支付

1. 课程购买

> *课程购买需要先查询课程的详细信息以及课程可用的优惠券，然后才能提交订单*

1.1 课程详情

1.1.1 API

请求地址：/api/course/courseDetail

请求参数:{courseId: 5}

返回值:

```
{
    "code":2000000000,
    "message":"default success",
    "result":{
        "courseId":5,
        "courseName":"七年级历史下",
        "subjectInfo":"历史 七年级 下",
        "courseTime":365,
        "price":99
    },
    "success":true
}
```

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

import lombok.Data;

import java.math.BigDecimal;

@Data
public class CourseDetailModel {
```

```
    private Long courseId;
    private String courseName;
    private String subjectInfo;
    private Integer courseTime;
    private BigDecimal price;
}
```

1.1.2 需求

> *课程详情就是根据课程id，将课程的信息展示出来，供用户查看*

1.1.3 编码

Controller:

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

Service:

```
 CallResult courseDetail(CourseParam courseParam);
```

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

Domain

```
public CallResult<Object> courseDetail() {
        Long courseId = this.courseParam.getCourseId();
        Course course =
this.courseDomainRepository.findCourseById(courseId);
        if (course == null){
            return
CallResult.fail(BusinessCodeEnum.COURSE_NOT_EXIST.getCode(),"订单不存在");
        }
        CourseDetailModel courseDetailModel = new CourseDetailModel();
        courseDetailModel.setCourseId(courseId);
        courseDetailModel.setCourseName(course.getCourseName());
        courseDetailModel.setCourseTime(course.getOrderTime());
        courseDetailModel.setPrice(course.getCourseZhePrice());
        List<SubjectModel> subjectList =
this.courseDomainRepository.createSubjectDomain(null).findSubjectModelLi
stByCourseId(courseId);
        StringBuilder subjectStr = new StringBuilder();
        for (SubjectModel subject : subjectList){
            subjectStr.append(subject.getSubjectName()).append("
").append(subject.getSubjectGrade()).append("
").append(subject.getSubjectTerm()).append(",");
        }
        int length = subjectStr.toString().length();
        if (length > 0){
            subjectStr = new StringBuilder(subjectStr.substring(0,
length - 1));
        }
        courseDetailModel.setSubjectInfo(subjectStr.toString());
        return CallResult.success(courseDetailModel);
    }
```

1.1.4 测试

1.2 个人优惠券

> *查看课程详情的时候，需要告诉用户是否有可用的优惠券，优惠券是通过推广渠道，比如微信发放给用户的*

1.2.1 数据库设计

```
CREATE TABLE `xt`.`t_coupon`  (
  `id` bigint(0) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT
NULL,
  `coupon_desc` varchar(255) CHARACTER SET utf8 COLLATE utf8_unicode_ci
NOT NULL COMMENT '详细描述信息',
  `price` decimal(10, 2) NOT NULL COMMENT '优惠金额',
  `max` decimal(10, 2) NOT NULL COMMENT '满多少使用',
  `number` int(0) NOT NULL COMMENT '总数量',
  `use_number` int(0) NOT NULL COMMENT '发放数量',
  `status` tinyint(0) NOT NULL COMMENT '状态 1 有效 2 无效',
  `start_time` bigint(0) NOT NULL COMMENT '可以使用时间 -1代表任何时间',
  `expire_time` bigint(0) NOT NULL COMMENT '优惠券过期时间 -1 代表任何时间',
  `dis_status` tinyint(0) NOT NULL COMMENT '0 不需要满减 1 需要满减',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 3 CHARACTER SET = utf8 COLLATE =
utf8_unicode_ci ROW_FORMAT = DYNAMIC;
```

```
CREATE TABLE `xt`.`t_user_coupon`  (
  `id` bigint(0) NOT NULL AUTO_INCREMENT,
  `user_id` bigint(0) NOT NULL,
  `coupon_id` bigint(0) NOT NULL,
  `status` tinyint(0) NOT NULL COMMENT '0 未使用 1 使用 2 过期',
  `expire_time` bigint(0) NOT NULL,
  `start_time` bigint(0) NOT NULL,
  `use_time` bigint(0) NOT NULL COMMENT '使用时间',
  `source` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NOT NULL COMMENT '来源',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 3 CHARACTER SET = utf8mb4 COLLATE =
utf8mb4_unicode_ci ROW_FORMAT = DYNAMIC;
```

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

import lombok.Data;

import java.math.BigDecimal;

@Data
public class Coupon {
    private Long id;
    private String name;
```

```
    private String couponDesc;
    private BigDecimal price;
    //满多少才能使用优惠券
    private BigDecimal max;
    private Integer number;
    private Integer useNumber;
    private Integer status;
    //开始时间
    private Long startTime;
    //过期时间
    private Long expireTime;
    //是否需要满减 0不需要 1 需要
    private Integer disStatus;
}
```

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

import lombok.Data;

@Data
public class UserCoupon {
    private Long id;
    private Long userId;
    private Long couponId;
    // 0 未使用 1 已使用 2过期
    private Integer status;

    private Long startTime;

    private Long expireTime;
    //使用时间
    private Long useTime;
    //来源
    private String source;
}
```

1.2.2 API

请求路径：/api/course/myCoupon

请求参数: {courseId: 5}

返回数据:

```
{
    "code":2000000000,
    "message":"default success",
    "result":[
        {
            "name":"优惠券减100",
            "amount":100,
            "couponId":1
        },
        {
            "name":"优惠券满200减100",
            "amount":50,
            "couponId":2
        }
    ],
    "success":true
}
```

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

import lombok.Data;

import java.math.BigDecimal;

@Data
public class UserCouponModel {
    private String name;
    private BigDecimal amount;
    private Long couponId;
}
```

1.2.3 前端处理

CouponSelect.vue:

```
<template>
  <el-select v-bind="$attrs" v-on="$listeners" no-data-text="没有可用的优惠券" placeholder="请选择优惠券">
    <el-option v-for="(item, index) in subjects" :label="item.name" :key="index" :value="item.couponId"/>
  </el-select>
</template>
```

1.2.3 需求和逻辑

1. 用户必须有优惠券
2. 优惠券必须可用
    1. 不能过期
    2. 不能被使用过
    3. 如果是满减券，课程金额需要符合
3. 返回可优惠的金额

1.2.4 编码

Controller

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

Service

```
CallResult myCoupon(CourseParam courseParam);
```

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

Domain

```
 public CallResult<Object> myCoupon() {
        Long userId = UserThreadLocal.get();
        Long courseId = this.courseParam.getCourseId();
        List<UserCouponModel> list =
this.courseDomainRepository.createCouponDomain(null).findUserCoupon(user
Id,courseId);
        return CallResult.success(list);
    }

    public Course findCourseById(Long courseId) {
        return courseDomainRepository.findCourseById(courseId);
    }
```

```
 @Autowired
    private CouponDomainRepository couponDomainRepository;

    public CouponDomain createCouponDomain(CouponParam couponParam) {
        return this.couponDomainRepository.createDomain(couponParam);
    }
```

CouponDomainRepository:

