# 1. 读写分离

> *读写分离的前提是，mysql做了主从，也就是一主多从的结构，或者多主多从的结构*

**为什么要做读写分离?**

1. 应用的第一个瓶颈一定是在数据库（磁盘的读写速度是最慢的）
2. 写操作会加锁，加锁的后果就是读必须等写操作完成，极大的拖累了系统的运行速度
3. 基本上所有的应用都是读多写少

## 1.1 MYSQL主从复制原理

> *mysql主从复制的基础是bin-log日志，slave通过一个I/O线程与主服务器保持通信，并监控master的二进制日志文件的变化，如果发现master二进制日志文件发生变化，则会把变化复制到自己的中继日志中，然后slave的一个SQL线程会把相关的"事件"执行到自己的数据库中，以此实现从数据库和主数据库的一致性，也就实现了主从复制。*

**注意：bin-log是可以做数据恢复的，所以生产环境部署mysql一定要开启bin-log**

### Binlog 的日志格式

支持三种格式类型:

1. STATEMENT：基于SQL语句的复制（statement-based replication, SBR）
    1. 每一条会修改数据的sql都会记录在binlog中
    2. 有些函数在主从复制过程中不被识别
2. ROW：基于行的复制（row-based replication, RBR）
    1. 不记录每一条SQL语句的上下文信息，仅需记录哪条数据被修改了，修改成了什么样子了
    2. 会产生大量的日志
3. MIXED：混合模式复制（mixed-based replication, MBR）
    1. MySQL会根据执行的SQL语句选择日志保存方式

在 MySQL 5.7.7 之前，默认的格式是 `STATEMENT`，在 MySQL 5.7.7 及更高版本中，默认值是`ROW`。日志格式通过 binlog-format 指定，如 `binlog-format=STATEMENT`、`binlog-format=ROW`、`binlog-format=MIXED`。

## 1.2 ShardingJDBC

官网：[http://shardingsphere.apache.org/index_zh.html](http://shardingsphere.apache.org/index_zh.html)

> *用到的是其中一个组件：ShardingSphere-JDBC*

定位为轻量级 Java 框架，在 Java 的 JDBC 层提供的额外服务。 它使用客户端直连数据库，以 jar 包形式提供服务，无需额外部署和依赖，可理解为增强版的 JDBC 驱动，完全兼容 JDBC 和各种 ORM 框架。

- 适用于任何基于 JDBC 的 ORM 框架，如：JPA, Hibernate, Mybatis, Spring JDBC Template 或直接使用 JDBC。
- 支持任何第三方的数据库连接池，如：DBCP, C3P0, BoneCP, Druid, HikariCP 等。
- 支持任意实现 JDBC 规范的数据库，目前支持 MySQL，Oracle，SQLServer，PostgreSQL 以及任何遵循 SQL92 标准的数据库。

### 背景

面对日益增加的系统访问量，数据库的吞吐量面临着巨大瓶颈。 对于同一时刻有大量并发读操作和较少写操作类型的应用系统来说，将数据库拆分为主库和从库，主库负责处理事务性的增删改操作，从库负责处理查询操作，能够有效的避免由数据更新导致的行锁，使得整个系统的查询性能得到极大的改善。

通过一主多从的配置方式，可以将查询请求均匀的分散到多个数据副本，能够进一步的提升系统的处理能力。 使用多主多从的方式，不但能够提升系统的吞吐量，还能够提升系统的可用性，可以达到在任何一个数据库宕机，甚至磁盘物理损坏的情况下仍然不影响系统的正常运行。

## 1.3 使用

> *通过引入依赖，添加对应的配置，即可轻松集成读写分离*

`sso`和`web`都需要引入读写分离，因为这两个系统面向的是海量的用户

```
 <dependency>
            <groupId>org.apache.shardingsphere</groupId>
            <artifactId>shardingsphere-jdbc-core-spring-boot-
starter</artifactId>
            <version>5.0.0-beta</version>
        </dependency>
```

配置:

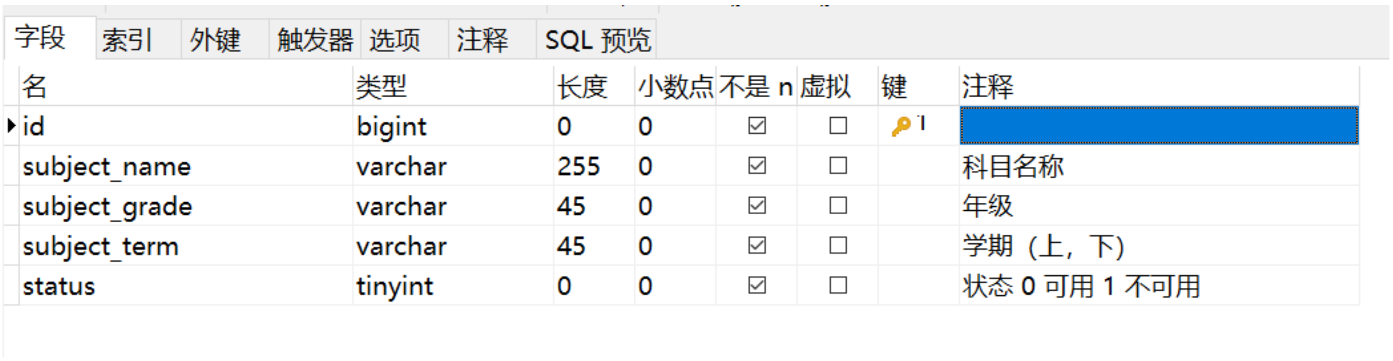
```
#读写分离
##shardingsphere配置
spring.shardingsphere.datasource.common.type=com.zaxxer.hikari.HikariDat
aSource
spring.shardingsphere.datasource.common.driver-class-
name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.common.username=root
spring.shardingsphere.datasource.common.password= root

## 一主2从
spring.shardingsphere.datasource.names=master,slave0,slave1

# 配置第 1 个数据源
spring.shardingsphere.datasource.master.type=com.zaxxer.hikari.HikariDat
aSource
spring.shardingsphere.datasource.master.driver-class-
name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.master.jdbc-
url=jdbc:mysql://localhost:3306/xt?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC
spring.shardingsphere.datasource.master.username=root
spring.shardingsphere.datasource.master.password=root

# 配置第 2 个数据源
spring.shardingsphere.datasource.slave0.type=com.zaxxer.hikari.HikariDat
aSource
spring.shardingsphere.datasource.slave0.driver-class-
name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.slave0.jdbc-
url=jdbc:mysql://localhost:3306/xt?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC
spring.shardingsphere.datasource.slave0.username=root
spring.shardingsphere.datasource.slave0.password=root
# 配置第 3 个数据源
spring.shardingsphere.datasource.slave1.type=com.zaxxer.hikari.HikariDat
aSource
spring.shardingsphere.datasource.slave1.driver-class-
name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.slave1.jdbc-
url=jdbc:mysql://localhost:3306/xt?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC
spring.shardingsphere.datasource.slave1.username=root
spring.shardingsphere.datasource.slave1.password=root

# 写数据源名称
spring.shardingsphere.rules.readwrite-splitting.data-sources.ms.write-
data-source-name=master
# 读数据源名称，多个从数据源用逗号分隔
spring.shardingsphere.rules.readwrite-splitting.data-sources.ms.read-
data-source-names=slave0,slave1
# 负载均衡算法名称
spring.shardingsphere.rules.readwrite-splitting.data-sources.ms.load-
balancer-name=round-robin

## 负载均衡算法配置
spring.shardingsphere.rules.readwrite-splitting.load-balancers.round-
robin.type=ROUND_ROBIN
## 负载均衡算法属性配置
spring.shardingsphere.rules.readwrite-splitting.load-balancers.round-
robin.props.workId=1
#打印sql
spring.shardingsphere.props.sql-show=true
```

测试，观察控制台日志中，Actual SQL 是走的哪个数据库，select语句会走slave，insert，update等语句会走master，上述负载均衡算法用的是轮询

# 2. 科目

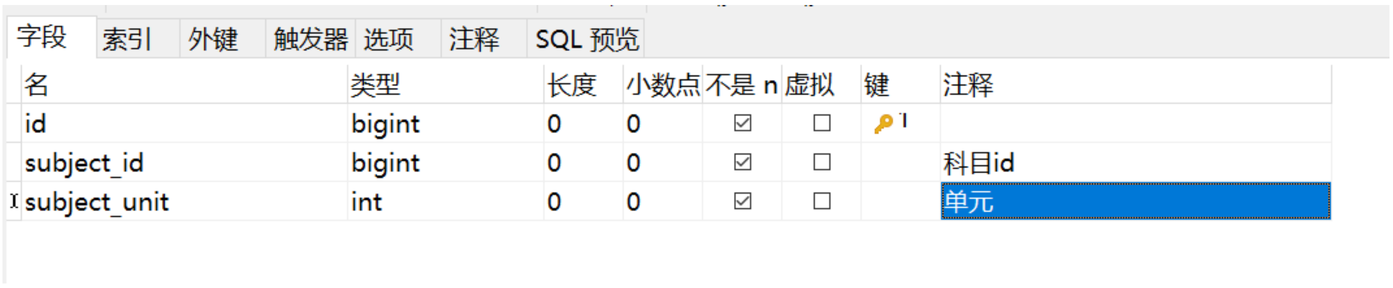
> *习题必须有所属的科目，比如语文，历史，道德，英语，政治等*

## 2.1 数据库设计



| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| id | bigint | 0 | 0 | ☑ | ☐ | 1 | |
| subject_name | varchar | 255 | 0 | ☑ | ☐ | | 科目名称 |
| subject_grade | varchar | 45 | 0 | ☑ | ☐ | | 年级 |
| subject_term | varchar | 45 | 0 | ☑ | ☐ | | 学期（上，下) |
| status | tinyint | 0 | 0 | ☑ | ☐ | | 状态 0 可用 1 不可用 |

每个科目都有不同的单元，每个单元的题目也不一样，练习的时候需要针对单元进行练习，也有可能针对所有单元练习。



| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| id | bigint | 0 | 0 | ☑ | ☐ | 🔑1 | |
| subject_id | bigint | 0 | 0 | ☑ | ☐ | | 科目id |
| subject_unit | int | 0 | 0 | ☑ | ☐ | | 单元 |

```java
package com.mszlu.xt.pojo;

import lombok.Data;

import java.util.List;

/**
 * @author Jarno
 */
@Data
public class Subject {
    private Long id;
    private String subjectName;
    private String subjectGrade;
    private String subjectTerm;
    private Integer status;
}
```

```java
package com.mszlu.xt.pojo;

import lombok.Data;

/**
 * @author Jarno
 */
@Data
public class SubjectUnit {
    private Long id;
    private Long subjectId;
    private Integer subjectUnit;
}
```

