校园商铺平台项目：

项目的骨架是基于SSM框架，即Spring、SpringMVC和MyBatis实现封装良好的通用模块。分为三大系统模块：前端展示系统、店家管理系统和超级管理员系统。前端展示系统主要负责：头条展示、店铺类别展示、区域展示、店铺信息展示（店铺列表、店铺详情、店铺查询）、商品展示（商品列表、商品查询、商品详情）；店家管理系统主要负责：Local账号的维护、微信登陆（微信账号的维护）、店铺信息的维护（登陆进去）、权限验证（验证是否店家以及是否有这个权限去修改商品的信息）、商品类别的维护；超级管理员系统主要负责：头条管理、类别管理、区域管理、账号管理、店铺信息管理。

责任描述：

完成SSM框架的搭建，完成前端展示系统和店家管理员系统的开发、测试、展示、备份。前端部分是SUI Mobile迅速搭建响应式界面，项目的前端部分采用html（而非JSP），与Java解耦，能够方便的将页面复用到别的语言的后台框架中，另外还通过微信开发者以及Eclipse远程调试方便在PC上调试放在远程服务器上的微信项目；后端除了SSM框架之外，还利用了Redis缓存，Thumnailator图片处理，MySQL主从库和定期备份等丰富的内容。在部署运营方面，是在华为云上面搭建自己的线上环境（华为云服务器、XShell、Centos），实现项目部署以及定时的数据备份。

1.项目的骨架是基于SSM框架，即Spring、SpringMVC和MyBatis。封装良好的通用模块非常重要。该项目是一个从前端到后端，从雏形到推广的完整项目，前端部分是SUI Mobile迅速搭建响应式界面，项目的前端部分采用html（而非JSP），与Java解耦，能够方便的将页面复用到别的语言的后台框架中，另外还通过微信开发者以及 Eclipse 远程调试方便在PC上调试放在远程服务器上的微信项目；后端除了SSM框架之外，还利用了Redis缓存，Thumnailator图片处理，MySQL主从库和定期备份等丰富的内容。在部署运营方面，是在华为云上面搭建自己的线上环境（华为云服务器、XShell、Centos），如何实现项目部署以及定时的数据备份。在短时间内实现项目。

