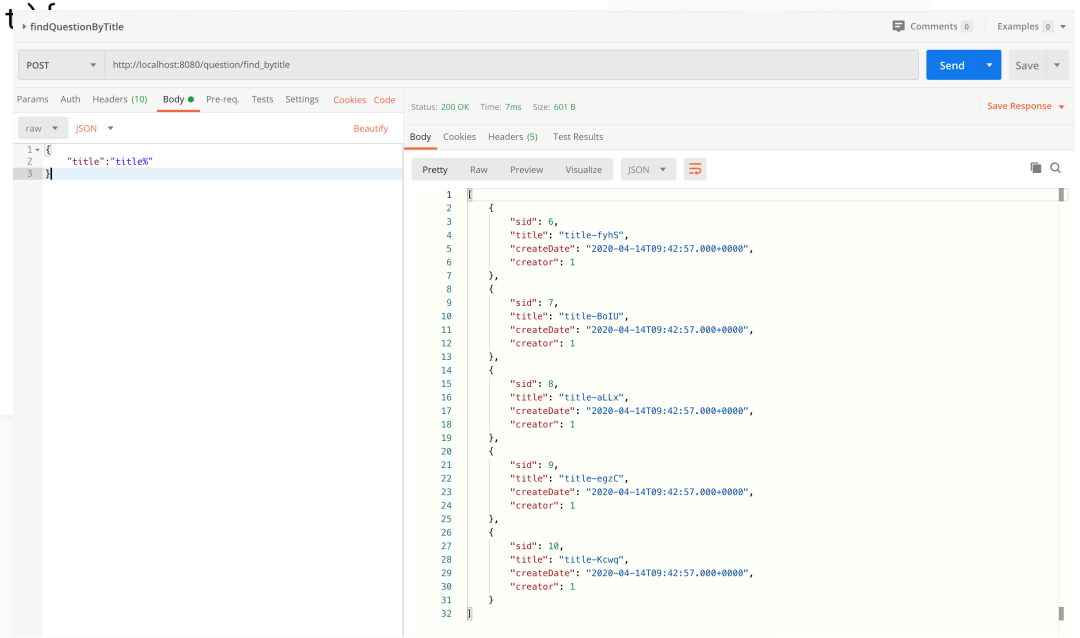
我们通过“名称约定”来完成了两个查询

- 根据title的模糊查询
- 根据title的模糊查询并排序

使用“约定”查询

直接调用我们“只写了名称”的方法即可使用

```
@PostMapping("/question/find_bytitle")
List<Question> findByTitle(@RequestBody Map<String, String> queryExample) {
    return convert(repository.findByTitleLike(queryExample.get("title")));
}

@PostMapping("/question/find_bytitle_orderby")
List<Question> findByDateBetween(@RequestBody Map<String, String> queryExample) {
    return convert(repository.findByTitleLikeOrderByTitleDesc(queryExample.get("title")));
}

private List<Question> convert(List<QaQuestionEntity> entityList){
    List<Question> questionList = new ArrayList<>();
    entityList.stream().forEach(item -> {
        Question q = new Question();
        BeanUtils.copyProperties(item, q);
        questionList.add(q);
    });

    return questionList;
}
```



有哪些命名模式?