## 2.2 问题

> *做到这，我们发现一个问题，不管是新闻还是科目，都是需要管理台进行管理的，也就是说web模块和admin模块都需要，如果每次都采用直接copy实体类的形式，无疑造成了重复性的代码*

新建mszlu-xt-common-pojo模块，引入lombok包，将News和Subject，SubjectUnit这些实体类放入，作为公共模块，web和admin都引入即可。

由于pojo模块中，是有用到mybatis-plus的相关注解，将之引入：

```
        <dependency>
                <groupId>com.baomidou</groupId>
                <artifactId>mybatis-plus-annotation</artifactId>
                <version>3.4.1</version>
                <scope>compile</scope>
            </dependency>
```

修改web模块和admin模块的代码，使之正常运行。

## 2.3 科目管理

将subject.html拷贝到pages目录下

### 2.3.1 分页查询

```
    //分页查询
                findPage() {
                    const params =
{"page":this.pagination.currentPage,"pageSize":20};
                    
 axios.post("/lzadmin/subject/findPage",params).then((res)=>{
                        if (res.data.success){
                            this.dataList = res.data.result.list;
                            this.pagination.total =
res.data.result.total;
                        }
                    });
                },
```

参数：

```java
package com.mszlu.xt.admin.model.param;

import lombok.Data;

import java.util.List;

/**
 * @author Jarno
 */
@Data
public class SubjectParam {
    private Long id;
    private String subjectName;
    private String subjectGrade;
    private String subjectTerm;
    private List<Integer> subjectUnits;

    private Integer status;

    private int page = 1;
    private int pageSize = 20;
}
```

controller代码:

```java
package com.mszlu.xt.admin.controller;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.admin.model.param.SubjectParam;
import com.mszlu.xt.admin.service.SubjectService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author Jarno
 */
@RestController
@RequestMapping("subject")
public class AdminSubjectController {
    @Autowired
    private SubjectService subjectService;


    @RequestMapping(value = "findPage")
    public CallResult findPage(@RequestBody SubjectParam subjectParam){
        return  subjectService.findSubjectList(subjectParam);
    }
}
```

Service代码:

```
package com.mszlu.xt.admin.service;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.admin.model.param.SubjectParam;

/**
 * @author Jarno
 */
public interface SubjectService {

    CallResult findSubjectList(SubjectParam subjectParam);

}
```

```
package com.mszlu.xt.admin.service.impl;

import com.mszlu.common.model.CallResult;
import com.mszlu.common.model.ListModel;
import com.mszlu.common.service.AbstractTemplateAction;
import com.mszlu.xt.admin.domain.SubjectDomain;
import com.mszlu.xt.admin.domain.repository.SubjectDomainRepository;
import com.mszlu.xt.admin.model.param.SubjectParam;
import com.mszlu.xt.admin.service.SubjectService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cglib.beans.BeanCopier;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.*;
```

```java
/**
 * @author Jarno
 */
@Service
@Transactional
public class SubjectServiceImpl extends AbstractService implements
SubjectService {

    @Autowired
    private SubjectDomainRepository subjectDomainRepository;
    @Override
    public CallResult findSubjectList(SubjectParam subjectParam) {
        SubjectDomain subjectDomain =
this.subjectDomainRepository.createDomain(subjectParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<ListModel>() {
            @Override
            public CallResult<ListModel> doAction() {
                return subjectDomain.findSubjectList();
            }
        });
    }


}
```

Domain代码：

```java
package com.mszlu.xt.admin.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mszlu.xt.admin.dao.SubjectMapper;
import com.mszlu.xt.admin.domain.SubjectDomain;
import com.mszlu.xt.admin.model.param.SubjectParam;
import com.mszlu.xt.pojo.Subject;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

/**
 * @author Jarno
 */
@Component
public class SubjectDomainRepository {

    @Resource
    private SubjectMapper subjectMapper;


    public SubjectDomain createDomain(SubjectParam subjectParam) {
        return new SubjectDomain(subjectParam,this);
    }

    public Page<Subject> findSubjectList(Integer currentPage, Integer
pageSize) {
        Page<Subject> page = new Page<>(currentPage,pageSize);
        return subjectMapper.selectPage(page,new LambdaQueryWrapper<>
());
    }

}
```

```java
package com.mszlu.xt.admin.domain;

import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mszlu.common.model.CallResult;
import com.mszlu.common.model.ListModel;
import com.mszlu.xt.admin.domain.repository.SubjectDomainRepository;
import com.mszlu.xt.admin.model.param.SubjectParam;
import com.mszlu.xt.pojo.Subject;
import java.util.List;

public class SubjectDomain {

    private SubjectParam subjectParam;
    private SubjectDomainRepository subjectDomainRepository;


    public SubjectDomain(SubjectParam subjectParam,
SubjectDomainRepository subjectDomainRepository){
        this.subjectParam = subjectParam;
        this.subjectDomainRepository = subjectDomainRepository;
    }

    public CallResult<ListModel> findSubjectList() {
        Page<Subject> subjectList = this.subjectDomainRepository
                .findSubjectList(
                        this.subjectParam.getPage(),
                        this.subjectParam.getPageSize()
                );
        ListModel listModel = new ListModel();
        listModel.setTotal((int) subjectList.getTotal());
        List<Subject> result = subjectList.getRecords();
        for (Subject subject : result) {
            subject.setSubjectName(subject.getSubjectName() +" " +
subject.getSubjectGrade() + " " +subject.getSubjectTerm());
        }
        listModel.setList(result);
        return CallResult.success(listModel);
    }
}
```

Mapper:

```
package com.mszlu.xt.admin.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.Subject;

public interface SubjectMapper extends BaseMapper<Subject> {
}
```

### 2.3.2 新增科目

**注意将数据库id字段，设为默认自增**

```
mybatis-plus.global-config.db-config.id-type=auto
```

```
   //添加
                handleAdd () {
                    this.formData.subjectUnits = this.subjectUnits;

 axios.post("/lzadmin/subject/saveSubject",this.formData).then((res)=>{
                        if (res.data.success){
                            this.$message({
                                message: '恭喜你，添加成功',
                                type: 'success'
                            });
                            this.dialogFormVisible = false;
                            this.findPage();
                        }
                    });
                }
```

Controller:

```
   @PostMapping(value = "saveSubject")
    public CallResult saveSubject(@RequestBody SubjectParam
subjectParam){
        return subjectService.saveSubject(subjectParam);
    }
```

Service:

```
   CallResult saveSubject(SubjectParam subjectParam);
```

```
package com.mszlu.xt.admin.service.impl;

import com.mszlu.xt.admin.domain.SubjectDomain;
import com.mszlu.xt.admin.domain.repository.SubjectDomainRepository;
import com.mszlu.xt.admin.params.SubjectParam;
import com.mszlu.xt.admin.service.SubjectService;
import com.mszlu.xt.common.model.CallResult;
import com.mszlu.xt.common.service.AbstractTemplateAction;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class SubjectServiceImpl extends AbstractService implements
SubjectService {

    @Autowired
    private SubjectDomainRepository subjectDomainRepository;

    @Override
    public CallResult findSubjectList(SubjectParam subjectParam) {
        SubjectDomain subjectDomain =
this.subjectDomainRepository.createDomain(subjectParam);

        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return subjectDomain.findSubjectList();
            }
        });
    }

    @Override
    @Transactional
    public CallResult saveSubject(SubjectParam subjectParam) {
        SubjectDomain subjectDomain =
this.subjectDomainRepository.createDomain(subjectParam);

        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {

            @Override
            public CallResult<Object> checkParam() {
                return subjectDomain.checkSaveSubjectParam();
            }

            @Override
            public CallResult<Object> checkBiz() {
                return subjectDomain.checkSaveSubjectBiz();
            }

            @Override
            public CallResult<Object> doAction() {
                return subjectDomain.saveSubject();
            }
        });
    }
}
```