彩蛋：如何利用

2.技术储备：（1）掌握Java Web基础和Maven构建项目

（2）掌握Spring框架基础

3.环境准备：

（1） JDK8提供Java基础类库的支持

（2） Maven3.3.9对项目的jar包进行管理

（3） MySQL Server存储项目中的数据

（4） Chrome浏览器对项目进行展示演示

（5） Tomcat8服务器运行Java项目

（6） IDE选用Eclipse

4.大致安排：

（1）场景推动，从简设计实体类及表

（2）按照功能模块划分，从Dao层开始，再到Controller层的设计，最后到前端开发

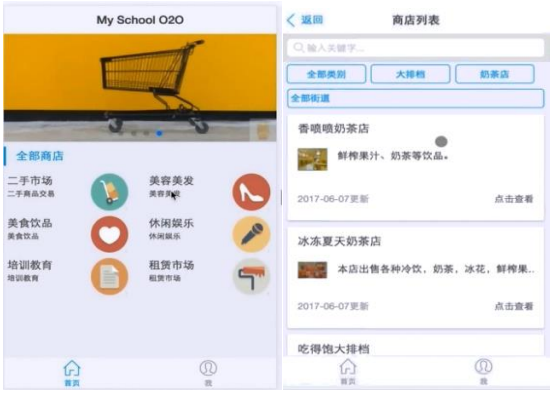
（3）串联知识点

5. 系统功能的目标

（1）前端展示系统

主要负责：头条展示、店铺类别展示、区域展示、店铺信息展示（店铺列表、店铺详情、店铺查询）、商品展示（商品列表、商品查询、商品详情）

图一是**头条展示和店铺类别展示**，点击进去之后就是**店铺列表信息的展示（页面上方是店铺子类别的展示）**



可以在其中输入想要查询的店铺，点击进这个奶茶店，进入**商品展示模块（展示的商品的详情）**

红框部分展示的是店铺下面的商品类别，下拉是店铺中的商品列表

随便输入一个商品的名字点击进去就可以查看商品的信息

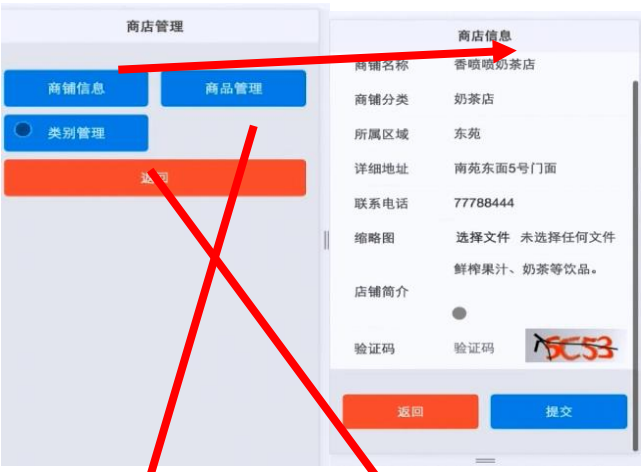
（2）店家管理系统

主要负责：Local 账号的维护、微信登陆（微信账号的维护）、店铺信息的维护（登陆进去）、权限验证（验证你是不是这个店家以及是不是有这个权限去修改商品的信息）、商品类别的