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

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
```

```java
import com.mszlu.xt.pojo.Coupon;
import com.mszlu.xt.pojo.UserCoupon;
import com.mszlu.xt.web.dao.CouponMapper;
import com.mszlu.xt.web.dao.UserCouponMapper;
import com.mszlu.xt.web.domain.CouponDomain;
import com.mszlu.xt.web.domain.CourseDomain;
import com.mszlu.xt.web.model.params.CouponParam;
import com.mszlu.xt.web.model.params.CourseParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

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

@Component
public class CouponDomainRepository {

    @Resource
    private UserCouponMapper userCouponMapper;
    @Resource
    private CouponMapper couponMapper;
    @Autowired
    private CourseDomainRepository courseDomainRepository;

    public CouponDomain createDomain(CouponParam couponParam) {
        return new CouponDomain(this,couponParam);
    }

    public List<UserCoupon> findUserCouponByUserId(Long userId) {
        LambdaQueryWrapper<UserCoupon> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserCoupon::getUserId,userId);
        //0代表 未使用 应该做成枚举
        queryWrapper.eq(UserCoupon::getStatus,0);
        List<UserCoupon> userCouponList =
userCouponMapper.selectList(queryWrapper);
        return userCouponList;
    }

    public Coupon findCouponById(Long couponId) {
        return couponMapper.selectById(couponId);
    }

    public CourseDomain createCourseDomain(CourseParam courseParam) {
        return courseDomainRepository.createDomain(courseParam);
```

```
    }
}
```

CouponDomain:

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

import com.mszlu.xt.pojo.Coupon;
import com.mszlu.xt.pojo.Course;
import com.mszlu.xt.pojo.UserCoupon;
import com.mszlu.xt.web.domain.repository.CouponDomainRepository;
import com.mszlu.xt.web.model.UserCouponModel;
import com.mszlu.xt.web.model.params.CouponParam;

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

public class CouponDomain {

    private CouponDomainRepository couponDomainRepository;
    private CouponParam couponParam;
    public CouponDomain(CouponDomainRepository couponDomainRepository,
CouponParam couponParam) {
        this.couponDomainRepository = couponDomainRepository;
        this.couponParam =couponParam;
    }

    public List<UserCouponModel> findUserCoupon(Long userId,Long
courseId) {
        List<UserCoupon> userCouponList =
couponDomainRepository.findUserCouponByUserId(userId);
        List<UserCouponModel> userCouponModelList = new ArrayList<>();
        for (UserCoupon userCoupon : userCouponList) {
            Long startTime = userCoupon.getStartTime();
            Long expireTime = userCoupon.getExpireTime();
            long currentTimeMillis = System.currentTimeMillis();
            if (startTime != -1 && currentTimeMillis < startTime){
                continue;
            }
            if (expireTime != -1 && currentTimeMillis > expireTime){
                continue;
            }
```

```
            Long couponId = userCoupon.getCouponId();
            Coupon coupon =
couponDomainRepository.findCouponById(couponId);
            Integer disStatus = coupon.getDisStatus();
            if (disStatus == 1){
                //需要满足满减条件
                Course course =
this.couponDomainRepository.createCourseDomain(null).findCourseById(cour
seId);
                BigDecimal courseZhePrice = course.getCourseZhePrice();
                if (coupon.getMax().compareTo(courseZhePrice) > 0){
                    //最大满减 大于课程价格 代表不可使用
                    continue;
                }
            }
            UserCouponModel userCouponModel = new UserCouponModel();
            userCouponModel.setAmount(coupon.getPrice());
            userCouponModel.setCouponId(couponId);
            userCouponModel.setName(coupon.getName());
            userCouponModelList.add(userCouponModel);

        }
        return userCouponModelList;
    }
}
```

Mapper

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

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

public interface UserCouponMapper extends BaseMapper<UserCoupon> {
}
```

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

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

public interface CouponMapper extends BaseMapper<Coupon> {
}
```

1.2.5 测试

2. 创建订单

> *用户选择提交订单，需要创建一个订单，返回给用户付款的金额，选择的课程以及订单号，订单号要保证唯一*

2.1 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	
user_id	bigint	0	0	☑	☐		
course_id	bigint	0	0	☑	☐		
order_id	varchar	127	0	☑	☐		
order_amount	decimal	10	2	☑	☐		订单金额
order_status	tinyint	0	0	☑	☐		
pay_type	tinyint	0	0	☑	☐		支付方式
pay_status	tinyint	0	0	☑	☐		
create_time	bigint	0	0	☑	☐		
expire_time	bigint	0	0	☑	☐		过期时间
coupon_id	bigint	0	0	☑	☐		使用的优惠券id 0 为未使用
pay_order_id	varchar	127	0	☑	☐		
pay_time	bigint	0	0	☑	☐		支付时间

```sql
CREATE TABLE `xt`.`order`  (
  `id` bigint(0) NOT NULL AUTO_INCREMENT,
  `user_id` bigint(0) NOT NULL,
  `course_id` bigint(0) NOT NULL,
```

```
  `order_id` varchar(127) CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci NOT NULL,
  `order_amount` decimal(10, 2) NOT NULL COMMENT '订单金额',
  `order_status` tinyint(0) NOT NULL,
  `pay_type` tinyint(0) NOT NULL COMMENT '支付方式',
  `pay_status` tinyint(0) NOT NULL,
  `create_time` bigint(0) NOT NULL,
  `expire_time` bigint(0) NOT NULL COMMENT '过期时间',
  `coupon_id` bigint(0) NOT NULL DEFAULT 0 COMMENT '使用的优惠券id 0 为未使
用',
  `pay_order_id` varchar(127) CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci NOT NULL,
  `pay_time` bigint(0) NOT NULL COMMENT '支付时间',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 10 CHARACTER SET = utf8mb4 COLLATE =
utf8mb4_unicode_ci ROW_FORMAT = DYNAMIC;
```

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

import lombok.Data;

import java.math.BigDecimal;

@Data
public class Order {

    private Long id;
    private Long userId;
    private String orderId;
    private String payOrderId;
    private Long courseId;
    private BigDecimal orderAmount;
    private Integer orderStatus;
    private Integer payType;
    private Integer payStatus;
    private Long createTime;
    private Integer expireTime;
    private Long payTime;
    private Long couponId;

}
```

```
package com.mszlu.xt.web.model.enums;
```

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

/**
 * @author Jarno
 */
public enum OrderStatus {
    /**
     * look name
     */
    INIT(0,"初始化"),
    COMMIT(1,"已提交"),
    PAYED(2,"已付款"),
    CANCEL(3,"已取消"),
    REFUND(4,"已退款");

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

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

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

    private int code;
    private String msg;

    OrderStatus(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;
    }
}
```

```
package com.mszlu.xt.web.model.enums;

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

/**
 * @author Jarno
 */
public enum PayType {
    /**
     * look name
     */
    WX(1,"wx"),
    ALI_PAY(2,"支付宝");

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

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

    /**
     * 根据code获取枚举值
     * @param code
     * @return
```

```
     */
    public static PayType valueOfCode(int code){
        return CODE_MAP.get(code);
    }

    private int code;
    private String msg;

    PayType(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;
    }
}
```

```
package com.mszlu.xt.web.model.enums;

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

/**
 * @author Jarno
 */
public enum PayStatus {
    /**
     * look name
     */
    NO_PAY(1,"未付款"),
```

```
    PAYED(2,"已付款");

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

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

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

    private int code;
    private String msg;

    PayStatus(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;
    }
}
```