Domain:

```
package com.mszlu.xt.admin.model;
```

```java
import lombok.Data;

/**
 * @author Jarno
 */
@Data
public class SubjectModel {
    private Long id;
    private String subjectName;
    private String subjectGrade;
    private String subjectTerm;
    private Integer status;

    public void fillSubjectName() {
        this.subjectName = this.subjectName + "-" +subjectGrade + "-" +
subjectTerm;
    }
}
```

```java
package com.mszlu.common.enums;

import java.util.HashMap;
import java.util.Map;

/**
 * @author Jarno
 */
public enum StatusEnum {
    /**
     * look name
     */
    NORMAL(0,"正常"),
    DELETE(1,"删除");

    private static final Map<Integer, StatusEnum> CODE_MAP = new
HashMap<>(3);

    static{
        for(StatusEnum statusEnum: values()){
            CODE_MAP.put(statusEnum.getCode(), statusEnum);
        }
    }

    /**
     * 根据code获取枚举值
     * @param code
     * @return
     */
    public static StatusEnum valueOfCode(int code){
        return CODE_MAP.get(code);
    }

    private int code;
    private String msg;

    StatusEnum(int code, String msg) {
        this.code = code;
        this.msg = msg;
    }

    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }
}
```

```java
package com.mszlu.xt.admin.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mszlu.xt.admin.dao.SubjectMapper;
import com.mszlu.xt.admin.dao.SubjectUnitMapper;
import com.mszlu.xt.admin.domain.SubjectDomain;
import com.mszlu.xt.admin.model.SubjectModel;
import com.mszlu.xt.admin.params.SubjectParam;
import com.mszlu.xt.pojo.Subject;
import com.mszlu.xt.pojo.SubjectUnit;
import org.apache.commons.lang3.StringUtils;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class SubjectDomainRepository {

    @Resource
    private SubjectMapper subjectMapper;
    @Resource
    private SubjectUnitMapper subjectUnitMapper;

    public SubjectDomain createDomain(SubjectParam subjectParam) {
        return new SubjectDomain(this,subjectParam);
    }

    public Page<Subject> findSubjectListPage(int currentPage, int
pageSize, String queryString) {
        Page<Subject> page = new Page<>(currentPage,pageSize);
        LambdaQueryWrapper<Subject> queryWrapper = new
LambdaQueryWrapper<>();
        if (StringUtils.isNotBlank(queryString)){
            queryWrapper.like(Subject::getSubjectName,queryString);
        }
        return subjectMapper.selectPage(page, queryWrapper);
    }

    public Subject findSubjectByCondition(String subjectName, String
subjectGrade, String subjectTerm) {

        LambdaQueryWrapper<Subject> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(Subject::getSubjectName,subjectName)
                    .eq(Subject::getSubjectGrade,subjectGrade)
                    .eq(Subject::getSubjectTerm,subjectTerm);
        queryWrapper.last("limit 1");
        return subjectMapper.selectOne(queryWrapper);
    }

    public void save(Subject subject) {
        this.subjectMapper.insert(subject);
    }

    public void saveSubjectUnit(Long subjectId, Integer subjectUnit) {
        SubjectUnit subjectUnit1 = new SubjectUnit();
        subjectUnit1.setSubjectId(subjectId);
        subjectUnit1.setSubjectUnit(subjectUnit);
        subjectUnitMapper.insert(subjectUnit1);
    }
}
```

```java
package com.mszlu.xt.admin.domain;

import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mszlu.xt.admin.domain.repository.SubjectDomainRepository;
import com.mszlu.xt.admin.model.SubjectModel;
import com.mszlu.xt.admin.params.SubjectParam;
import com.mszlu.xt.common.enums.Status;
import com.mszlu.xt.common.model.BusinessCodeEnum;
import com.mszlu.xt.common.model.CallResult;
import com.mszlu.xt.common.model.ListPageModel;
import com.mszlu.xt.pojo.Subject;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.BeanUtils;

import java.util.ArrayList;
import java.util.List;

public class SubjectDomain {

    private SubjectDomainRepository subjectDomainRepository;

    private SubjectParam subjectParam;

    public SubjectDomain(SubjectDomainRepository
subjectDomainRepository, SubjectParam subjectParam) {
        this.subjectDomainRepository = subjectDomainRepository;
        this.subjectParam = subjectParam;
    }
    public List<SubjectModel> copyList(List<Subject> subjectList){

        List<SubjectModel> subjectModels = new ArrayList<>();
        for (Subject subject : subjectList) {
            SubjectModel target = new SubjectModel();
            BeanUtils.copyProperties(subject, target);
            subjectModels.add(target);
        }
        return subjectModels;
    }

    public CallResult<Object> findSubjectList() {
        int currentPage = this.subjectParam.getCurrentPage();
        int pageSize = this.subjectParam.getPageSize();
        String queryString = this.subjectParam.getQueryString();
        Page<Subject> subjectPage =
subjectDomainRepository.findSubjectListPage(currentPage,pageSize,querySt
ring);

        ListPageModel<SubjectModel> listPageModel = new ListPageModel<>
();
        List<Subject> records = subjectPage.getRecords();

        List<SubjectModel> list = copyList(records);
        list.forEach(SubjectModel::fillSubjectName);
        listPageModel.setList(list);
        //total代表是总条目数
        listPageModel.setSize(subjectPage.getTotal());
        return CallResult.success(listPageModel);
    }

    public CallResult<Object> saveSubject() {
        //学科添加肯定不能重复
        Subject subject = new Subject();
        BeanUtils.copyProperties(this.subjectParam,subject);
        subject.setStatus(Status.NORMAL.getCode());
        this.subjectDomainRepository.save(subject);
        List<Integer> subjectUnits =
this.subjectParam.getSubjectUnits();
        for (Integer subjectUnit : subjectUnits) {

 this.subjectDomainRepository.saveSubjectUnit(subject.getId(),subjectUni
t);
        }
        return CallResult.success();
    }

    public CallResult<Object> checkSaveSubjectBiz() {
        //判断是否重复
        String subjectName = this.subjectParam.getSubjectName();
        String subjectGrade = this.subjectParam.getSubjectGrade();
        String subjectTerm = this.subjectParam.getSubjectTerm();
        Subject subject =
this.subjectDomainRepository.findSubjectByCondition(subjectName,subjectG
rade,subjectTerm);
        if (subject != null){
            return
CallResult.fail(BusinessCodeEnum.CHECK_BIZ_NO_RESULT.getCode(),"不能重复进
行添加");
        }
        return CallResult.success();
    }

    public CallResult<Object> checkSaveSubjectParam() {
        String subjectName = this.subjectParam.getSubjectName();
        String subjectGrade = this.subjectParam.getSubjectGrade();
        String subjectTerm = this.subjectParam.getSubjectTerm();
        if (StringUtils.isBlank(subjectName)
            || StringUtils.isBlank(subjectGrade)
            || StringUtils.isBlank(subjectTerm)){
            return
CallResult.fail(BusinessCodeEnum.CHECK_PARAM_NO_RESULT.getCode(),"参数不能
为空");
        }
        return CallResult.success();
    }
}
```

Mapper:

```
package com.mszlu.xt.admin.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.Subject;
import com.mszlu.xt.pojo.SubjectUnit;

public interface SubjectUnitMapper extends BaseMapper<SubjectUnit> {
}
```

### 2.3.3 编辑科目

1. 回显数据

```
@PostMapping(value = "findSubjectById")
    public CallResult findSubjectById(@RequestBody SubjectParam
subjectParam){
        CallResult callResult =
subjectService.findSubjectById(subjectParam);
        return callResult;
    }
```

```
   @Override
    public CallResult findSubjectById(SubjectParam subjectParam) {
        SubjectDomain subjectDomain =
this.subjectDomainRepository.createDomain(subjectParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction() {
            @Override
            public CallResult doAction() {
                return subjectDomain.findSubjectById();
            }
        });
    }
    public CallResult findSubjectById() {
        Long id = this.subjectParam.getId();
        Subject subject =
this.subjectDomainRepository.findSubjectById(id);
        SubjectModel target = new SubjectModel();
        BeanUtils.copyProperties(subject, target);
        List<Integer> subjectUnits =
this.subjectDomainRepository.findSubjectUnit(id);
        target.setSubjectUnits(subjectUnits);
        return CallResult.success(target);
    }
```

```
package com.mszlu.xt.admin.model;

import lombok.Data;

import java.util.List;

/**
 * @author Jarno
 */
@Data
public class SubjectModel {
    private Long id;
    private String subjectName;
    private String subjectGrade;
    private String subjectTerm;
    private Integer status;
    private List<Integer> subjectUnits;
}
```

```
 public Subject findSubjectById(Long id) {
        return subjectMapper.selectById(id);
    }

public List<Integer> findSubjectUnit(Long id) {
        LambdaQueryWrapper<SubjectUnit> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(SubjectUnit::getSubjectId,id);
        return
subjectUnitMapper.selectList(queryWrapper).stream().map(SubjectUnit
::getSubjectUnit).collect(Collectors.toList());
    }
```

2. 编辑

```
@RequestMapping(value = "updateSubject")
    public CallResult updateSubject(@RequestBody SubjectParam
subjectParam){
        CallResult callResult =
subjectService.updateSubject(subjectParam);
        return callResult;
    }
```

```
@Override
    public CallResult updateSubject(SubjectParam subjectParam) {
        SubjectDomain subjectDomain =
this.subjectDomainRepository.createDomain(subjectParam);

        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {

            @Override
            public CallResult<Object> checkParam() {
                return subjectDomain.checkSaveSubjectParam();
            }

            @Override
            public CallResult<Object> doAction() {
                return subjectDomain.updateSubject();
            }
        });
    }
```

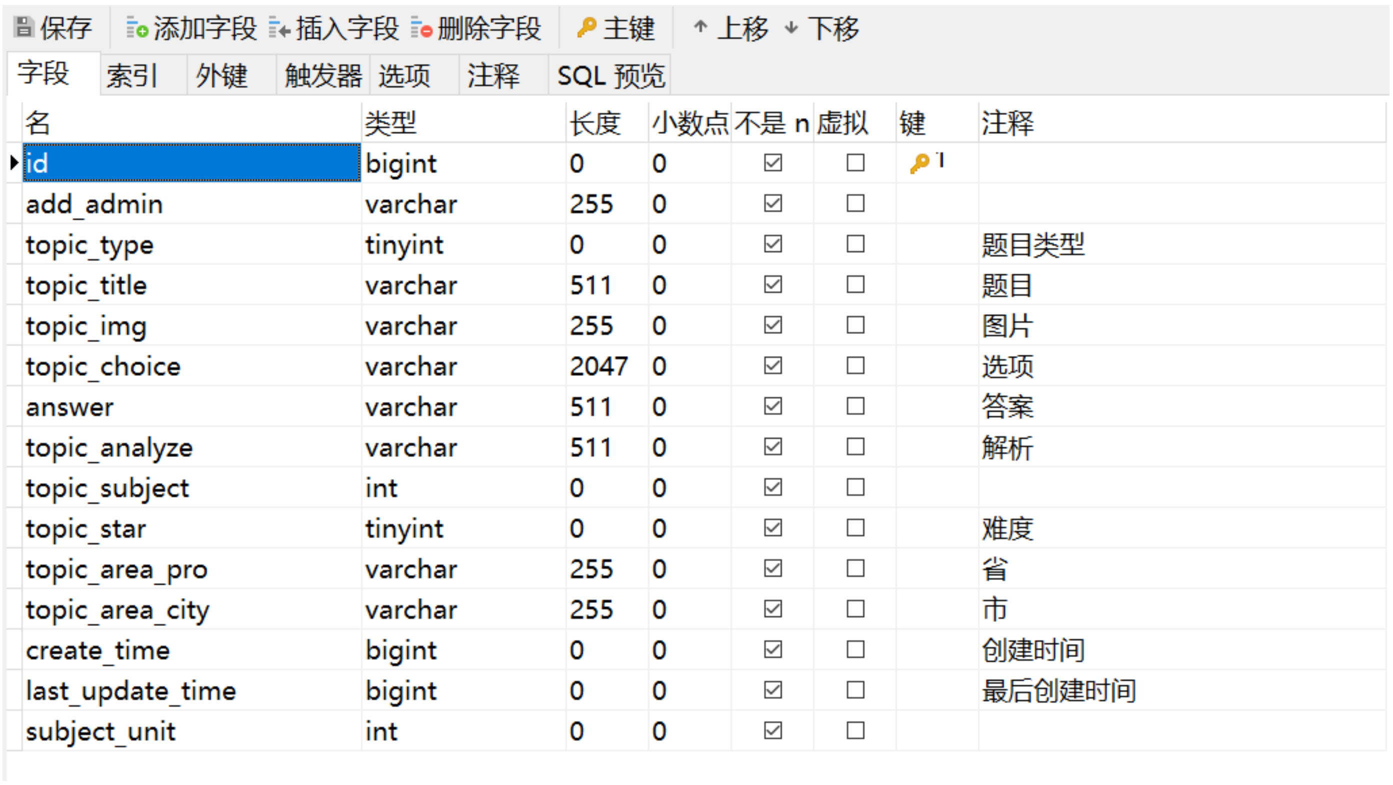
```
public CallResult<Object> updateSubject() {
        /**
         * 1. 更新 subject表
         * 2. 更新 单元表，先删除原有的关联关系 ，在进行更新
         */
        List<Integer> subjectUnits =
this.subjectParam.getSubjectUnits();
        Subject subject = new Subject();
        BeanUtils.copyProperties(this.subjectParam,subject);
        //如果想要判断重复 如何判断
        String subjectName = this.subjectParam.getSubjectName();
        String subjectGrade = this.subjectParam.getSubjectGrade();
        String subjectTerm = this.subjectParam.getSubjectTerm();
        Subject newSubject =
this.subjectDomainRepository.findSubjectByCondition(subjectName,sub
jectGrade,subjectTerm);
        if (newSubject != null &&
!newSubject.getId().equals(subject.getId())){
            return
CallResult.fail(BusinessCodeEnum.CHECK_BIZ_NO_RESULT.getCode(),"不能
添加重复的数据");
        }
        //TODO 差一个逻辑，学科和课程是有关联的，如果课程如果使用了学科，是不
能将学科进行删除的

        this.subjectDomainRepository.update(subject);

        //单元表 先删 后新增

 this.subjectDomainRepository.deleteUnitBySubjectId(subject.getId()
);
        for (Integer subjectUnit : subjectUnits) {

 this.subjectDomainRepository.saveSubjectUnit(subject.getId(),subje
ctUnit);
        }
        return CallResult.success();
    }
```

```
public void update(Subject subject) {
        this.subjectMapper.updateById(subject);
    }

    public void deleteUnitBySubjectId(Long subjectId) {
        LambdaQueryWrapper<SubjectUnit> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(SubjectUnit::getSubjectId,subjectId);
        this.subjectUnitMapper.delete(queryWrapper);
    }
```