维护

登陆进店铺列表中

进入香喷喷奶茶店



包括商铺信息的管理，可以修改商铺信息，还有商品的管理（控制商品的上下架），还有类别的管理（对店铺下面的商品类别做增删改查的操作）

（3）超级管理员系统

头条管理：就是首页的滚动条

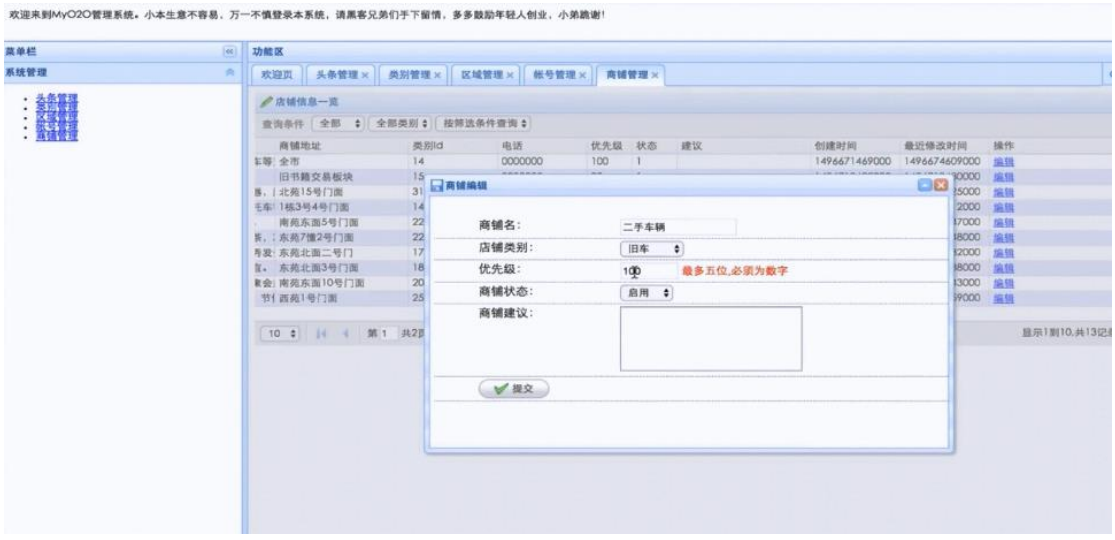
类别管理：管理首页的大类以及二级小类

区域管理

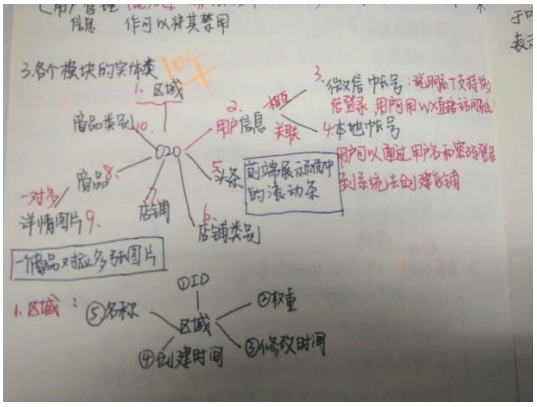
账号管理

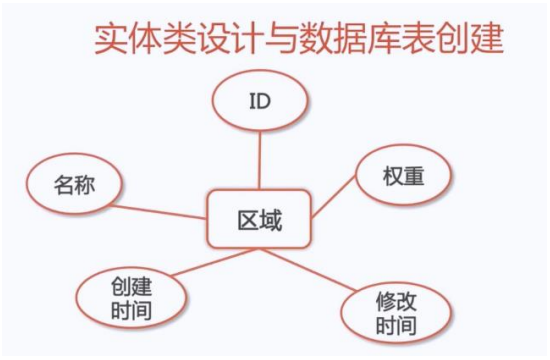
如果账号在里面做了非法操作可以将其禁用

店铺信息管理（只有取得审核通过的店铺才能进行展示给顾客使用，还可以控制店铺的排序《优先级越高，在前端将会显示更优先的排位》）

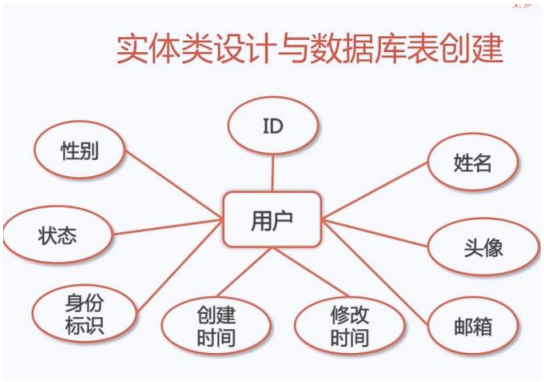


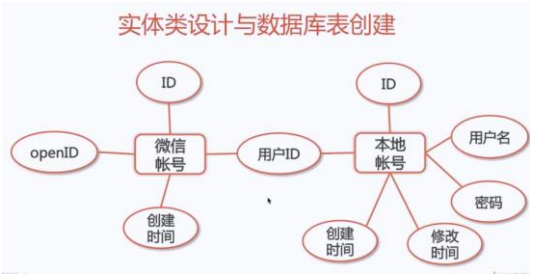
6. 系统的实体类





在 Eclipse 中创建实体类，接着在 MySQL 中创建对应的表（表的引擎采用 InnoDB,因为要经常修改，I 的写的性能略高于 M，M 的读的性能远大于 I，如果要支持事务的化必须使用 I）