2.2 Api

请求接口: /api/order/submitOrder

参数: {courseId: 4, couponId: 1}

```
package com.mszlu.xt.web.model.params;

import lombok.Data;

import javax.servlet.http.HttpServletRequest;

@Data
public class OrderParam {


    private int page = 1;
    private int pageSize = 20;
    private Long userId;
    private Long courseId;
    private Long couponId;
    private Integer payType;
    private String orderId;

    private Integer orderStatus;

    private HttpServletRequest request;

    private String nickname;
}
```

返回:

```
{
    "code":2000000000,
    "message":"default success",
    "result":{
        "orderId":"16359392255958257314441226356203335618",
        "subject":"历史",
        "courseName":"七年级历史上",
        "amount":0.01
    },
    "success":true
}
```

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

import lombok.Data;

import java.math.BigDecimal;

@Data
public class OrderDisplayModel {
    private String orderId;
    private String subject;
    private String courseName;
    private BigDecimal amount;
}
```

2.3 业务逻辑

1. 检查课程是否存在
2. 检查优惠券是否可用
3. 初始化订单，生成订单号
4. 展示课程中的科目详情

2.4 编码

Controller

```
package com.mszlu.xt.web.api;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.model.service.OrderService;
```

```
import com.mszlu.xt.web.model.params.OrderParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("order")
public class OrderApi {

    @Autowired
    private OrderService orderService;

    @PostMapping("submitOrder")
    public CallResult submitOrder(@RequestBody OrderParam orderParam){
        return orderService.submitOrder(orderParam);
    }
}
```

Service

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

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.web.model.params.OrderParam;

public interface OrderService {

    CallResult submitOrder(OrderParam orderParam);
}
```

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

import com.mszlu.common.model.CallResult;
import com.mszlu.common.service.AbstractTemplateAction;
import com.mszlu.xt.model.service.OrderService;
import com.mszlu.xt.web.domain.OrderDomain;
import com.mszlu.xt.web.domain.repository.OrderDomainRepository;
import com.mszlu.xt.web.model.params.OrderParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
@Service
public class OrderServiceImpl extends AbstractService implements
OrderService {

    @Autowired
    private OrderDomainRepository orderDomainRepository;

    @Override
    public CallResult submitOrder(OrderParam orderParam) {
        OrderDomain orderDomain =
orderDomainRepository.createDomain(orderParam);
        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return orderDomain.submitOrder();
            }
        });
    }
}
```

Domain

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

import com.mszlu.xt.pojo.Order;
import com.mszlu.xt.web.dao.OrderMapper;
import com.mszlu.xt.web.domain.CouponDomain;
import com.mszlu.xt.web.domain.CourseDomain;
import com.mszlu.xt.web.domain.OrderDomain;
import com.mszlu.xt.web.domain.SubjectDomain;
import com.mszlu.xt.web.model.params.CouponParam;
import com.mszlu.xt.web.model.params.CourseParam;
import com.mszlu.xt.web.model.params.OrderParam;
import com.mszlu.xt.web.model.params.SubjectParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class OrderDomainRepository {
```

```
    @Resource
    private OrderMapper orderMapper;
    @Autowired
    private CourseDomainRepository courseDomainRepository;
    @Autowired
    private CouponDomainRepository couponDomainRepository;
    @Autowired
    private SubjectDomainRepository subjectDomainRepository;

    public OrderDomain createDomain(OrderParam orderParam) {
        return new OrderDomain(this,orderParam);
    }

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

    public CourseDomain createCourseDomain(CourseParam courseParam) {
        return courseDomainRepository.createDomain(courseParam);
    }

    public CouponDomain createCouponDomain(CouponParam couponParam) {
        return couponDomainRepository.createDomain(couponParam);
    }

    public void saveOrder(Order order) {
        this.orderMapper.insert(order);
    }
}
```

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

import com.mszlu.common.model.BusinessCodeEnum;
import com.mszlu.common.model.CallResult;
import com.mszlu.common.utils.CommonUtils;
import com.mszlu.common.utils.UserThreadLocal;
import com.mszlu.xt.pojo.*;
import com.mszlu.xt.web.domain.repository.OrderDomainRepository;
import com.mszlu.xt.web.model.OrderDisplayModel;
import com.mszlu.xt.web.model.SubjectModel;
import com.mszlu.xt.web.model.enums.OrderStatus;
```

```
import com.mszlu.xt.web.model.enums.PayStatus;
import com.mszlu.xt.web.model.enums.PayType;
import com.mszlu.xt.web.model.params.OrderParam;

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

public class OrderDomain {
    private OrderDomainRepository orderDomainRepository;
    private OrderParam orderParam;

    public OrderDomain(OrderDomainRepository orderDomainRepository,
OrderParam orderParam) {
        this.orderDomainRepository = orderDomainRepository;
        this.orderParam = orderParam;
    }

    public CallResult<Object> submitOrder() {
        Long userId = UserThreadLocal.get();
        Long courseId = this.orderParam.getCourseId();
        Course course =
this.orderDomainRepository.createCourseDomain(null).findCourseById(cours
eId);
        if (course == null){
            return
CallResult.fail(BusinessCodeEnum.COURSE_NOT_EXIST.getCode(),"course not
exist");
        }
        Long couponId = this.orderParam.getCouponId();
        BigDecimal couponPrice = new BigDecimal(0);
        if (couponId != null){
            Coupon coupon =
this.orderDomainRepository.createCouponDomain(null).findCouponById(coupo
nId);
            if (coupon != null){
                couponPrice = checkCoupon(userId,coupon);
            }
        }else{
            couponId = -1L;
        }
        Order order = new Order();
        order.setCourseId(courseId);
        order.setCouponId(couponId);
        long createTime = System.currentTimeMillis();
        order.setCreateTime(createTime);
```

```
        order.setExpireTime(course.getOrderTime());
//
order.setOrderAmount(course.getCourseZhePrice().subtract(couponPrice));
        //为了测试 价格定为1分
        order.setOrderAmount(BigDecimal.valueOf(0.01));
        String orderId = createTime +
String.valueOf(CommonUtils.random5Num()) + userId;
        order.setOrderId(orderId);
        order.setOrderStatus(OrderStatus.INIT.getCode());
        order.setPayType(PayType.WX.getCode());//默认微信支付
        order.setPayStatus(PayStatus.NO_PAY.getCode());
        order.setUserId(userId);
        order.setPayOrderId(orderId);
        order.setPayTime(0L);
        this.orderDomainRepository.saveOrder(order);

        List<SubjectModel> subjectList =
this.orderDomainRepository.createSubjectDomain(null).findSubjectModelLis
tByCourseId(courseId);
        OrderDisplayModel orderDisplayModel = new OrderDisplayModel();
        orderDisplayModel.setAmount(order.getOrderAmount());
        orderDisplayModel.setCourseName(course.getCourseName());
        orderDisplayModel.setOrderId(orderId);
        StringBuilder subject = new StringBuilder();
        for (SubjectModel subjectModel : subjectList){
            subject.append(subjectModel.getSubjectName()).append(",");
        }
        if (subject.toString().length() > 0){
            subject = new
StringBuilder(subject.substring(0,subject.toString().length() - 1));
        }
        orderDisplayModel.setSubject(subject.toString());
        return CallResult.success(orderDisplayModel);
    }

    private BigDecimal checkCoupon(Long userId,Coupon coupon) {
        //检查优惠券是否可用，并返回优惠券的价格，订单生成的时候 需要将价格考虑进去
        //并且暂时标记 优惠券被使用，如果订单取消 可以将优惠券才还回去
        Long couponId = coupon.getId();
        CouponDomain couponDomain =
this.orderDomainRepository.createCouponDomain(null);
        UserCoupon userCoupon =
couponDomain.findUserCouponByUserId(userId,couponId);
        if (userCoupon == null){
            //条件不符合
```