# 3. 习题

> *习题的难点：*
>
> 1. *包含多种题型，填空题，多选，单选，判断，问答题*
> 2. *题目中需要包含图片，答案中也需要包含图片*

## 3.1 数据库设计



| 名 | 类型 | 长度 | 小数点 | 不是 n | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| id | bigint | 0 | 0 | ☑ | ☐ | 1 | |
| add_admin | varchar | 255 | 0 | ☑ | ☐ | | |
| topic_type | tinyint | 0 | 0 | ☑ | ☐ | | 题目类型 |
| topic_title | varchar | 511 | 0 | ☑ | ☐ | | 题目 |
| topic_img | varchar | 255 | 0 | ☑ | ☐ | | 图片 |
| topic_choice | varchar | 2047 | 0 | ☑ | ☐ | | 选项 |
| answer | varchar | 511 | 0 | ☑ | ☐ | | 答案 |
| topic_analyze | varchar | 511 | 0 | ☑ | ☐ | | 解析 |
| topic_subject | int | 0 | 0 | ☑ | ☐ | | |
| topic_star | tinyint | 0 | 0 | ☑ | ☐ | | 难度 |
| topic_area_pro | varchar | 255 | 0 | ☑ | ☐ | | 省 |
| topic_area_city | varchar | 255 | 0 | ☑ | ☐ | | 市 |
| create_time | bigint | 0 | 0 | ☑ | ☐ | | 创建时间 |
| last_update_time | bigint | 0 | 0 | ☑ | ☐ | | 最后创建时间 |
| subject_unit | int | 0 | 0 | ☑ | ☐ | | |

习题考虑到随着业务规模的扩大，可能是有过千万的数据，但需要很长时间，所以还是采用单表设计，但是id使用分布式id

```java
package com.mszlu.xt.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import lombok.Data;


/**
 * @author Jarno
 */
@Data
public class Topic {

    @TableId(type = IdType.ASSIGN_ID)
    private Long id;

    private String addAdmin;

    private String topicTitle;

    private Integer topicType;

    private String topicImg;

    private String topicChoice;

    private Integer topicStar;

    private String topicAreaPro;

    private String topicAreaCity;

    private String topicAnswer;

    private String topicAnalyze;

    private Long topicSubject;

    private Long createTime;

    private Long lastUpdateTime;

    private Integer subjectUnit;

}
```

> *观察topic表，可以发现topic_title和topic_img，题目中可以用①②这种代替图片位置，图片上标识①②即可，图片存储路径，用英文逗号分隔，代表多个*

**设计数据库，更多的是一种规则的说明，只要大家遵守规则即可**

那么选项呢?

> *topicChoice: 这个代表选项，因为选项多种多样，所以这里使用json格式的字符串进行存储，但是json的格式需要固定*

```java
package com.mszlu.common.enums;

import java.util.HashMap;
import java.util.Map;

/**
 * @author Jarno
 */
public enum TopicType {
    /**
     * look name
     */
    FILL_BLANK(0,"fill blank"),
    RADIO(1,"radio"),
    MUL_CHOICE(2,"mul choice"),
    QA(3,"question or answer"),
    JUDGE(4,"judge");

    private static final Map<Integer, TopicType> CODE_MAP = new
HashMap<>(3);

    static{
        for(TopicType topicType: values()){
            CODE_MAP.put(topicType.getCode(), topicType);
        }
    }

    /**
     * 根据code获取枚举值
     * @param code
     * @return
     */
    public static TopicType valueOfCode(int code){
        return CODE_MAP.get(code);
    }

    private int code;
    private String msg;

    TopicType(int code, String msg) {
        this.code = code;
        this.msg = msg;
    }

    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }
}
```

填空选项:

```java
package com.mszlu.common.model.topic;

import lombok.Data;

/**
 * @author Jarno
 */
@Data
public class FillBlankChoice {
    private int id;
    private String content;
}
```

判断题和问答题：answer字段，对于判断题：0 对 1 错

单选和多选：

**选项内容由两部分组成，文字+图片url**

Map的key 为 A，B，C，D，E，F，G，H，最多也就这么多选项了，可以写死

value部分，根据图片url，及http进行分割，约定了图片包在[]内。

List<Map<String,List>>

```java
package com.mszlu.common.model.topic;

import lombok.Data;
import org.apache.commons.lang3.StringUtils;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

@Data
public class ContentAndImage {
    private String content;
    private List<String> imageList;

    public static ContentAndImage deal(String str){
        if (StringUtils.isEmpty(str)){
            return null;
        }
        List<String> imageList = new ArrayList<>();
        String content = str;
        if (str.contains("http")){
            if (str.contains("[http")){
                //多个图片
                Pattern pt = Pattern.compile("http?://[-A-Za-z0-9+&@#/%?
=~_|!:,.;]+[-A-Za-z0-9+&@#/%=~_|]");
                Matcher mt = pt.matcher(str);
                if (mt.find()) {
                    String imagesStr = mt.group();
                    String[] images = imagesStr.split(",");
                    imageList.addAll(Arrays.asList(images));
                }
            }else {
                Pattern pt = Pattern.compile("http?://[-A-Za-z0-9+&@#/%?
=~_|!:,.;]+[-A-Za-z0-9+&@#/%=~_|]");
                Matcher mt = pt.matcher(str);
                if (mt.find()) {
                    imageList.add(mt.group());
                }
                content = str.split("http")[0];
            }
        }
        ContentAndImage contentAndImageModel = new ContentAndImage();
        contentAndImageModel.setContent(content);
        contentAndImageModel.setImageList(imageList);
        return contentAndImageModel;
    }

    public static void main(String[] args) {
        Pattern pt = Pattern.compile("http?://[-A-Za-z0-9+&@#/%?
=~_|!:,.;]+[-A-Za-z0-9+&@#/%=~_|]");
        Matcher mt = pt.matcher("河姆渡黑陶钵
[http://static.lzxtedu.com/7a1/11a.png,http://static.lzxtedu.com/7a1/11b
.png]");
        if (mt.find()) {
            System.out.println(mt.group());
        }
    }
}
```

## 3.2 分页查询

将资料中topic.html拷贝到pages目录下

Controller:

```java
package com.mszlu.xt.admin.controller;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.admin.model.param.TopicParam;
import com.mszlu.xt.admin.service.TopicService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

/**
 * @author Jarno
 */
@RestController
@RequestMapping("topic")
@Slf4j
public class AdminTopicController {

    @Autowired
    private TopicService topicService;


    @RequestMapping(value = "findPage")
    public CallResult findPage(@RequestBody TopicParam topicParam){
        return topicService.findTopicList(topicParam);
    }

}
```

参数:

```java
package com.mszlu.xt.admin.model.param;

import lombok.Data;

import java.util.List;

/**
```

```
 * @author Jarno
 */
@Data
public class TopicParam {
    private Long id;
    private Integer topicType;
    private String topicTitle;
    private String topicImg;
    private String answer;
    private String topicAnalyze;
    private Long topicSubject;
    private Integer topicStar;
    private String topicAreaPro;
    private String topicAreaCity;
    private Integer page = 1;
    private Integer pageSize = 20;

    //subject
    private Long subjectId;

    private Long userId;
    //题目ID
    private Long topicId;
    //单元
    private Integer subjectUnit;
    //单元
    private List<Integer> subjectUnitList;

    private String subjectName;
    private String subjectGrade;
    private String subjectTerm;
}
```

Service:

```
package com.mszlu.xt.admin.service;


import com.mszlu.common.model.CallResult;
import com.mszlu.xt.admin.model.param.TopicParam;

/**
 * @author Jarno
 */
public interface TopicService {

    CallResult findTopicList(TopicParam topicParam);

}
```

```
package com.mszlu.xt.admin.service.impl;

import com.mszlu.common.model.CallResult;
import com.mszlu.common.service.AbstractTemplateAction;
import com.mszlu.xt.admin.domain.TopicDomain;
import com.mszlu.xt.admin.domain.repository.TopicDomainRepository;
import com.mszlu.xt.admin.model.param.TopicParam;
import com.mszlu.xt.admin.service.TopicService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

/**
 * @author Jarno
 */
@Service
@Transactional
public class TopicServiceImpl extends AbstractService implements
TopicService {
    @Autowired
    private TopicDomainRepository topicDomainRepository;
    @Override
    public CallResult findTopicList(TopicParam topicParam) {
        TopicDomain topicDomain =
this.topicDomainRepository.createDomain(topicParam);
        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return topicDomain.findTopicList();
            }
        });
    }
}
```