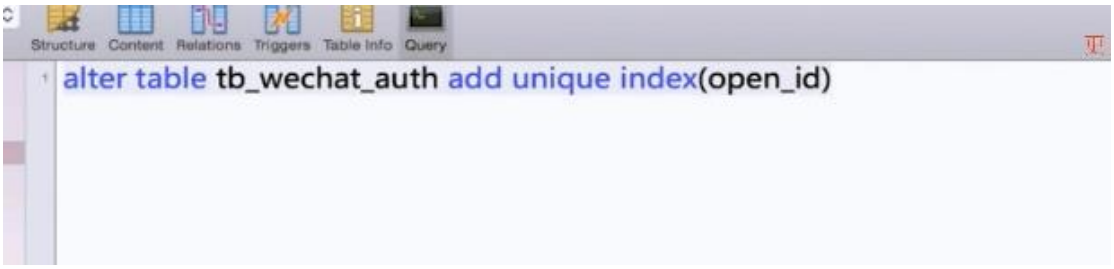




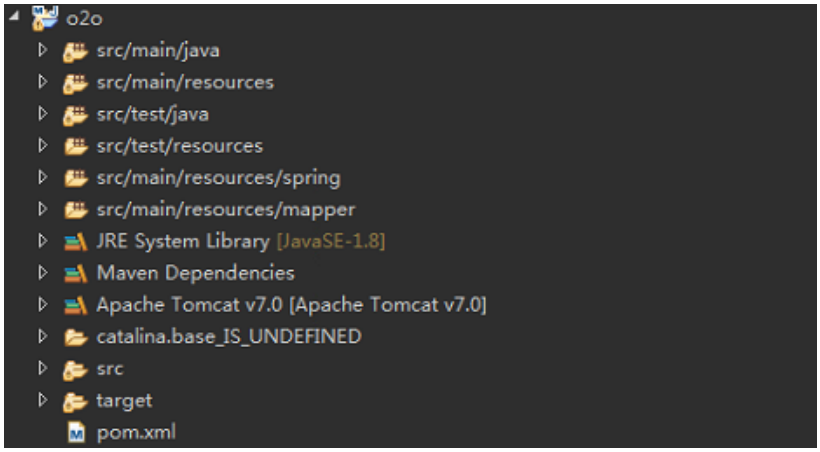
第一次用到主外键和索引（微信账号和用户账号）关联表，

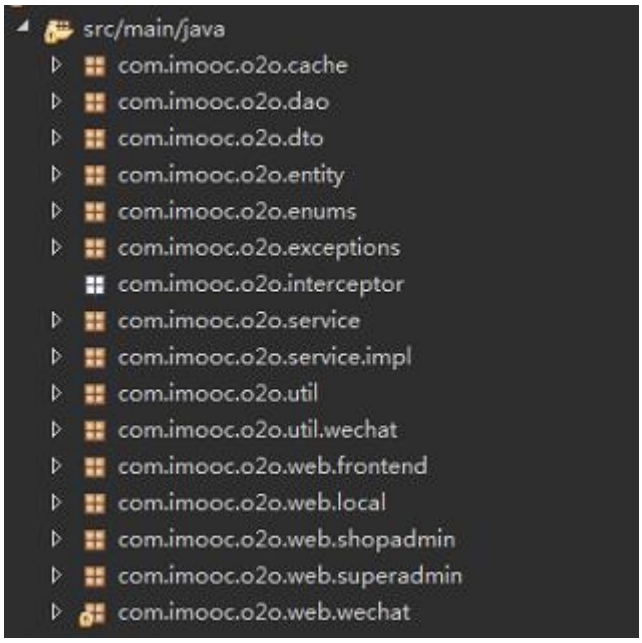
```
create table `tb_local_auth`(
   `local_auth_id` int(10) NOT NULL AUTO_INCREMENT,
   `user_id` int(10) NOT NULL,
   `username` varchar(128) NOT NULL,
   `password` varchar(128) NOT NULL,
   `create_time`datetime DEFAULT NULL,
   `last_edit_time` datetime DEFAULT NULL,
   PRIMARY KEY( `local_auth_id`),
   UNIQUE KEY `uk_local_profile` (`username`),
   constraint `fk_wechatauth_profile` foreign key(`user_id`) references `tb_person_info`(`user_id`)
)
```

这里用了唯一索引（通过这个索引来获取微信的信息，提高查询效率）
索引就相当于字典里面的偏旁和部首，创建索引可以加快查询速度，但是创建索引也是需要维护的



7.配置 Maven 及 SSM 各项配置（2-6）



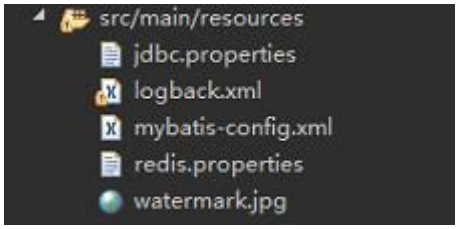


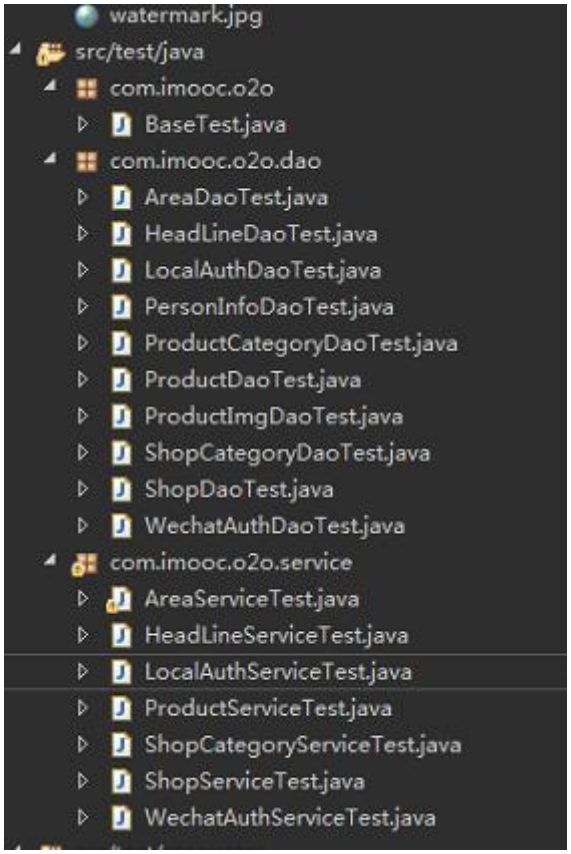
Dao 和数据库打交道，或者文件的读写操作或者 redis 的读写操作，不需要 impl 实现，因为用的是 Mybatis，可以在 mapper 中直接实现