```
            return BigDecimal.ZERO;
        }
        Long startTime = userCoupon.getStartTime();
        Long expireTime = userCoupon.getExpireTime();
        long currentTimeMillis = System.currentTimeMillis();
        if (expireTime != -1 && currentTimeMillis > expireTime){
            //过期了
            return BigDecimal.ZERO;
        }
        if (startTime != -1 && currentTimeMillis < startTime){
            //未到使用时间
            return BigDecimal.ZERO;
        }
        //标记为 已使用 未消费
        userCoupon.setStatus(4);
        couponDomain.updateCouponStatus(userCoupon);
        return coupon.getPrice();
    }
}
```

CouponDomain:

```
 public UserCoupon findUserCouponByUserId(Long userId, Long couponId) {
        return
this.couponDomainRepository.findUserCouponByUserIdAndCouponId(userId,cou
ponId);
    }

    public void updateCouponStatus(UserCoupon userCoupon) {
        this.couponDomainRepository.updateCouponStatus(userCoupon);
    }
```

```
public UserCoupon findUserCouponByUserIdAndCouponId(Long userId, Long
couponId) {
        LambdaQueryWrapper<UserCoupon> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserCoupon::getUserId,userId);
        queryWrapper.eq(UserCoupon::getCouponId,couponId);
        //0代表 未使用 应该做成枚举
        queryWrapper.eq(UserCoupon::getStatus,0);
        queryWrapper.last("limit 1");
        UserCoupon userCoupon =
userCouponMapper.selectOne(queryWrapper);
```

```
        return userCoupon;
    }

    public void updateCouponStatus(UserCoupon userCoupon) {
        LambdaUpdateWrapper<UserCoupon> updateWrapper =
Wrappers.lambdaUpdate();
        updateWrapper.eq(UserCoupon::getId,userCoupon.getId());
        updateWrapper.set(UserCoupon::getStatus,userCoupon.getStatus());
        userCouponMapper.update(null, updateWrapper);
    }
```

CourseDomain:

```
   public Course findCourseById(Long courseId) {
        return courseDomainRepository.findCourseById(courseId);
    }
```

Mapper

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

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

public interface OrderMapper extends BaseMapper<Order> {
}
```

2.5 测试

3. 创建订单的问题

> *我们需要想这么一个问题，如果用户创建了订单，但是没有后续的支付操作会怎么样?*
>
> 1. *优惠券被占用了，用户无法使用*
> 2. *此条订单成为了垃圾数据，一直是等待支付的一个状态*
> 3. *参考电商，一般30分未支付的话，订单就会被删除*

这里我们利用RocketMQ的延时消息来解决

> *逻辑：当订单创建成功，发送一条延时消息，延时30分钟进行消费，在消费的时候进行判断，如果订单未支付就进行取消操作，如果有使用优惠券，将优惠券的状态回退*

3.1 发送延时消息

```
 //16代表30分钟 延迟30m执行消费 3代表10秒
        Map<String,String> map = new HashMap<>();
        map.put("orderId",order.getOrderId());
        map.put("time","10");

 this.orderDomainRepository.mqService.sendDelayedMessage("create_order_d
elay",map,3);
```

```
 @Autowired
    public MqService mqService;
```

```
package com.mszlu.xt.web.domain.mq;

import lombok.extern.slf4j.Slf4j;
import org.apache.rocketmq.spring.core.RocketMQTemplate;
import org.joda.time.DateTime;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.support.MessageBuilder;
import org.springframework.stereotype.Component;

@Component
@Slf4j
public class MqService {

    @Autowired
    private RocketMQTemplate rocketMQTemplate;

    //延迟等级 RocketMQ不支持任意时间的延时，只支持以下几个固定的延时等级  "1s 5s
10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h"
    public void sendDelayedMessage(String topic,Object data,int delay){
        MessageBuilder<Object> messageBuilder =
MessageBuilder.withPayload(data);
        //timeout 发送的超时时间
        rocketMQTemplate.syncSend(topic, messageBuilder.build(), 3000,
delay);
```

```
        //采用同步发送，确保消息一定发送成功
        log.info("MqService 发延迟消息的时间:{},消息内容:{},topic:{},delay:
{}",new DateTime(),data,topic,delay);
    }
}
```

3.2 延时消息消费

```
package com.mszlu.xt.web.domain.mq;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.TypeReference;
import com.mszlu.xt.pojo.Order;
import com.mszlu.xt.web.domain.repository.OrderDomainRepository;
import com.mszlu.xt.web.model.enums.OrderStatus;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.apache.rocketmq.spring.annotation.RocketMQMessageListener;
import org.apache.rocketmq.spring.core.RocketMQListener;
import org.joda.time.DateTime;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.util.Map;

@Component
@RocketMQMessageListener(topic = "create_order_delay",consumerGroup =
"create_order_delay_group")
@Slf4j
public class OrderCreateMQConsumer implements RocketMQListener<String> {

    @Autowired
    private OrderDomainRepository orderDomainRepository;

    @Override
    public void onMessage(String message) {
        log.info("订单延迟消费的时间:{}, 消息:{}",new DateTime(),message);
        //判断订单是否支付 如果没有 则取消
        if (StringUtils.isNotBlank(message)){
            Map<String, String> messageMap =
JSON.parseObject(message,new TypeReference<Map<String,String>>(){});
            String orderId = messageMap.get("orderId");
```

```
            String time = messageMap.get("time");
            Integer delayTime = Integer.parseInt(time);
            Order order  =
orderDomainRepository.findOrderByOrderId(orderId);
            if (order != null){
                Integer orderStatus = order.getOrderStatus();
                if (OrderStatus.INIT.getCode() == orderStatus
                        || OrderStatus.COMMIT.getCode() == orderStatus){
                    //订单不是 已付款 取消 退款的 证明 此订单无效 需要取消
                    long currentTimeMillis = System.currentTimeMillis();
                    if (currentTimeMillis - order.getCreateTime() >=
delayTime * 1000){
                        //满足条件 需要取消
                        boolean isUpdate =
this.orderDomainRepository.updateOrderStatus(order,OrderStatus.CANCEL.ge
tCode());
                        if (!isUpdate){
                            throw  new RuntimeException("订单状态可能被修
改，不能修改");
                        }
                        log.info("订单已经被取消");
                    }else{
                        throw  new RuntimeException("订单时间不满足,不应该取
消");
                    }
                }
                log.info("订单不满足条件~~");
            }
        }
    }
}
```