Spring Data支持非常多常用的命名模式，以支持常见的快速查询，非常方便

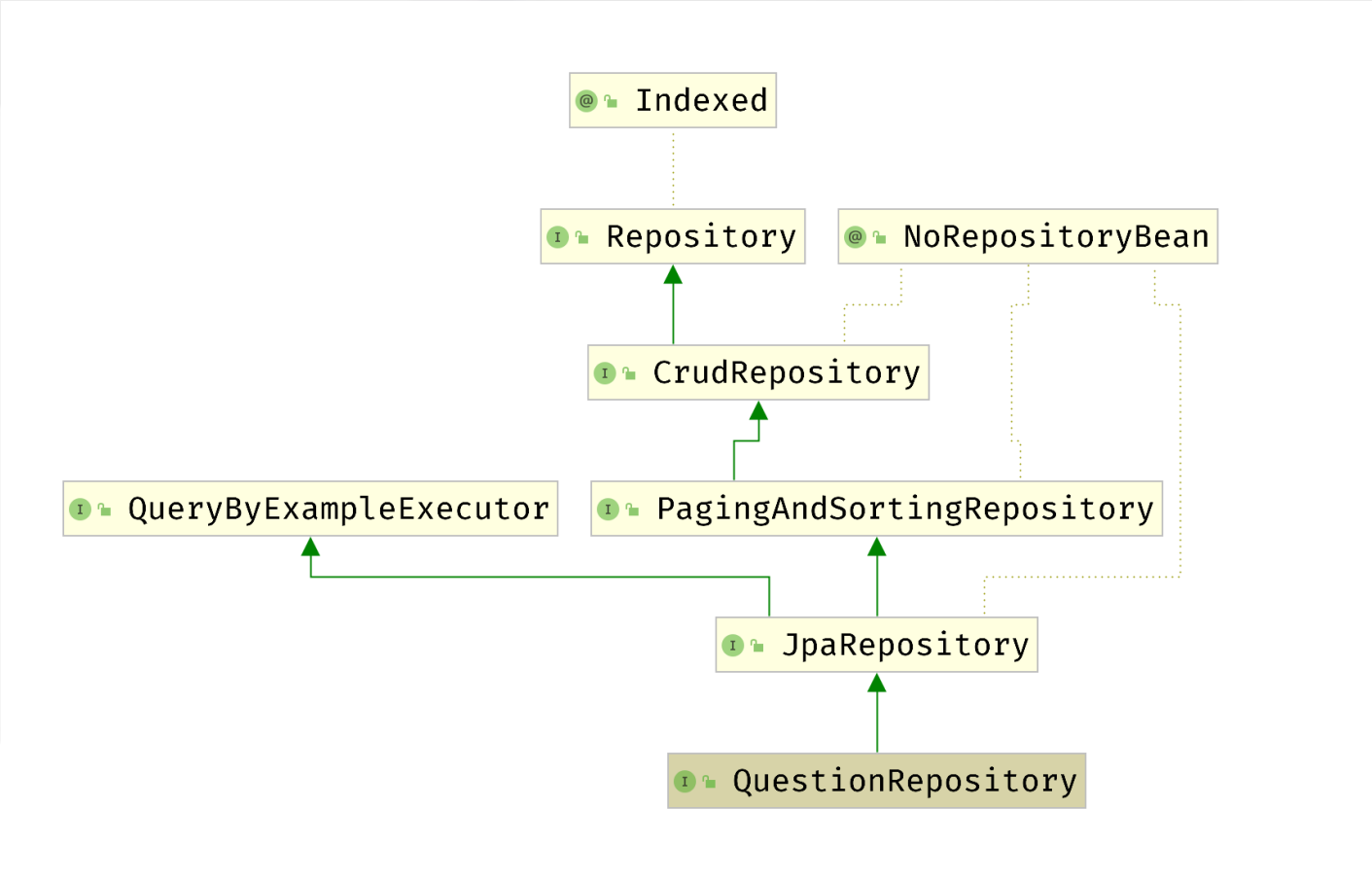
Table 3. Supported keywords inside method names

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is, Equals	findByFirstname, findByFirstnameIs, findByFirstnameEquals	… where x.firstname = ?1
Between	findByStartDateBetween	… where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull, Null	findByAge(Is)Null	… where x.age is null
IsNotNull, NotNull	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods

分页查询

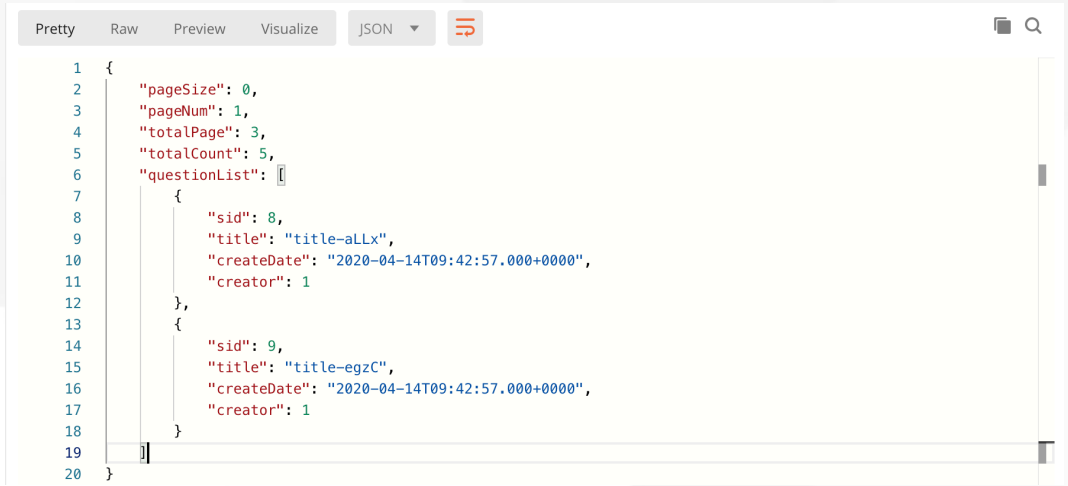
Spring Data JPA提供了分页模式。我们可以通过下图清楚的看到我们的Repository的继承位置，而PagingAndSortingRepository父接口提供了分页的功能，其中提供的分页功能可以直接使用



分页查询

直接使用父接口提供的支持分页的findAll方法如下：

```java
@PostMapping("/question/find_all_pagination")
QuestionPage findPagination() {
    Page<QaQuestionEntity> thisPage = repository.findAll(PageRequest.of( page: 1,  size: 2));
    QuestionPage retPage = new QuestionPage();
    retPage.setPageNum(1);
    retPage.setPageSize(2);
    retPage.setTotalPage(thisPage.getTotalPages());
    retPage.setTotalCount(thisPage.getTotalElements());
    retPage.setQuestionList(convert(thisPage.getContent()));
    return  retPage;
}
```



直接使用sql语句查询

我们还可以直接通过使用sql语句查询，我们使用Query标注可以直接写sql语句，通过映射返回值的方式获取结果，比如：

```
public interface AnswerRepository extends JpaRepository<QaAnswerEntity, Integer> {
    @Query(value="select count(*) from qa_answer", nativeQuery = true)
    Long countAll();
```

直接使用新增加的方法：

```
@PostMapping("/answer/query_sql")
String queryBySql(@RequestBody Map<String, String> queryExample) {
    return repository.countAll().toString();
}
```

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.at-query
https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query.spel-expressions

多表查询~直接使用SQL语句

我们同样可以使用SQL语句查询来解决复杂的查询或者多表连接的问题

```
public interface AnswerRepository extends JpaRepository<QaAnswerEntity, Integer> {
    @Query(value="select count(*) from qa_answer", nativeQuery = true)
    Long countAll();

    @Query(value="select u.sid as sid, u.loginname as name, count(q.sid) as answerCount " +
            "from qa_answer q, mst_user u where q.creator = u.sid and u.sid = ?1", nativeQuery = true)
    AnswerUserView countAllByUser(int uid);

}
```