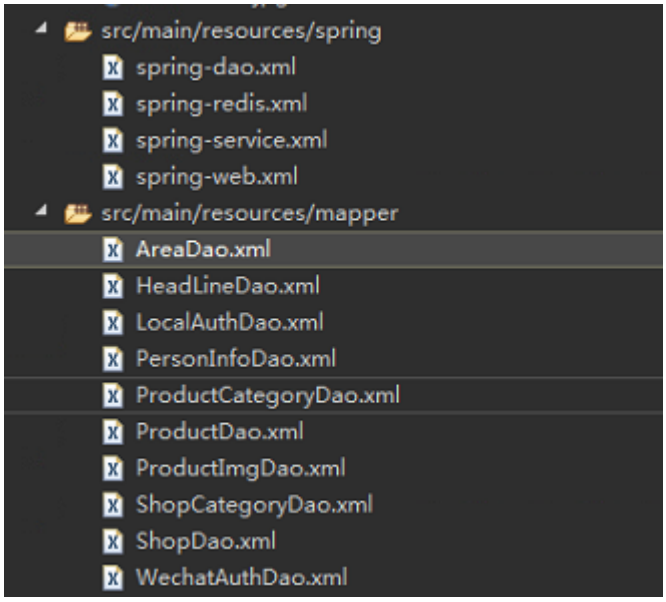
Dto 使用来弥补到的

Web 就是 Controller 层

Service 是业务实现，需要实现，因此需要 impl







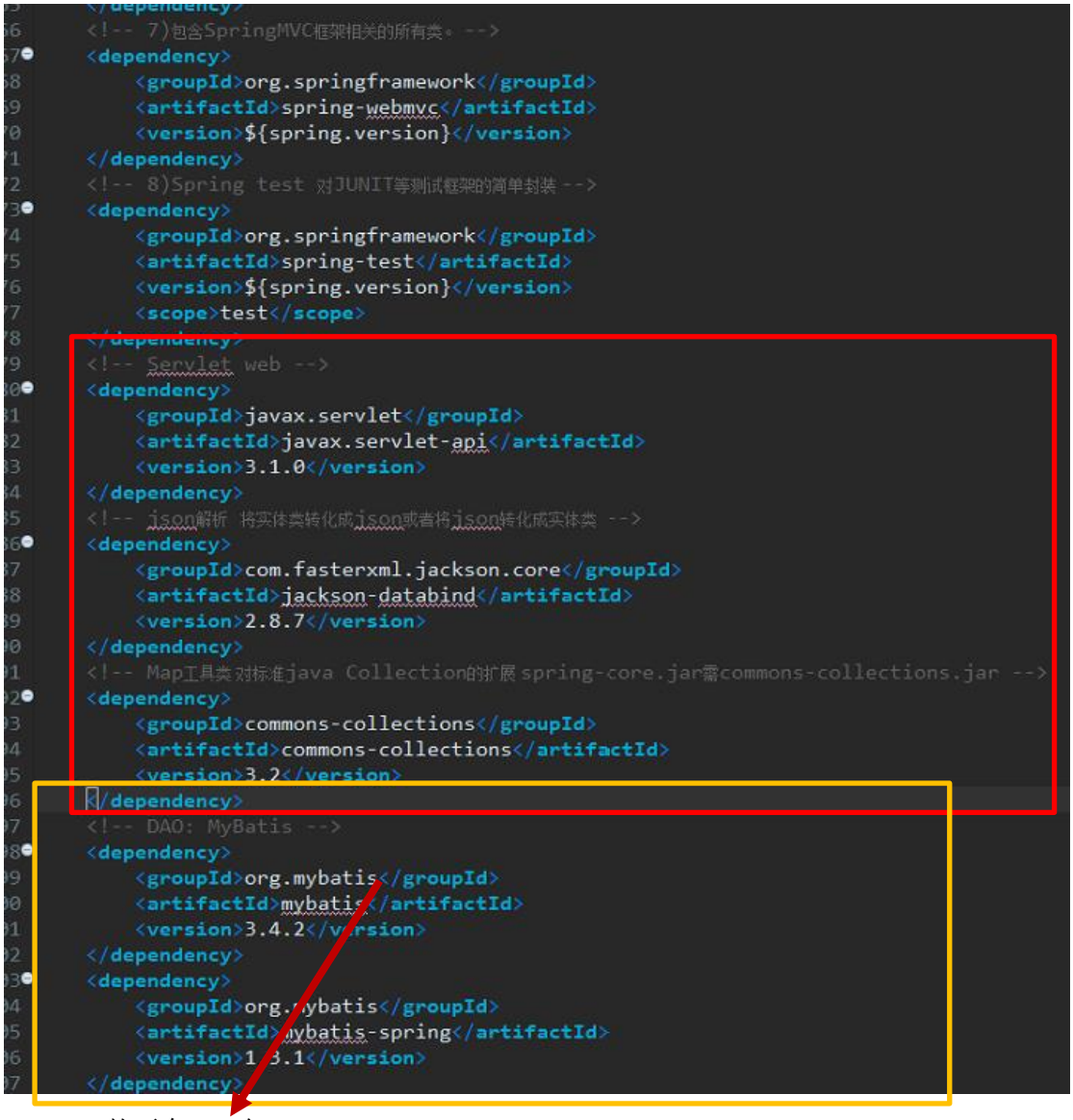
配置 pom.xml，引入 junit（只出现在单元测试中），logback（）

```xml
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/ch.qos.logback/logback-classic -->
  <dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
  </dependency>
```