> *抛异常 代表 此条消息未消费，会隔一会消费一次，直至消费完成*

4. 微信支付二维码

微信支付：https://pay.weixin.qq.com/wiki/doc/apiv3/open/pay/chapter2_7_2.shtml

> *在创建完订单之后，前端会将orderId传给后台，请求微信支付的二维码，后端通过微信的api接口，实现微信支付的二维码返回，并更新订单状态*

4.1 接口说明

请求路径: /api/order/wxPay

请求参数: {orderId: "1636384694008288651444122635620335618", payType: 1}

返回值:

```
{
    "code":2000000000,
    "message":"default success",
    "result":"weixin://wxpay/bizpayurl?pr=7JbNuGOzz",
    "success":true
}
```

返回的是支付链接，由前端转为二维码显示

4.2 业务逻辑

1. 检查订单是否存在
2. 检查订单状态是否符合
3. 检查支付状态是否符合
4. 检查购买的课程是否可用
5. 生成微信的支付订单号
    1. 微信每次支付都需要重新生成订单号
    2. 将微信支付订单和此次购买订单号做关联
6. 调用微信API，生成支付二维码

4.3 编码

Controller

```
@PostMapping("wxPay")
    public CallResult wxPay(@RequestBody OrderParam orderParam){
        return orderService.wxPay(orderParam);
    }
```

Service

```
 CallResult wxPay(OrderParam orderParam);
```

```
 @Override
    public CallResult wxPay(OrderParam orderParam) {
        OrderDomain orderDomain =
orderDomainRepository.createDomain(orderParam);
        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return orderDomain.wxPay();
            }
        });
    }
```

Domain

```
public CallResult<Object> wxPay() {
        Long userId = UserThreadLocal.get();
        String orderId = this.orderParam.getOrderId();
        Order order =
this.orderDomainRepository.findOrderByOrderId(orderId);
        if (order == null){
            return
CallResult.fail(BusinessCodeEnum.ORDER_NOT_EXIST.getCode(),"订单不存在");
        }
        if (order.getOrderStatus().equals(OrderStatus.PAYED.getCode())){
            return
CallResult.fail(BusinessCodeEnum.ORDER_AREADY_PAYED.getCode(),"订单已付
款");
        }
        if (order.getPayStatus().equals(PayStatus.PAYED.getCode())){
            return
CallResult.fail(BusinessCodeEnum.ORDER_AREADY_PAYED.getCode(),"订单已付
款");
        }
```

```
        Integer payType = this.orderParam.getPayType();
        //用于测试
        order.setOrderAmount(BigDecimal.valueOf(0.01));
        order.setPayType(payType);
        Long courseId = order.getCourseId();
        Course course =
this.orderDomainRepository.createCourseDomain(null).findCourseById(cours
eId);
        if (course == null) {
            return CallResult.fail(-999,"课程已被删除");
        }
        String payOrderId = System.currentTimeMillis() +
String.valueOf(CommonUtils.random5Num()) + userId%10000;
        order.setPayOrderId(payOrderId);
        this.orderDomainRepository.updatePayOrderId(order);

        WxPayDomain wxPayDomain = new
WxPayDomain(this.orderDomainRepository.wxPayConfiguration);
        WxPayUnifiedOrderRequest orderRequest = new
WxPayUnifiedOrderRequest();

 orderRequest.setNotifyUrl(this.orderDomainRepository.wxPayConfiguration
.wxNotifyUrl);
        orderRequest.setBody(course.getCourseName());
        orderRequest.setOutTradeNo(payOrderId);
        orderRequest.setProductId(String.valueOf(courseId));

 orderRequest.setTotalFee(BaseWxPayRequest.yuanToFen(String.valueOf(orde
r.getOrderAmount().doubleValue())));//元转成分
        orderRequest.setSpbillCreateIp("182.92.102.161");
        orderRequest.setTradeType("NATIVE");
        orderRequest.setTimeStart(new
DateTime(order.getCreateTime()).toString("yyyyMMddHHmmss"));
        try {
            WxPayNativeOrderResult wxPayNativeOrderResult =
wxPayDomain.getWxPayService().createOrder(orderRequest);

 this.orderDomainRepository.updateOrderStatusAndPayType(order,OrderStatu
s.COMMIT.getCode());
            return
CallResult.success(wxPayNativeOrderResult.getCodeUrl());
        } catch (WxPayException e) {
            e.printStackTrace();
```

```
            return
CallResult.fail(BusinessCodeEnum.PAY_ORDER_CREATE_FAIL.getCode(),"create
order fail");
        }
    }
```

1. 添加wx支付相关的配置

```
# 微信支付相关配置
wx.pay.appId=123
wx.open.config.pay.secret=123
wx.pay.mchId=123
wx.pay.mchKey=123
wx.notify.url=http://www.mszlu.com/api/order/notify
```

2. 配置类

```
package com.mszlu.common.wx.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class WxPayConfiguration {

    @Value("${wx.pay.appId}")
    public String payAppId;
    @Value("${wx.pay.mchId}")
    public String mchId;
    @Value("${wx.pay.mchKey}")
    public String mchKey;
    @Value("${wx.notify.url}")
    public String wxNotifyUrl;
}
```

3. 开启微信支付支持

```
package com.mszlu.common.wx.config;

import com.mszlu.common.service.impl.ServiceTemplateImpl;
import org.springframework.context.annotation.Import;
```

```
import java.lang.annotation.*;

@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Import(WxPayConfiguration.class)
public @interface EnableWxPay {
}
```

```
package com.mszlu.xt.web.config;

import com.mszlu.common.cache.EnableCache;
import com.mszlu.common.service.EnableService;
import com.mszlu.common.wx.config.EnableWxPay;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;

@Configuration
@EnableCache
@EnableService
@EnableWxPay
public class InitConfig {
}
```

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

import com.github.binarywang.wxpay.config.WxPayConfig;
import com.github.binarywang.wxpay.service.WxPayService;
import com.github.binarywang.wxpay.service.impl.WxPayServiceImpl;
import com.mszlu.common.wx.config.WxPayConfiguration;

public class WxPayDomain {
    private WxPayConfiguration wxPayConfig;
    public WxPayDomain(WxPayConfiguration config){
        this.wxPayConfig = config;
    }

    public WxPayService getWxPayService(){
        WxPayConfig payConfig = new WxPayConfig();
        payConfig.setAppId(wxPayConfig.payAppId);
```

```
            payConfig.setMchId(wxPayConfig.mchId);
            payConfig.setMchKey(wxPayConfig.mchKey);
            payConfig.setKeyPath("classpath:apiclient_cert.p12");
            WxPayService wxPayService = new WxPayServiceImpl();
            wxPayService.setConfig(payConfig);
            return wxPayService;
        }