其中返回值可以使用Projection机制来获取（注意interface，get方法及上面sql语句返回字段名匹配

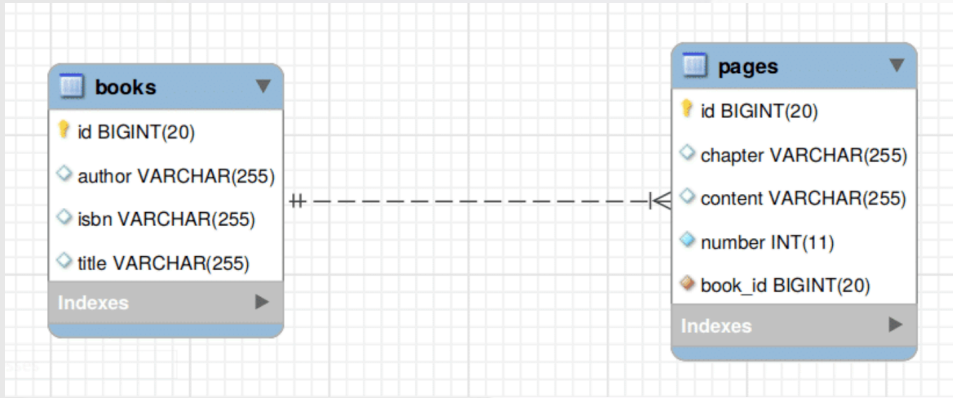
```
/**
 * @author pengbin
 * @version 1.0
 * @date 2020-04-14 21:56
 */
public interface AnswerUserView {
    int getSid();
    String getName();
    int getAnswerCount();
}
```

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#projections

多表查询~使用对象映射

如果我们创建了外键关系，我们可以使用JPA提供的OneToMany、ManyToMany等标注进行操作。可以参考下面的文章内容

https://attacomsian.com/blog/spring-data-jpa-one-to-many-mapping



Page Entity

```
@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "book_id", nullable = false)
private Book book;
```

Book Entity

```
@OneToMany(mappedBy = "book", fetch = FetchType.LAZY,
        cascade = CascadeType.ALL)
private Set<Page> pages;
```

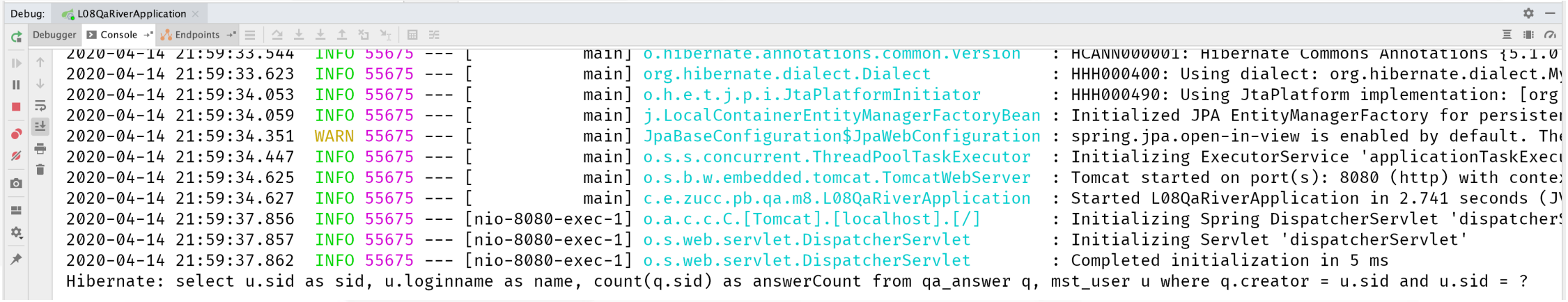
开启SQL语句日志

为了方便查看Spring Data JPA发往数据库的日志，我们可以打开SQL语句日志

JpaRepository.class | application.properties | AnswerController.java | AnswerRepository.java | AnswerUserView.java

```
spring.datasource.url=jdbc:mysql://localhost/zuccqa
spring.datasource.username=root
spring.datasource.password=123456
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql=true
```

打开sql日志后可以在控制台中查看SQL语句日志：



END

Pb&Lois