Spring 中需要的 jar 包

```xml
<properties>
  <spring.version>4.3.7.RELEASE</spring.version>
</properties>
```

```xml
<!-- Spring -->
<!-- 1)包含Spring 框架基本的核心工具类。Spring 其它组件要都要使用到这个包里的类，是其它组件的基本核心 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 2)这个jar 文件是所有应用都要用到的，它包含访问配置文件、创建和管理bean 以及进行Inversion of Control
/ Dependency Injection（IoC/DI）操作相关的所有类。如果应用只需基本的IoC/DI 支持，引入spring-core.jar
及spring-beans.jar 文件就可以了。 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 3)这个jar 文件为Spring 核心提供了大量扩展。可以找到使用Spring ApplicationContext特性时所需的全部类，JDNI
所需的全部类，instrumentation组件以及校验Validation 方面的相关类。 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 4) 这个jar 文件包含对Spring 对JDBC 数据访问进行封装的所有类。 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 5) 为JDBC、Hibernate、JDO、JPA等提供的一致的声明式和编程式事务管理。 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 6)Spring web 包含Web应用开发时，用到Spring框架时所需的核心类，
包括自动载入WebApplicationContext特性的类、Struts与JSF集成类、文件上传的支持类、Filter类和大量工具辅助类。 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
</dependency>
```



Mybatis 的两个 jar 包

```xml
    <!-- 数据库 -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.37</version>
    </dependency>
    <dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>
        <version>0.9.1.2</version>
    </dependency>
    <!-- 图片处理 -->
    <!-- https://mvnrepository.com/artifact/net.coobird/thumbnailator -->
    <dependency>
        <groupId>net.coobird</groupId>
        <artifactId>thumbnailator</artifactId>
        <version>0.4.8</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.github.penggle/kaptcha -->
    <dependency>
        <groupId>com.github.penggle</groupId>
        <artifactId>kaptcha</artifactId>
        <version>2.3.2</version>
    </dependency>
    <dependency>
        <groupId>commons-fileupload</groupId>
        <artifactId>commons-fileupload</artifactId>
        <version>1.3.2</version>
    </dependency>
    <!-- redis客户端:Jedis -->
    <dependency>
        <groupId>redis.clients</groupId>
        <artifactId>jedis</artifactId>
        <version>2.9.0</version>
    </dependency>
</dependencies>
```