    }
```

DomainRepository

```
    @Autowired
    public WxPayConfiguration wxPayConfiguration;
ublic void updatePayOrderId(Order order) {
        LambdaUpdateWrapper<Order> updateWrapper =
Wrappers.lambdaUpdate();
        updateWrapper.eq(Order::getId,order.getId());
        updateWrapper.set(Order::getPayOrderId,order.getPayOrderId());
        this.orderMapper.update(null, updateWrapper);
    }


    public boolean updateOrderStatusAndPayType(Order order,Integer
updateOrderStatus) {
        LambdaUpdateWrapper<Order> updateWrapper =
Wrappers.lambdaUpdate();
        updateWrapper.eq(Order::getOrderId,order.getOrderId());
        //需要防止在修改的时候 被别的线程所修改
        updateWrapper.eq(Order::getOrderStatus,order.getOrderStatus());
        updateWrapper.set(Order::getOrderStatus,updateOrderStatus);
        updateWrapper.set(Order::getPayType,order.getPayType());
        int update = this.orderMapper.update(null, updateWrapper);
        return update > 0;
    }
```

4.4 测试

5. 支付回调

> *在上面生成微信支付二维码的时候，有一个通知回调地址，当用户扫码在手机支付完成之后，服务端如何知道用户支付完成了呢?*
>
> *很明显，微信是知道的，这时候微信的服务端就会主动调用我们设置的回调地址，这个回调地址是我们需要实现的一个接口，微信会把支付的相关信息回传给这个接口。*
>
> *在回调接口中，我们需要处理用户的订单，将其标识为已支付，并且将课程变更为已购买状态，同时告诉微信，订单处理完成，那么整个支付流程就走完了。*

这里有个小知识点：万一微信回调接口的时候，我们的服务器正好挂了，那么是不是会出现，用户付钱了，但是并没有购买课程成功呢?

> *这个知识点就是，微信如果没有收到成功的返回值，那么会一直重复请求接口，在通知一直不成功的情况下，微信总共会发起多次通知，通知频率为 15s/15s/30s/3m/10m/20m/30m/30m/30m/60m/3h/3h/3h/6h/6h — 总计 24h4m*
>
> *这种也有个专业说法，称为指数退避重试*

因为这种机制的存在，如果我们的接口响应慢，或者网络抖动造成微信不能及时收到回执，那么微信会重复调用，所以我们一定要保证接口的幂等性

> *有同学可能会有疑问：如果24h4m，我们的服务器一直没有启动，那么这个支付信息是不是就丢了，提供客服服务，用户提供订单号，我们可以使用订单去微信方进行此订单的实际支付情况。*
>
> *在一些大型系统中，会提供一个状态为正在支付，正在支付的订单，不会取消，会定时清查这批状态的订单，去微信进行验证，如果确实支付了，进行支付成功处理。*

5.1 接口说明

请求路径: /api/order/notify

请求参数: 微信传递的参数，xml格式，使用string接收即可，需要根据文档进行数据解密

返回值: 微信要求的返回值

```
{
    "code": "SUCCESS",
    "message": "成功"
}
```

5.2 业务逻辑

1. 解密参数，获取微信传递的参数
    1. 是否支付成功
    2. 商户订单号
    3. 支付时间等
2. 如果支付成功，获取订单号，查询订单详情
3. 修改订单状态
4. 将对应的课程添加到用户已购买列表（如果已购买，原有基础上增加过期时长）
5. 获取优惠券信息，如果用户使用，将其优惠券标识为已使用
6. 记录流水信息（微信回调返回的所有信息，用于对账）

5.3 涉及到的数据库

名	类型	长度	小数点	不是 n	虚拟	键	注释
id	bigint	0	0	☑	☐	🔑 1	
order_id	varchar	255	0	☑	☐		商户订单号
transaction_id	varchar	255	0	☑	☐		微信支付订单号
pay_type	tinyint	0	0	☑	☐		支付类型
user_id	bigint	0	0	☑	☐		用户id
pay_info	varchar	1024	0	☑	☐		支付信息

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

import lombok.Data;
```

```
@Data
public class OrderTrade {
    private Long id;
    private String orderId;
    private String transactionId;
    private Integer payType;
    private Long userId;
    private String payInfo;
}
```

5.4 编码

5.4.1 Controller

```
@PostMapping("notify")
    public String notifyOrder(@RequestBody String xmlData){
        System.out.println("notify 数据: "+xmlData);
        CallResult callResult = orderService.notifyOrder(xmlData);
        if (callResult.isSuccess()){
            return WxPayNotifyResponse.success("成功");
        }
        return WxPayNotifyResponse.fail("失败");
    }
```

5.4.2 Service

```
CallResult notifyOrder(String xmlData);
```

```
@Override
    public CallResult notifyOrder(String xmlData) {
        OrderDomain orderDomain =
orderDomainRepository.createDomain(null);
        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return orderDomain.notifyOrder(xmlData);
            }
        });
    }
```

5.4.3 Domain

```
public CallResult<Object> notifyOrder(String xmlData) {
        WxPayDomain wxPayDomain = new
WxPayDomain(this.orderDomainRepository.wxPayConfiguration);
        try {
            WxPayOrderNotifyResult notifyResult  =
wxPayDomain.getWxPayService().parseOrderNotifyResult(xmlData);
            String returnCode = notifyResult.getReturnCode();
            if ("SUCCESS".equals(returnCode)){
                log.info(JSON.toJSONString(notifyResult));
                String orderId = notifyResult.getOutTradeNo();
                String transactionId = notifyResult.getTransactionId();
                Order order =
this.orderDomainRepository.findOrderByOrderId(orderId);
                if (order == null){
                    return
CallResult.fail(BusinessCodeEnum.ORDER_NOT_EXIST.getCode(),"order not
exist");
                }
                order.setOrderStatus(OrderStatus.PAYED.getCode());
                order.setPayStatus(PayStatus.PAYED.getCode());
                order.setPayTime(System.currentTimeMillis());

 this.orderDomainRepository.updateOrderStatusAndPayStatus(order);
                //添加支付信息
                OrderTrade orderTrade =
this.orderDomainRepository.findOrderTrade(orderId);
                if (orderTrade != null){

 orderTrade.setPayInfo(JSON.toJSONString(notifyResult));

 this.orderDomainRepository.updateOrderTrade(orderTrade);
                }else{
                    orderTrade = new OrderTrade();

 orderTrade.setPayInfo(JSON.toJSONString(notifyResult));
                    orderTrade.setOrderId(orderId);
                    orderTrade.setUserId(order.getUserId());
                    orderTrade.setPayType(order.getPayType());
                    orderTrade.setTransactionId(transactionId);

 this.orderDomainRepository.saveOrderTrade(orderTrade);
                }
```

```
                //添加课程

 this.orderDomainRepository.createUserCourseDomain(null).saveUserCourse(
order);
                Long couponId = order.getCouponId();
                if (couponId > 0){
                    UserCoupon userCoupon =
this.orderDomainRepository.createCouponDomain(null).findUserCouponByUser
Id(order.getUserId(),couponId);
                    if (userCoupon != null){
                        userCoupon.setStatus(1);

 this.orderDomainRepository.createCouponDomain(null).updateUserCoupon(us
erCoupon);
                    }
                }
                return CallResult.success();
            }
            log.error("notifyOrder error:
{}",notifyResult.getReturnMsg());
            return CallResult.fail();
        } catch (WxPayException e) {
            e.printStackTrace();
            return CallResult.fail();
        }
    }
```

```
@Autowired
    private UserCourseDomainRepository userCourseDomainRepository;
  @Resource
    private OrderTradeMapper orderTradeMapper;
public void updateOrderTrade(OrderTrade orderTrade) {
        this.orderTradeMapper.updateById(orderTrade);
    }

    public void saveOrderTrade(OrderTrade orderTrade) {
        this.orderTradeMapper.insert(orderTrade);
    }

    public UserCourseDomain createUserCourseDomain(UserCourseParam
userCourseParam) {
        return userCourseDomainRepository.createDomain(userCourseParam);
    }
```

CouponDomain:

```
public void updateUserCoupon(UserCoupon userCoupon) {
        this.couponDomainRepository.updateCouponStatus(userCoupon);
    }
```