Domain:

```
package com.mszlu.xt.admin.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mszlu.xt.admin.dao.TopicMapper;
import com.mszlu.xt.admin.domain.SubjectDomain;
import com.mszlu.xt.admin.domain.TopicDomain;
import com.mszlu.xt.admin.model.param.SubjectParam;
import com.mszlu.xt.admin.model.param.TopicParam;
import com.mszlu.xt.pojo.Subject;
import com.mszlu.xt.pojo.Topic;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.util.List;
import java.util.Map;

/**
 * @author Jarno
 */
@Component
public class TopicDomainRepository {

    @Resource
    private TopicMapper topicMapper;
    @Autowired
    private SubjectDomainRepository subjectDomainRepository;

    public TopicDomain createDomain(TopicParam topicParam) {
        return new TopicDomain(topicParam,this);
    }


    public Page<Topic> findPage(int currentPage, int size, String
topicTitle, Long subjectId) {
        Page<Topic> page = new Page<>(currentPage,size);
        LambdaQueryWrapper<Topic> queryWrapper = new
LambdaQueryWrapper<>();
        if (StringUtils.isNotBlank(topicTitle)) {
            queryWrapper.like(Topic::getTopicTitle, topicTitle);
        }
        if (subjectId != null){
            queryWrapper.eq(Topic::getSubject,subjectId);
        }
        return topicMapper.selectPage(page, queryWrapper);
    }

    public SubjectDomain createSubjectDomain(SubjectParam subjectParam)
{
        return subjectDomainRepository.createDomain(subjectParam);
    }
}
```

```
package com.mszlu.xt.admin.domain;

import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.fasterxml.jackson.core.type.TypeReference;
import com.mszlu.common.enums.TopicType;
import com.mszlu.common.model.CallResult;
import com.mszlu.common.model.ListModel;
import com.mszlu.xt.admin.domain.repository.TopicDomainRepository;
import com.mszlu.xt.admin.model.TopicModel;
import com.mszlu.xt.admin.model.param.TopicParam;
import com.mszlu.xt.pojo.Subject;
import com.mszlu.xt.pojo.Topic;
import org.apache.commons.beanutils.BeanUtilsBean;
import org.apache.commons.lang3.StringUtils;
import org.joda.time.DateTime;
import org.springframework.beans.BeanUtils;
import org.springframework.cglib.beans.BeanCopier;

import java.util.*;

/**
 * @author Jarno
 */
public class TopicDomain {
    private TopicParam topicParam;
    private TopicDomainRepository topicDomainRepository;
    public TopicDomain(TopicParam topicParam, TopicDomainRepository 
topicDomainRepository){
        this.topicParam = topicParam;
        this.topicDomainRepository = topicDomainRepository;
    }

    private String displayTopicType(Integer topicType) {
        switch (TopicType.valueOfCode(topicType)){
            case FILL_BLANK:
                return "填空题";
            case RADIO:
                return "单选题";
            case QA:
                return "问答题";
            case MUL_CHOICE:
                return "多选题";
            case JUDGE:
                return "判断题";
            default:return "";
        }
    }
    private String displayTopicSubject(List<Subject> subjectList, Long 
subject) {
        for (Subject subject1 : subjectList){
            if (subject1.getId().equals(subject)){
                return subject1.getSubjectName() +" "+ 
subject1.getSubjectGrade() +" "+ subject1.getSubjectTerm();
            }
        }
        return "";
    }
    private TopicModel copy(Topic topic){
        TopicModel topicModel = new TopicModel();
        BeanUtils.copyProperties(topic,topicModel);
        return topicModel;
    }
    private List<TopicModel> copyList(List<Topic> topicList){
        if (topicList == null){
            return null;
        }
        List<Subject> subjectList = 
this.topicDomainRepository.createSubjectDomain(null).allSubjectList();
        List<TopicModel> topicModelList = new ArrayList<>();
        for (Topic topic : topicList){
            TopicModel model = copy(topic);

 model.setTopicTypeStr(displayTopicType(topic.getTopicType()));

 model.setSubjectStr(displayTopicSubject(subjectList,topic.getTopicSubje
ct()));
            model.setCreateTime(new
DateTime(topic.getCreateTime()).toString("yyyy-MM-dd HH:mm:ss"));
            model.setLastUpdateTime(new
DateTime(topic.getLastUpdateTime()).toString("yyyy-MM-dd HH:mm:ss"));
            topicModelList.add(model);
        }
        return topicModelList;
    }


    public CallResult<Object> findTopicList() {
        int page = this.topicParam.getPage();
        int pageSize = this.topicParam.getPageSize();
        String topicTitle = topicParam.getTopicTitle();
        Long subjectId = topicParam.getSubjectId();
        Page<Topic> topicList =
this.topicDomainRepository.findPage(page,pageSize,topicTitle,subjectId);
        List<TopicModel> topicModelList =
copyList(topicList.getRecords());
        ListPageModel<TopicModel> listModel = new ListPageModel<>();
        listModel.setList(topicModelList);
        listModel.setSize(topicList.getTotal());
        return CallResult.success(listModel);
    }

}
```

```
//SubjectDomainRepository
public List<Subject> findAll() {
        return this.subjectMapper.selectList(new LambdaQueryWrapper<>
());
    }
```

Model:

```java
package com.mszlu.xt.admin.model;

import lombok.Data;

import java.util.List;
import java.util.Map;

/**
 * @author Jarno
 */
@Data
public class TopicModel {

    private Long id;

    private String addAdmin;

    private String topicTitle;

    private String topicTypeStr;

    private Integer topicType;

    private String topicImg;

    private String topicChoice;

    private Integer topicStar;

    private String topicAreaPro;

    private String topicAreaCity;

    private String topicAnswer;

    private String topicAnalyze;

    private String subjectStr;

    private Long subject;

    private String createTime;

    private String lastUpdateTime;
}
```

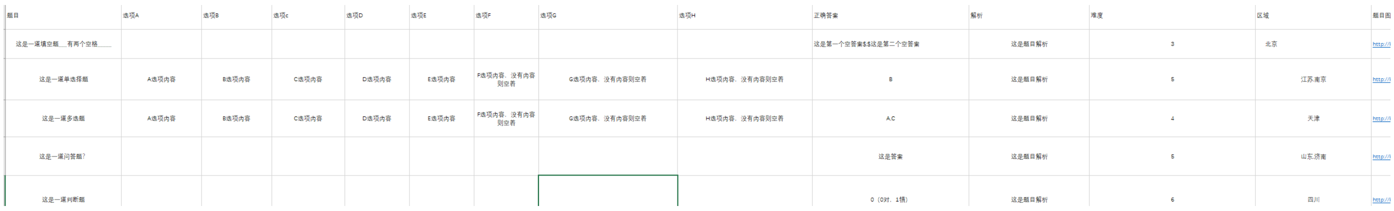
Mapper:

```java
package com.mszlu.xt.admin.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.Topic;

public interface TopicMapper extends BaseMapper<Topic> {
}
```

## 3.3 批量导入

> *习题的添加不可能一道题一道题的去添加，一般是由老师或者录题人员来取做，使用的工具大多是word文档或者是excel文档。*
>
> *word文档格式不固定，但是excel文档，只要规定了模板，那么录题人员只需要copy题目，就可以很便捷的录入大量的习题，这个时候使用系统提供的批量导入功能即可。*

#### 需要设计一个可行的excel模板



| 题目 | 选项A | 选项B | 选项c | 选项D | 选项E | 选项F | 选项G | 选项H | 正确答案 | 解析 | 难度 | 区域 | 题目图 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 这是一道填空题___有两个空格_____ | | | | | | | | | 这是第一个空答案$$这是第二个空答案 | 这是题目解析 | 3 | 北京 | http://i |
| 这是一道单选择题 | A选项内容 | B选项内容 | C选项内容 | D选项内容 | E选项内容 | F选项内容，没有内容则空着 | G选项内容，没有内容则空着 | H选项内容，没有内容则空着 | B | 这是题目解析 | 5 | 江苏.南京 | http://i |
| 这是一道多选题 | A选项内容 | B选项内容 | C选项内容 | D选项内容 | E选项内容 | F选项内容，没有内容则空着 | G选项内容，没有内容则空着 | H选项内容，没有内容则空着 | A,C | 这是题目解析 | 4 | 天津 | http://i |
| 这是一道问答题? | | | | | | | | | 这是答案 | 这是题目解析 | 5 | 山东.济南 | http://i |
| 这是一道判断题 | | | | | | | | | 0（0对，1错） | 这是题目解析 | 6 | 四川 | http://i |

将模板内容填充，在题目管理，上传即可

### 3.3.1 POI工具类

> *excel有几个概念需要明确：*
>
> 1. *工作簿*
> 2. *工作表 Sheet*
> 3. *行 Row*
> 4. *列 Col 单元格*

```xml
  <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
        </dependency>
```

```java
package com.mszlu.common.utils;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.springframework.web.multipart.MultipartFile;

public class POIUtils {
    private final static String xls = "xls";
    private final static String xlsx = "xlsx";
    private final static String DATE_FORMAT = "yyyy/MM/dd";
    /**
     * 读入excel文件，解析后返回
     * @param file
     * @throws IOException
     */
    public static List<String[]> readExcel(MultipartFile file) throws
IOException {
        //检查文件
        checkFile(file);
        //获得Workbook工作薄对象
        Workbook workbook = getWorkBook(file);
        //创建返回对象，把每行中的值作为一个数组，所有行作为一个集合返回
        List<String[]> list = new ArrayList<String[]>();
        if(workbook != null){
            for(int sheetNum = 0;sheetNum <
workbook.getNumberOfSheets();sheetNum++){
                //获得当前sheet工作表
                Sheet sheet = workbook.getSheetAt(sheetNum);
                if(sheet == null){
                    continue;
                }
                //获得当前sheet的开始行
                int firstRowNum  = sheet.getFirstRowNum();
                //获得当前sheet的结束行
                int lastRowNum = sheet.getLastRowNum();
                //循环除了第一行的所有行
                for(int rowNum = firstRowNum+1;rowNum <=
lastRowNum;rowNum++){
                    //获得当前行
                    Row row = sheet.getRow(rowNum);
                    if(row == null){
                        continue;
                    }
                    //获得当前行的开始列
                    int firstCellNum = row.getFirstCellNum();
                    //获得当前行的列数
                    int lastCellNum = row.getPhysicalNumberOfCells();
                    String[] cells = new
String[row.getPhysicalNumberOfCells()];
                    //循环当前行
                    for(int cellNum = firstCellNum; cellNum <
lastCellNum;cellNum++){
                        Cell cell = row.getCell(cellNum);
                        cells[cellNum] = getCellValue(cell);
                    }
                    list.add(cells);
                }
            }
            workbook.close();
        }
        return list;
    }

    //校验文件是否合法
    public static void checkFile(MultipartFile file) throws IOException{
        //判断文件是否存在
        if(null == file){
            throw new FileNotFoundException("文件不存在！");
        }
        //获得文件名
        String fileName = file.getOriginalFilename();
        //判断文件是否是excel文件
        if(!fileName.endsWith(xls) && !fileName.endsWith(xlsx)){
            throw new IOException(fileName + "不是excel文件");
        }
    }
    public static Workbook getWorkBook(MultipartFile file) {
        //获得文件名
        String fileName = file.getOriginalFilename();
        //创建Workbook工作薄对象，表示整个excel
        Workbook workbook = null;
        try {
            //获取excel文件的io流
            InputStream is = file.getInputStream();
            //根据文件后缀名不同(xls和xlsx)获得不同的Workbook实现类对象
            if(fileName.endsWith(xls)){
                //2003
                workbook = new HSSFWorkbook(is);
            }else if(fileName.endsWith(xlsx)){
                //2007
                workbook = new XSSFWorkbook(is);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return workbook;
    }
    public static String getCellValue(Cell cell){
        String cellValue = "";
        if(cell == null){
            return cellValue;
        }
        //如果当前单元格内容为日期类型，需要特殊处理
        String dataFormatString =
cell.getCellStyle().getDataFormatString();
        if(dataFormatString != null &&
dataFormatString.equals("m/d/yy")){
            cellValue = new
SimpleDateFormat(DATE_FORMAT).format(cell.getDateCellValue());
            return cellValue;
        }
```

```
            //把数字当成String来读，避免出现1读成1.0的情况
            if(cell.getCellType() == Cell.CELL_TYPE_NUMERIC){
                cell.setCellType(Cell.CELL_TYPE_STRING);
            }
            //判断数据的类型
            switch (cell.getCellType()){
                case Cell.CELL_TYPE_NUMERIC: //数字
                    cellValue = String.valueOf(cell.getNumericCellValue());
                    break;
                case Cell.CELL_TYPE_STRING: //字符串
                    cellValue = String.valueOf(cell.getStringCellValue());
                    break;
                case Cell.CELL_TYPE_BOOLEAN: //Boolean
                    cellValue = String.valueOf(cell.getBooleanCellValue());
                    break;
                case Cell.CELL_TYPE_FORMULA: //公式
                    cellValue = String.valueOf(cell.getCellFormula());
                    break;
                case Cell.CELL_TYPE_BLANK: //空值
                    cellValue = "";
                    break;
                case Cell.CELL_TYPE_ERROR: //故障
                    cellValue = "非法字符";
                    break;
                default:
                    cellValue = "未知类型";
                    break;
            }
            return cellValue;
        }
    }
```