UserCourseDomain:

```
public void saveUserCourse(Order order) {
        Long courseId = order.getCourseId();
        Long userId = order.getUserId();
        UserCourse course =
this.userCourseDomainRepository.findUserCourseByUserIdAndCourseId(userId
,courseId);
        if (course == null){
            course = new UserCourse();
            course.setCourseId(courseId);
            course.setUserId(userId);
            course.setCreateTime(System.currentTimeMillis());
            course.setExpireTime(System.currentTimeMillis() +
order.getExpireTime() * 24 * 60 * 60 * 1000L);
            course.setStudyCount(0);
            this.userCourseDomainRepository.saveUserCourse(course);
        }else{
            Long expireTime = course.getExpireTime();
            long currentTimeMillis = System.currentTimeMillis();
            if (currentTimeMillis >= expireTime){
                expireTime = currentTimeMillis;
            }
            course.setExpireTime(expireTime + order.getExpireTime() * 24
* 60 * 60 * 1000L);
            this.userCourseDomainRepository.updateUserCourse(course);
        }
    }
```

```
public Integer countUserCourseByUserId(Long userId, List<Long>
courseIdList, long currentTime) {
        LambdaQueryWrapper<UserCourse> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.in(UserCourse::getCourseId,courseIdList);
        queryWrapper.eq(UserCourse::getUserId,userId);
        queryWrapper.gt(UserCourse::getExpireTime,currentTime);
        return this.userCourseMapper.selectCount(queryWrapper);
    }
```

```
    public void saveUserCourse(UserCourse course) {
        this.userCourseMapper.insert(course);
    }

    public void updateUserCourse(UserCourse course) {
        this.userCourseMapper.updateById(course);
    }
```

5.5 测试

支付完成之后，微信会回调公网接口：http://pay.mszlu.com/api/order/notify

我们本地的接口 并收不到，这时候我提供了http://pay.mszlu.com/api/order/all/notify

这个接口，使用postman进行访问，可以得到微信回调的信息，

拿到信息，直接postman发起请求访问：http://www.mszlu.com/api/order/notify，本地就会收到回调，完成支付流程，这样来间接进行测试。

作业

有点问题：

1. 支付回调 加了事务，支付回调 一旦有了问题 能回滚吗？用户已经付钱
2. 收到支付信息后，可以将支付的结果 发送到mq中，mq持久化，同步发送
    1. mq如果你消费不成功 可以进行重复消费 直到成功
    2. mq也可以增大系统的吞吐量
3. 我们信不信任微信？安全问题，收到了微信发送的回调信息，有没有想过，万一不是微信发的怎么办？或者说信息有误怎么办？收到回调信息之后，再次去微信进行订单查询，确保订单的结果是正确的

6. 查询订单

> *支付完成后，页面并不会跳转，那么前端是如何知道用户扫码支付了呢？*
>
> *答案是：不停的请求查询订单接口，确定订单的状态，如果支付完成，自动跳转*

6.1 API

请求路径: /api/order/findOrder

请求参数: {"orderId":"123123"}

返回值:

```
```

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

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

import java.math.BigDecimal;

@Data
public class OrderViewModel {

    private String orderId;
    private CourseViewModel course;
    private BigDecimal oAmount;
    private Integer orderStatus;
    private Integer payType;
    private Integer payStatus;
    private String createTime;
    private String expireTime;
    private BigDecimal couponAmount;

    private String userName;

    private String payOrderId;
    private String payTime;

}
```

6.2 编码

6.2.1 Controller

```
  @PostMapping("findOrder")
    public CallResult findOrder(@RequestBody OrderParam orderParam){
        return orderService.findOrder(orderParam);
    }
```

6.2.2 Service

```
CallResult findOrder(OrderParam orderParam);
```

```
@Override
    public CallResult findOrder(OrderParam orderParam) {
        OrderDomain orderDomain =
orderDomainRepository.createDomain(orderParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return orderDomain.findOrder();
            }
        });
    }
```

6.2.3 Domain

```
public CallResult<Object> findOrder() {
        String orderId = this.orderParam.getOrderId();
        if (StringUtils.isEmpty(orderId)){
            return
CallResult.fail(BusinessCodeEnum.ORDER_NOT_EXIST.getCode(),BusinessCodeE
num.ORDER_NOT_EXIST.getMsg());
        }
        Order order =
this.orderDomainRepository.findOrderByOrderId(orderId);
        if (order == null){
            return
CallResult.fail(BusinessCodeEnum.ORDER_NOT_EXIST.getCode(),BusinessCodeE
num.ORDER_NOT_EXIST.getMsg());
        }
```

```
        OrderViewModel orderViewModel = new OrderViewModel();
        orderViewModel.setOrderId(order.getOrderId());
        CourseViewModel courseViewModel =
this.orderDomainRepository.createCourseDomain(null).findCourseViewModel(
order.getCourseId());
        orderViewModel.setCourse(courseViewModel);
        orderViewModel.setOAmount(order.getOrderAmount());
        orderViewModel.setOrderStatus(order.getOrderStatus());
        orderViewModel.setPayStatus(order.getPayStatus());
        orderViewModel.setPayType(order.getPayType());
        orderViewModel.setCreateTime(new
DateTime(order.getCreateTime()).toString("yyyy-MM-dd HH:mm:ss"));
        orderViewModel.setExpireTime(new DateTime(order.getCreateTime()
+ order.getExpireTime()*24*60*60*1000).toString("yyyy-MM-dd HH:mm:ss"));
        Long couponId = order.getCouponId();
        if (couponId <= 0){
            orderViewModel.setCouponAmount(new BigDecimal(0));
        }else{
            Coupon coupon =
this.orderDomainRepository.createCouponDomain(null).findCouponById(coupo
nId);
            BigDecimal price = coupon.getPrice();
            orderViewModel.setCouponAmount(price);
        }
        return CallResult.success(orderViewModel);
    }
```

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

import lombok.Data;

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

@Data
public class CourseViewModel {
    private Long id;
    private String courseName;
    private String courseDesc;
    private BigDecimal coursePrice;
    private BigDecimal courseZhePrice;
    private Integer orderTime;
    private Integer studyCount;
    private List<Long> subjectIdList;
```

```
    private List<SubjectModel> subjectList;
    private SubjectModel subjectInfo;
    //0 未购买 1 已购买
    private Integer buy;
    private String expireTime;
    private String imageUrl;

    //用户名称
    private String userName;




}
```

CourseDomain:

```
public CourseViewModel findCourseViewModel(Long courseId) {
        Course course =
this.courseDomainRepository.findCourseById(courseId);
        return copyViewModel(course);
    }
    public CourseViewModel copyViewModel(Course course){
        CourseViewModel courseViewModel = new CourseViewModel();
        courseViewModel.setId(course.getId());
        courseViewModel.setCourseDesc(course.getCourseDesc());
        courseViewModel.setCourseName(course.getCourseName());
        courseViewModel.setCoursePrice(course.getCoursePrice());
        courseViewModel.setCourseZhePrice(course.getCourseZhePrice());
        courseViewModel.setOrderTime(course.getOrderTime());
        courseViewModel.setImageUrl(course.getImageUrl());
        List<SubjectModel> subjectModelList =
courseDomainRepository.createSubjectDomain(null).findSubjectListByCourse
Id(course.getId());
        courseViewModel.setSubjectList(subjectModelList);
        return courseViewModel;
    }
```