### 3.3.2 代码

> *逻辑：读取excel，按照模板读取每行对应的数据，构建Topic对象，根据题目进行查询，如果有进行更新，如果没有进行插入*

```
 @RequestMapping("uploadExcel/{subjectId}")
    public CallResult upload(@RequestParam("excelFile") MultipartFile
multipartFile, @PathVariable("subjectId") Long subjectId){
        int updateNum = 0;
        if (subjectId == -1){
            return CallResult.fail(-999,"subjectId 不能为空");
        }
        try {
            List<String[]> list = POIUtils.readExcel(multipartFile);
            for (String[] strs : list){
                if (StringUtils.isEmpty(strs[0])){
                    continue;
                }
                Integer subjectUnit = Integer.parseInt(strs[0]);
                if (StringUtils.isEmpty(strs[1])){
                    continue;
                }
                Integer topicType = Integer.parseInt(strs[1]);

                String topicTitle = strs[2];
                List<Map<String, ContentAndImage>> choiceList = new
ArrayList<>();
                String choiceA = strs[3];
                ContentAndImage a = ContentAndImage.deal(choiceA);
                if (a != null){
                    Map<String,ContentAndImage> aMap = new HashMap<>();
                    aMap.put("A",a);
                    choiceList.add(aMap);
                }
                String choiceB = strs[4];
                ContentAndImage b = ContentAndImage.deal(choiceB);
                if (b != null){
                    Map<String,ContentAndImage> bMap = new HashMap<>();
                    bMap.put("B",b);
                    choiceList.add(bMap);
                }
                String choiceC = strs[5];
                ContentAndImage c = ContentAndImage.deal(choiceC);
                if (c != null){
                    Map<String,ContentAndImage> cMap = new HashMap<>();
                    cMap.put("C",c);
                    choiceList.add(cMap);
                }
                String choiceD = strs[6];
                ContentAndImage d = ContentAndImage.deal(choiceD);
                if (d != null){
                    Map<String,ContentAndImage> dMap = new HashMap<>();
                    dMap.put("D",d);
                    choiceList.add(dMap);
                }
                String choiceE = strs[7];
                ContentAndImage e = ContentAndImage.deal(choiceE);
                if (e != null){
                    Map<String,ContentAndImage> eMap = new HashMap<>();
                    eMap.put("E",e);
                    choiceList.add(eMap);
                }
                String choiceF = "";
                if (strs.length >= 9) {
                    choiceF = strs[8];
                }
                ContentAndImage f = ContentAndImage.deal(choiceF);
                if (f != null){
                    Map<String,ContentAndImage> fMap = new HashMap<>();
                    fMap.put("F",f);
                    choiceList.add(fMap);
                }
                String choiceG = "";
                if (strs.length >= 10) {
                    choiceG = strs[9];
                }
                ContentAndImage g = ContentAndImage.deal(choiceG);
                if (g != null){
                    Map<String,ContentAndImage> gMap = new HashMap<>();
                    gMap.put("G",g);
                    choiceList.add(gMap);
                }
                String choiceH = "";
                if (strs.length >= 11) {
                    choiceH = strs[10];
                }
                ContentAndImage h = ContentAndImage.deal(choiceH);
                if (h != null){
                    Map<String,ContentAndImage> hMap = new HashMap<>();
                    hMap.put("H",h);
                    choiceList.add(hMap);
                }
                String topicAnswer = "";
                if (strs.length >= 12) {
                    topicAnswer = strs[11];
                }
                String topicAnalyze = "";
                if (strs.length >= 13) {
```

```
                    topicAnalyze = strs[12];
                }
                Integer topicStar = 3;
                if (strs.length >= 14) {
                    if (!StringUtils.isEmpty(strs[13])) {
                        topicStar = Integer.parseInt(strs[13]);
                    }
                }
                String topicAreaCity = "";
                if (strs.length >= 15) {
                    topicAreaCity = strs[14];
                }
                String topicImage = null;
                if (strs.length >= 16) {
                    topicImage = strs[15];
                }
                List<String> topicImageList = new ArrayList<>();
                if (!StringUtils.isEmpty(topicImage)) {
                    Pattern pt = Pattern.compile("http?://[-A-Za-z0-
9+&@#/%?=~_|!:,.;]+[-A-Za-z0-9+&@#/%=~_|]");
                    Matcher mt = pt.matcher(topicImage);
                    if (mt.find()) {
                        String imagesStr = mt.group();
                        String[] images = imagesStr.split(",");
                        topicImageList.addAll(Arrays.asList(images));
                    }
                }
                Topic topic = new Topic();
                topic.setTopicTitle(topicTitle);
                topic.setTopicType(topicType);
                topic.setSubjectUnit(subjectUnit);
                topic.setTopicAnalyze(topicAnalyze);
                if (TopicType.FILL_BLANK.getCode() == topicType){
                    String[] strAnswer = topicAnswer.split("\\$;\\$");
                    List<Map<String,Object>> map = new ArrayList<>();
                    for (int i = 1;i<=strAnswer.length;i++){
                        Map<String,Object> m = new HashMap<>();
                        m.put("id",i);
                        m.put("content",strAnswer[i-1]);
                        map.add(m);
                    }
                    topic.setTopicChoice(JSON.toJSONString(map));
                }else{
                    topic.setTopicChoice(JSON.toJSONString(choiceList));
                }
                topic.setTopicAnswer(topicAnswer);

                topic.setCreateTime(System.currentTimeMillis());
                topic.setLastUpdateTime(System.currentTimeMillis());
                topic.setTopicStar(topicStar);
                topic.setTopicAreaCity(topicAreaCity);
                topic.setTopicImg(JSON.toJSONString(topicImageList));
                topic.setAddAdmin("admin");
                topic.setTopicAreaPro("");
                topic.setTopicSubject(subjectId);
                Topic t =
this.topicService.findTopicByTitle(topic.getTopicTitle());
                if (t != null){
                    log.info("update topic:{}",topic.getTopicTitle());
                    topic.setId(t.getId());
                    updateNum++;
                    this.topicService.updateTopic(topic);
                }else{
                    this.topicService.saveTopic(topic);
                }
//
            }
//             System.out.println(JSON.toJSONString(list));
        } catch (IOException e) {
            e.printStackTrace();
            return CallResult.fail();
        }
        log.info("update num:{}",updateNum);
        return CallResult.success();
    }
```

Service:

```
<dependency>
            <groupId>com.mszlu</groupId>
            <artifactId>mszlu-xt-common-pojo</artifactId>
        </dependency>
```

```
 Topic findTopicByTitle(String topicTitle);

    void updateTopic(Topic topic);

    void saveTopic(Topic topic);
```

```
 @Override
    public Topic findTopicByTitle(String topicTitle) {
        TopicDomain topicDomain =
this.topicDomainRepository.createDomain(null);
        CallResult<Topic> topicCallResult =
this.serviceTemplate.executeQuery(new AbstractTemplateAction<Topic>() {
            @Override
            public CallResult<Topic> doAction() {
                return topicDomain.findTopicByTitle(topicTitle);
            }
        });
        if (topicCallResult.isSuccess()){
            return topicCallResult.getResult();
        }
        return null;
    }

    @Override
    public void updateTopic(Topic topic) {
        TopicDomain topicDomain =
this.topicDomainRepository.createDomain(null);
        this.serviceTemplate.execute(new AbstractTemplateAction<Object>
() {
            @Override
            public CallResult<Object> doAction() {
                return topicDomain.updateTopic(topic);
            }
        });
    }
    @Override
    public void saveTopic(Topic topic) {
        TopicDomain topicDomain =
this.topicDomainRepository.createDomain(null);
        this.serviceTemplate.execute(new AbstractTemplateAction<Object>
() {
            @Override
            public CallResult<Object> doAction() {
                return topicDomain.saveTopic(topic);
                }
            });
        }
```

Domain:

```
   public CallResult<Topic> findTopicByTitle(String topicTitle) {
          Topic topic =
  this.topicDomainRepository.findTopicByTitle(topicTitle);
          return CallResult.success(topic);
      }

      public CallResult<Object> updateTopic(Topic topic) {
          this.topicDomainRepository.updateTopicImage(topic);
          return CallResult.success();
      }

      public CallResult<Object> saveTopic(Topic topic) {
          this.topicDomainRepository.saveTopic(topic);
          return CallResult.success();
      }
```

```
   public Topic findTopicByTitle(String topicTitle) {
          LambdaQueryWrapper<Topic> queryWrapper = new
  LambdaQueryWrapper<>();
          queryWrapper.eq(Topic::getTopicTitle,topicTitle);
          queryWrapper.last("limit 1");
          Topic topic = topicMapper.selectOne(queryWrapper);
          return topic;
      }

      public void updateTopicImage(Topic topic) {
          this.topicMapper.updateById(topic);
      }

      public void saveTopic(Topic topic) {
          this.topicMapper.insert(topic);
      }
```

### 3.3.4 所有科目列表

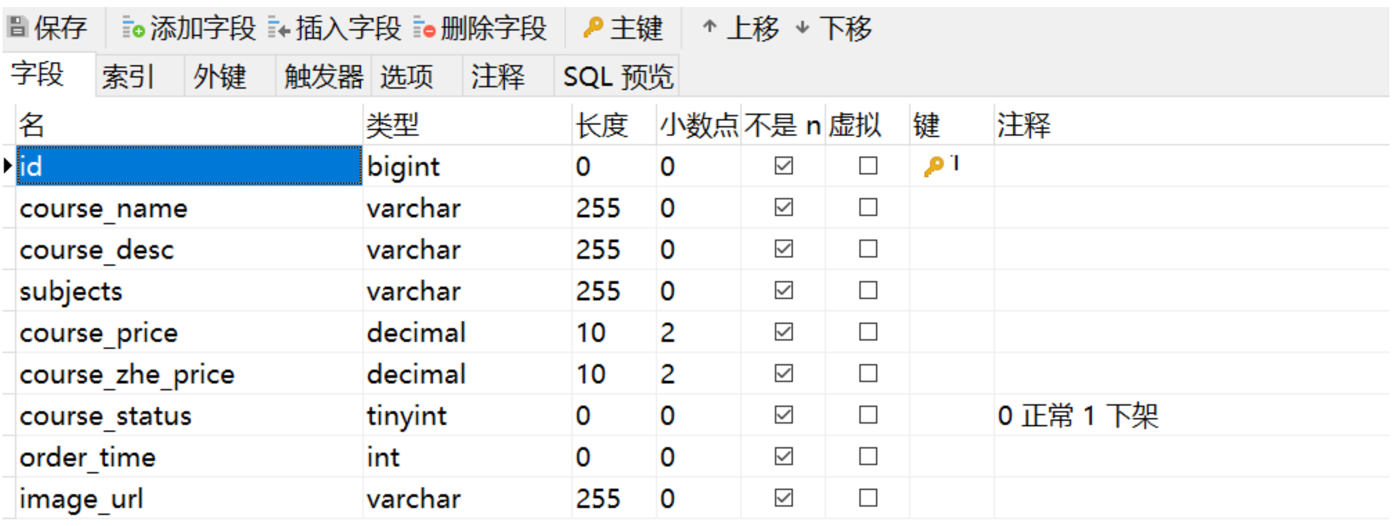
测试导入之前，需要将所有的科目列表查询出来，习题必须依赖科目存在。

```
//前端传递参数的时候，将pageSize设置的大一点即可
@PostMapping(value = "allSubjectList")
    public CallResult allSubjectList(@RequestBody SubjectParam 
subjectParam){
        return subjectService.findSubjectList(subjectParam);
    }
```

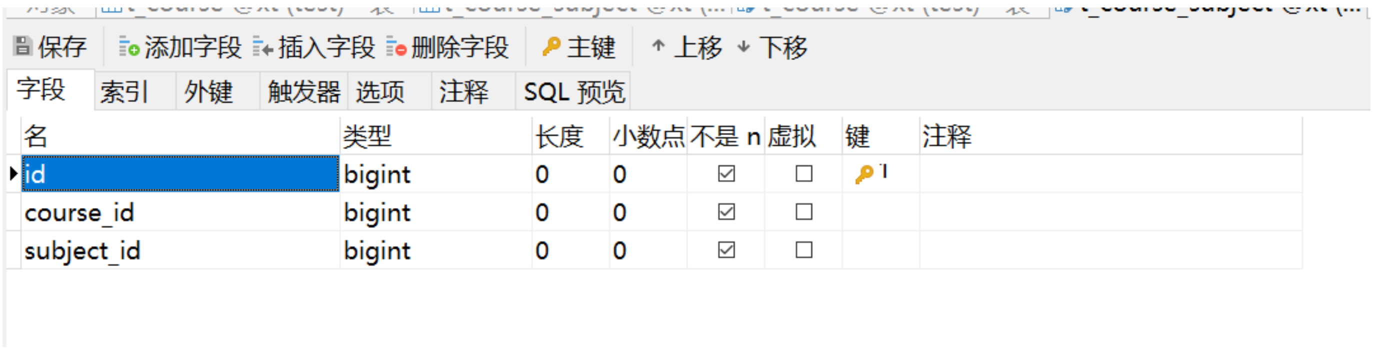
# 4. 课程

> *科目有了，习题有了，接下来就是做课程了，课程包含了多个科目，此科目下的所有的习题都可以进行练习，在我们的系统了，一个课程就是一个商品*

## 4.1 数据库设计



| 名 | 类型 | 长度 | 小数点 | 不是 n | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| id | bigint | 0 | 0 | ☑ | ☐ | 🔑 1 | |
| course_name | varchar | 255 | 0 | ☑ | ☐ | | |
| course_desc | varchar | 255 | 0 | ☑ | ☐ | | |
| subjects | varchar | 255 | 0 | ☑ | ☐ | | |
| course_price | decimal | 10 | 2 | ☑ | ☐ | | |
| course_zhe_price | decimal | 10 | 2 | ☑ | ☐ | | |
| course_status | tinyint | 0 | 0 | ☑ | ☐ | | 0 正常 1 下架 |
| order_time | int | 0 | 0 | ☑ | ☐ | | |
| image_url | varchar | 255 | 0 | ☑ | ☐ | | |



| 名 | 类型 | 长度 | 小数点 | 不是 n | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| id | bigint | 0 | 0 | ☑ | ☐ | 🔑 1 | |
| course_id | bigint | 0 | 0 | ☑ | ☐ | | |
| subject_id | bigint | 0 | 0 | ☑ | ☐ | | |

```java
package com.mszlu.xt.pojo;

import lombok.Data;

import java.math.BigDecimal;

@Data
public class Course {
    private Long id;
    private String courseName;
    private String courseDesc;
    private String subjects;
    private BigDecimal coursePrice;
    private BigDecimal courseZhePrice;
    private Integer orderTime;//天数
    private Integer courseStatus;//正常，下架
    private String imageUrl;

}
```

## 4.1 代码

课程这块 主要就是增删改查，前面有类似的代码。

参数：

```java
package com.mszlu.xt.admin.model.param;

import lombok.Data;

import java.math.BigDecimal;
import java.util.List;

/**
 * @author Jarno
 */
@Data
public class CourseParam {
    private Long userId;
    private Long id;
    private String courseName;
    private String courseDesc;
    private String subjects;
    private BigDecimal coursePrice;
    private BigDecimal courseZhePrice;
    private Integer courseStatus;
    private List<Long> subjectIdList;
    private Integer orderTime;//订购时长
    private int page = 1;
    private int pageSize = 20;
    private String subjectName;
    private String subjectGrade;
    private String subjectTerm;

    private Long courseId;
    private String imageUrl;
}
```

Controller:

```
package com.mszlu.xt.admin.controller;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.admin.model.param.CourseParam;
import com.mszlu.xt.admin.service.CourseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author Jarno
 */
@RestController
@RequestMapping("course")
public class AdminCourseController {

    @Autowired
    private CourseService courseService;

    @RequestMapping(value = "saveCourse")
    public CallResult saveCourse(@RequestBody CourseParam courseParam){
        return courseService.saveCourse(courseParam);
    }

    @RequestMapping(value = "updateCourse")
    public CallResult updateCourse(@RequestBody CourseParam courseParam)
{
        return courseService.updateCourse(courseParam);
    }

    @RequestMapping(value = "findCourseById")
    public CallResult findSubjectById(@RequestBody CourseParam
courseParam){
        return courseService.findCourseById(courseParam);
    }


    @RequestMapping(value = "findPage")
    public CallResult findPage(@RequestBody CourseParam courseParam){
        return courseService.findPage(courseParam);
    }
}
```

Service:

```
package com.mszlu.xt.admin.service;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.admin.model.param.CourseParam;

public interface CourseService {
    CallResult saveCourse(CourseParam courseParam);

    CallResult updateCourse(CourseParam courseParam);

    CallResult findCourseById(CourseParam courseParam);

    CallResult findPage(CourseParam courseParam);
}
```

```
package com.mszlu.xt.admin.service.impl;

import com.mszlu.common.model.CallResult;
```

```java
import com.mszlu.common.service.AbstractTemplateAction;
import com.mszlu.xt.admin.domain.CourseDomain;
import com.mszlu.xt.admin.domain.repository.CourseDomainRepository;
import com.mszlu.xt.admin.model.param.CourseParam;
import com.mszlu.xt.admin.service.CourseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

/**
 * @author Jarno
 */
@Service
@Transactional
public class CourseServiceImpl extends AbstractService implements
CourseService {
    @Autowired
    private CourseDomainRepository courseDomainRepository;

    @Override
    public CallResult saveCourse(CourseParam courseParam) {
        CourseDomain courseDomain =
this.courseDomainRepository.createDomain(courseParam);
        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return courseDomain.saveCourse();
            }
        });
    }

    @Override
    public CallResult updateCourse(CourseParam courseParam) {
        CourseDomain courseDomain =
this.courseDomainRepository.createDomain(courseParam);
        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return courseDomain.updateCourse();
            }
        });
    }
    @Override
    public CallResult findPage(CourseParam courseParam) {
        CourseDomain courseDomain =
this.courseDomainRepository.createDomain(courseParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return courseDomain.findPage();
            }
        });
    }

    @Override
    public CallResult findCourseById(CourseParam courseParam) {
        CourseDomain courseDomain =
this.courseDomainRepository.createDomain(courseParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return courseDomain.findCourseById();
            }
        });
    }

}
```