6.3 测试

7. 订单列表

7.1 API

请求路径: /api/order/orderList

请求参数: {"page":1,"pageSize":20}

返回值:


```
{
    "code":2000000000,
    "message":"default success",
    "result":{
        "pageCount":1,
        "page":1,
        "pageSize":0,
        "size":3,
        "list":[
            {
                "orderId":"1637636396536520487522",
                "course":{
                    "id":3,
                    "courseName":"全课程",
                    "courseDesc":"全课程",
                    "coursePrice":199,
                    "courseZhePrice":108,
                    "orderTime":365,
                    "studyCount":null,
                    "subjectIdList":null,
                    "subjectList":[
                        {
                            "id":1,
                            "subjectName":"初一政治",
                            "subjectGrade":"七年级",
                            "subjectTerm":"上",
                            "status":0,
                            "subjectUnits":null
                        },
```

```
                    {
                        "id":2,
                        "subjectName":"初一语文",
                        "subjectGrade":"七年级",
                        "subjectTerm":"下",
                        "status":0,
                        "subjectUnits":null
                    },
                    {
                        "id":4,
                        "subjectName":"初中英语",
                        "subjectGrade":"七年级",
                        "subjectTerm":"上",
                        "status":0,
                        "subjectUnits":null
                    }
                ],
                "subjectInfo":null,
                "buy":null,
                "expireTime":null,

 "imageUrl":"https://gimg2.baidu.com/image_search/src=http%3A%2F%2Fpic19
.nipic.com%2F20120225%2F9165552_152408494000_2.jpg&refer=http%3A%2F%2Fpi
c19.nipic.com&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=jpeg?
sec=1634921537&t=6219c2c069d33cc35ac8f31ef411dc24",
                "userName":null
            },
            "orderStatus":1,
            "payType":1,
            "payStatus":1,
            "createTime":"2021-11-23 10:59:56",
            "expireTime":"2021-12-10 11:40:25",
            "couponAmount":0,
            "userName":null,
            "payOrderId":null,
            "payTime":null,
            "oamount":0.01
        },
        {
            "orderId":"163767439765484398752 2",
            "course":{
                "id":6,
                "courseName":"七年级道法上",
                "courseDesc":"七年级道法上",
                "coursePrice":199,
```

```
            "courseZhePrice":99,
            "orderTime":365,
            "studyCount":null,
            "subjectIdList":null,
            "subjectList":[
                {
                    "id":7,
                    "subjectName":"道法",
                    "subjectGrade":"七年级",
                    "subjectTerm":"上",
                    "status":0,
                    "subjectUnits":null
                }
            ],
            "subjectInfo":null,
            "buy":null,
            "expireTime":null,
            "imageUrl":"",
            "userName":null
        },
        "orderStatus":2,
        "payType":1,
        "payStatus":2,
        "createTime":"2021-11-23 21:33:17",
        "expireTime":"2021-12-10 22:13:46",
        "couponAmount":0,
        "userName":null,
        "payOrderId":null,
        "payTime":null,
        "oamount":0.01
    },
    {
        "orderId":"163767900667312750752 2",
        "course":{
            "id":7,
            "courseName":"七年级道法下",
            "courseDesc":"七年级道法下",
            "coursePrice":1999,
            "courseZhePrice":99,
            "orderTime":365,
            "studyCount":null,
            "subjectIdList":null,
            "subjectList":[
                {
                    "id":8,
```

```
                            "subjectName":"道法",
                            "subjectGrade":"七年级",
                            "subjectTerm":"下",
                            "status":0,
                            "subjectUnits":null
                        }
                    ],
                    "subjectInfo":null,
                    "buy":null,
                    "expireTime":null,
                    "imageUrl":"",
                    "userName":null
                },
                "orderStatus":2,
                "payType":1,
                "payStatus":2,
                "createTime":"2021-11-23 22:50:06",
                "expireTime":"2021-12-10 23:30:35",
                "couponAmount":0,
                "userName":null,
                "payOrderId":null,
                "payTime":null,
                "oamount":0.01
            }
        ]
    },
    "success":true
}
```


7.2 编码

7.2.1 Controller

```
@PostMapping(value = "orderList")
    public CallResult orderList(@RequestBody OrderParam orderParam){
        return orderService.orderList(orderParam);
    }
```

7.2.2 Service

```
CallResult orderList(OrderParam orderParam);
```

```
 @Override
    public CallResult orderList(OrderParam orderParam) {
        OrderDomain orderDomain =
this.orderDomainRepository.createDomain(orderParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return orderDomain.orderList();
            }
        });
    }
```

7.2.3 Domain

```
public CallResult<Object> orderList() {
        int page = this.orderParam.getPage();
        int pageSize = this.orderParam.getPageSize();
        Long userId = UserThreadLocal.get();
        Page<Order> orderPage =
this.orderDomainRepository.orderList(userId,
OrderStatus.CANCEL.getCode(),page,pageSize);
        List<OrderViewModel> orderViewModelList = new ArrayList<>();
        for (Order order : orderPage.getRecords()){
            OrderViewModel orderViewModel = new OrderViewModel();
            orderViewModel.setOrderId(order.getOrderId());
            CourseViewModel courseViewModel =
this.orderDomainRepository.createCourseDomain(null).findCourseViewModel(
order.getCourseId());
            orderViewModel.setCourse(courseViewModel);
            orderViewModel.setOAmount(order.getOrderAmount());
            orderViewModel.setOrderStatus(order.getOrderStatus());
            orderViewModel.setPayStatus(order.getPayStatus());
            orderViewModel.setPayType(order.getPayType());
            orderViewModel.setCreateTime(new
DateTime(order.getCreateTime()).toString("yyyy-MM-dd HH:mm:ss"));
            orderViewModel.setExpireTime(new
DateTime(order.getCreateTime() +
order.getExpireTime()*24*60*60*1000).toString("yyyy-MM-dd HH:mm:ss"));
```

```
            Long couponId = order.getCouponId();
            if (couponId <= 0){
                orderViewModel.setCouponAmount(new BigDecimal(0));
            }else{
                Coupon coupon =
this.orderDomainRepository.createCouponDomain(null).findCouponById(coupo
nId);
                BigDecimal price = coupon.getPrice();
                orderViewModel.setCouponAmount(price);
            }
            orderViewModelList.add(orderViewModel);
        }
        ListPageModel listPageModel = new ListPageModel();
        int total = (int) orderPage.getTotal();
        listPageModel.setSize(total);
        listPageModel.setPageCount(orderPage.getPages());
        listPageModel.setPage(page);
        listPageModel.setList(orderViewModelList);
        return CallResult.success(listPageModel);
    }
```

```
 public Page<Order> orderList(Long userId, int orderStatus,int
currentPage,int pageSize) {
        LambdaQueryWrapper<Order> queryWrapper = Wrappers.lambdaQuery();
        queryWrapper.eq(Order::getUserId,userId);
        queryWrapper.ne(Order::getOrderStatus,orderStatus);
        Page<Order> page = new Page<>(currentPage,pageSize);
        return this.orderMapper.selectPage(page,queryWrapper);
    }
```

7.3 测试