Domain:

```
package com.mszlu.xt.admin.model;

import lombok.Data;

import java.math.BigDecimal;
import java.util.List;

@Data
public class CourseModel {
    private Long id;
    private String courseName;
    private String courseDesc;
```

```java
    private String subjects;
    private BigDecimal coursePrice;
    private BigDecimal courseZhePrice;
    private Integer courseStatus;//正常，下架
    private Integer orderTime;
    private List<Long> subjectIdList;
    private List<SubjectModel> subjectList;
    private String imageUrl;
}
```

```java
package com.mszlu.xt.admin.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mszlu.xt.admin.dao.CourseMapper;
import com.mszlu.xt.admin.dao.CourseSubjectMapper;
import com.mszlu.xt.admin.domain.CourseDomain;
import com.mszlu.xt.admin.domain.SubjectDomain;
import com.mszlu.xt.admin.model.param.CourseParam;
import com.mszlu.xt.admin.model.param.SubjectParam;
import com.mszlu.xt.pojo.Course;
import com.mszlu.xt.pojo.CourseSubject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.util.List;

/**
 * @author Jarno
 */
@Component
public class CourseDomainRepository {

    @Resource
    private CourseMapper courseMapper;
    @Resource
    private CourseSubjectMapper courseSubjectMapper;
    @Autowired
    private SubjectDomainRepository subjectDomainRepository;

    public void saveCourse(Course course) {
        courseMapper.insert(course);
    }

    public CourseDomain createDomain(CourseParam courseParam) {
        return new CourseDomain(courseParam,this);
    }
    public Course findCourse(Long id) {
        return courseMapper.selectById(id);
    }

    public void updateCurse(Course course) {
        this.courseMapper.updateById(course);
    }

    public SubjectDomain createSubjectDomain(SubjectParam subjectParam)
{
        return subjectDomainRepository.createDomain(subjectParam);
    }

    public Page<Course> findCourseListPage(int currentPage,int pageSize)
{
        Page<Course> page = new Page<>(currentPage,pageSize);
        return this.courseMapper.selectPage(page,new
LambdaQueryWrapper<>());
    }

    public void saveCourseSubject(Long subjectId, Long courseId) {
        CourseSubject courseSubject = new CourseSubject();
        courseSubject.setCourseId(courseId);
        courseSubject.setSubjectId(subjectId);
        this.courseSubjectMapper.insert(courseSubject);
    }

    public void deleteCourseSubject(Long courseId) {
        LambdaQueryWrapper<CourseSubject> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(CourseSubject::getCourseId,courseId);
        this.courseSubjectMapper.delete(queryWrapper);
    }

}
```

```
package com.mszlu.xt.admin.domain;
```

```java
import com.alibaba.fastjson.JSON;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mszlu.common.model.CallResult;
import com.mszlu.common.model.ListModel;
import com.mszlu.xt.admin.domain.repository.CourseDomainRepository;
import com.mszlu.xt.admin.model.CourseModel;
import com.mszlu.xt.admin.model.param.CourseParam;
import com.mszlu.xt.pojo.Course;
import com.mszlu.xt.pojo.Subject;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.BeanUtils;

import java.math.BigDecimal;
import java.util.*;

/**
 * @author Jarno
 */
public class CourseDomain {
    private CourseParam courseParam;
    private CourseDomainRepository courseDomainRepository;

    public CourseDomain(CourseParam courseParam, CourseDomainRepository
courseDomainRepository){
        this.courseParam = courseParam;
        this.courseDomainRepository = courseDomainRepository;
    }
    private CourseModel copy(Course course){
        CourseModel courseModel = new CourseModel();
        BeanUtils.copyProperties(course,courseModel);
        return courseModel;
    }
    private List<CourseModel> copyList(List<Course> courseList){
        if (courseList == null || courseList.size() == 0){
            return new ArrayList<>();
        }
        List<Subject> subjectList =
this.courseDomainRepository.createSubjectDomain(null).allSubjectList();
        List<CourseModel> courseModelList = new ArrayList<>();
        for (Course course : courseList){
            CourseModel model = copy(course);

 model.setSubjects(displaySubjects(subjectList,course.getSubjects()));
            courseModelList.add(model);
        }
        return courseModelList;
    }

    private String displaySubjects(List<Subject> subjectList, String
subjects) {
        String[] ss = StringUtils.split(subjects,",");
        String ds = "";
        Map<String,String> map = new HashMap<>();
        for (Subject subject:subjectList){

 map.put(String.valueOf(subject.getId()),subject.getSubjectName());
        }
        for (String s : ss){
            if (StringUtils.isEmpty(ds)){
                ds = map.get(s);
            }else{
                ds += ","+map.get(s);
            }
        }
        return ds;
    }

    private String displayCourseStatus(Integer courseStatus) {
        if (0 == courseStatus){
            return "正常";
        }
        if (1 == courseStatus){
            return "下架";
        }
        return "无此状态";
    }

    public CallResult<Object> saveCourse() {
        String courseName = this.courseParam.getCourseName();
        if (courseName == null){
            courseName = "";
        }
        String courseDesc = this.courseParam.getCourseDesc();
        if (courseDesc == null){
            courseDesc = "";
        }
        List<Long> subjectIdList = this.courseParam.getSubjectIdList();
        String subjects = "";
        for (Long subjectId: subjectIdList){
            subjects += subjectId + ",";
        }
        if (!StringUtils.isEmpty(subjects)){
            subjects = subjects.substring(0,subjects.length() - 1);
        }
        if (StringUtils.isEmpty(subjects)){
            return CallResult.fail(-999,"创建课程必须有学科");
        }
        List<String> subjectList =
this.courseDomainRepository.createSubjectDomain(null).allSubjectIdList()
;

        String[] subjectsArr = StringUtils.split(subjects,",");
        for (String subjectId : subjectsArr){
            if (!subjectList.contains(subjectId)){
                return CallResult.fail(-999,"学科不存在");
            }
        }
        String imageUrl = this.courseParam.getImageUrl();
        if (StringUtils.isEmpty(imageUrl)){
            imageUrl = "";
        }
        BigDecimal cPrice = this.courseParam.getCoursePrice();
        BigDecimal cZhePrice = this.courseParam.getCourseZhePrice();
        Integer courseStatus = this.courseParam.getCourseStatus();
        Integer orderTime = this.courseParam.getOrderTime();

        Course course = new Course();
        course.setCourseName(courseName);
        course.setCourseDesc(courseDesc);
        course.setSubjects(subjects);
        course.setCoursePrice(cPrice);
        course.setCourseZhePrice(cZhePrice);
        course.setCourseStatus(courseStatus);
        course.setOrderTime(orderTime);
        course.setImageUrl(imageUrl);

        this.courseDomainRepository.saveCourse(course);
        for (Long subjectId: subjectIdList){

 this.courseDomainRepository.saveCourseSubject(subjectId,course.getId())
;
        }
        return CallResult.success();
    }
```

```java
    public CallResult<Object> updateCourse() {
        Long id = this.courseParam.getId();
        Course course = this.courseDomainRepository.findCourse(id);
        if (course == null){
            return CallResult.fail(-999,"课程不存在");
        }
        String courseName = this.courseParam.getCourseName();
        if (courseName == null){
            courseName = "";
        }
        String courseDesc = this.courseParam.getCourseDesc();
        if (courseDesc == null){
            courseDesc = "";
        }
        List<Long> subjectIdList = this.courseParam.getSubjectIdList();
        String subjects = "";
        for (Long subjectId: subjectIdList){
            subjects += subjectId + ",";
        }
        if (!StringUtils.isEmpty(subjects)){
            subjects = subjects.substring(0,subjects.length() - 1);
        }
        if (this.courseParam.getOrderTime() != null){
            course.setOrderTime(this.courseParam.getOrderTime());
        }
        String imageUrl = this.courseParam.getImageUrl();
        if (!StringUtils.isEmpty(imageUrl)){
            course.setImageUrl(imageUrl);
        }
        BigDecimal cPrice = this.courseParam.getCoursePrice();
        BigDecimal cZhePrice = this.courseParam.getCourseZhePrice();
        Integer courseStatus = this.courseParam.getCourseStatus();

        course.setCourseName(courseName);
        course.setCourseDesc(courseDesc);
        course.setSubjects(subjects);
        course.setCoursePrice(cPrice);
        course.setCourseZhePrice(cZhePrice);
        course.setCourseStatus(courseStatus);
        course.setOrderTime(this.courseParam.getOrderTime());
        this.courseDomainRepository.updateCurse(course);
        this.courseDomainRepository.deleteCourseSubject(course.getId());
        for (Long subjectId: subjectIdList){
```

```
 this.courseDomainRepository.saveCourseSubject(subjectId,course.getId())
;
        }
        return CallResult.success();
    }

    public CallResult<Object> findPage() {
        int page = this.courseParam.getPage();
        int pageSize = this.courseParam.getPageSize();
        Page<Course> coursePage =
this.courseDomainRepository.findCourseListPage(page,pageSize);

        ListModel listModel = new ListModel();
        listModel.setTotal((int) coursePage.getTotal());
        List<Course> result = coursePage.getRecords();
        List<CourseModel> courseModelList = new ArrayList<>();
        for (Course course : result){
            CourseModel courseModel = copy(course);
            courseModelList.add(courseModel);
        }
        listModel.setList(courseModelList);
        return CallResult.success(listModel);
    }

    public CallResult<Object> findCourseById() {
        Long id = courseParam.getId();
        Course course = this.courseDomainRepository.findCourse(id);
        CourseModel courseModel = copy(course);
        List<Subject> subjectList =
this.courseDomainRepository.createSubjectDomain(null).findSubjectListByC
ourseId(course.getId());
        List<Long> subjectIdList = new ArrayList<>();
        for (Subject subject : subjectList){
            subjectIdList.add(subject.getId());
        }
        courseModel.setSubjectIdList(subjectIdList);

 courseModel.setSubjectList(this.courseDomainRepository.createSubjectDom
ain(null).allSubjectModelList());
        return CallResult.success(courseModel);
    }
}
```

```
public List<SubjectModel> allSubjectModelList() {
        List<Subject> all = subjectDomainRepository.findAll();
        List<SubjectModel> subjectModels = new ArrayList<>();
        for (Subject subject : all) {
            SubjectModel subjectModel = new SubjectModel();
            BeanUtils.copyProperties(subject,subjectModel);
            subjectModels.add(subjectModel);
        }
        return subjectModels;
    }

    public List<String> allSubjectIdList() {
        List<Subject> all = this.subjectDomainRepository.findAll();
        return all.stream().map(subject ->
subject.getId().toString()).collect(Collectors.toList());
    }

    public List<Subject> findSubjectListByCourseId(Long courseId) {
        return
this.subjectDomainRepository.findSubjectListByCourseId(courseId);
    }
```

```
 public List<Subject> findAll() {
        return this.subjectMapper.selectList(new LambdaQueryWrapper<>
());
    }

    public List<Subject> findSubjectListByCourseId(Long courseId) {
        return this.subjectMapper.findSubjectListByCourseId(courseId);
    }
```

```
package com.mszlu.xt.admin.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.Subject;

import java.util.List;

public interface SubjectMapper extends BaseMapper<Subject> {

    List<Subject> findSubjectListByCourseId(Long courseId);
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.mszlu.xt.admin.dao.SubjectMapper">

    <resultMap id="SubjectMap" type="com.mszlu.xt.pojo.Subject">
        <id column="id" property="id" jdbcType="BIGINT" />
        <result column="subject_name" property="subjectName"
jdbcType="VARCHAR" />
        <result column="subject_grade" property="subjectGrade"
jdbcType="VARCHAR" />
        <result column="subject_term" property="subjectTerm"
jdbcType="VARCHAR" />
    </resultMap>
    <resultMap id="SubjectUnitMap" type="com.mszlu.xt.pojo.SubjectUnit">
        <id column="id" property="id" jdbcType="BIGINT" />
        <result column="subject_id" property="subjectId"
jdbcType="BIGINT" />
        <result column="subject_unit" property="subjectUnit"
jdbcType="INTEGER" />
    </resultMap>

    <sql id="column_no_id">
        subject_name,subject_grade,subject_term
    </sql>
    <sql id="all_column">
        id,subject_name,subject_grade,subject_term
    </sql>


    <select id="findSubjectListByCourseId" parameterType="long"
resultMap="SubjectMap">
        SELECT <include refid="all_column" />
        FROM t_subject
        WHERE id IN (SELECT subject_id FROM t_course_subject WHERE
course_id=#{courseId})
    </select>

</mapper>
```

> *接下来，完成web端功能，最核心的代码*