逐层实现 SSM，

首先配置 Mysql，在 src/java/resource 下面配置 jdbc.propertires

```
1 jdbc.driver=com.mysql.jdbc.Driver
2 jdbc.url=jdbc:mysql://localhost:3306/o2o?useUnicode=true&characterEncoding=utf8
3 jdbc.username=WnplV/ietfQ=
4 jdbc.password=Q6fs4Ko6Mk7LPup1HZ8ACQ==
5
```

接着配置 Mybatis 配置

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!-- 配置全局属性 -->
    <settings>
        <!-- 使用jdbc的getGeneratedKeys获取数据库自增主键值 -->
        <setting name="useGeneratedKeys" value="true" />

        <!-- 使用列标签替换列别名 默认:true -->
        <setting name="useColumnLabel" value="true" />

        <!-- 开启驼峰命名转换:Table{create_time} -> Entity{createTime} -->
        <setting name="mapUnderscoreToCamelCase" value="true" />
    </settings>
</configuration>
```

列标签和列别名在刷题的时候用到过

接着进行 Spring 的配置

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- 配置整合mybatis过程 -->
    <!-- 1.配置数据库相关参数properties的属性: ${url} -->
    <bean class="com.imooc.o2o.util.EncryptPropertyPlaceholderConfigurer">
        <property name="locations">
            <list>
                <value>classpath:jdbc.properties</value>
                <value>classpath:redis.properties</value>
            </list>
        </property>
        <property name="fileEncoding" value="UTF-8"/>
    </bean>
    <!-- 2.数据库连接池 -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <!-- 配置连接池属性 -->
        <property name="driverClass" value="${jdbc.driver}" />
        <property name="jdbcUrl" value="${jdbc.url}" />
        <property name="user" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />

        <!-- c3p0连接池的私有属性 -->
        <property name="maxPoolSize" value="30" />
        <property name="minPoolSize" value="10" />
        <!-- 关闭连接后不自动commit -->
        <property name="autoCommitOnClose" value="false" />
        <!-- 获取连接超时时间 -->
        <property name="checkoutTimeout" value="10000" />
        <!-- 当获取连接失败重试次数 -->
        <property name="acquireRetryAttempts" value="2" />
    </bean>
```

```xml
    <!-- 3.配置SqlSessionFactory对象 -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 注入数据库连接池 -->
        <property name="dataSource" ref="dataSource" />
        <!-- 配置MyBaties全局配置文件:mybatis-config.xml -->
        <property name="configLocation" value="classpath:mybatis-config.xml" />
        <!-- 扫描entity包 使用别名 -->
        <property name="typeAliasesPackage" value="com.imooc.o2o.entity" />
        <!-- 扫描sql配置文件:mapper需要的xml文件 -->
        <property name="mapperLocations" value="classpath:mapper/*.xml" />
    </bean>

    <!-- 4.配置扫描Dao接口包，动态实现Dao接口，注入到spring容器中 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 注入sqlSessionFactory -->
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
        <!-- 给出需要扫描Dao接口包 -->
        <property name="basePackage" value="com.imooc.o2o.dao" />
    </bean>
</beans>
```

在使用 Mybatis 时需要配置 sqlSessionFactory，创建需要配置数据库连接池的对象

接着配置 Spring 和 Service 相关的操作

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">
    <!--1. 扫描service包下所有使用注解的类型 -->
    <context:component-scan base-package="com.imooc.o2o.service" />

    <!--2. 配置事务管理器 -->
    <bean id="transactionManager"
        class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <!-- 3.注入数据库连接池 -->
        <property name="dataSource" ref="dataSource" />
    </bean>

    <!--4. 配置基于注解的声明式事务 -->
    <tx:annotation-driven transaction-manager="transactionManager" />
</beans>
```

接着配置 web 层（注意 WEB-INF 中的文件默认后缀都是 html）

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">
    <!-- 配置SpringMVC -->
    <!-- 1.开启SpringMVC注解模式 -->
    <mvc:annotation-driven />

    <!-- 2.静态资源默认servlet配置 (1)加入对静态资源的处理: js,gif,png (2)允许使用"/"做整体映射 -->
    <mvc:resources mapping="/resources/**" location="/resources/" />
    <mvc:default-servlet-handler/>

    <!-- 3.定义视图解析器 -->
    <bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/html/"></property>
        <property name="suffix" value=".html"></property>
    </bean>
    <!-- 文件上传解析器 -->
    <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="defaultEncoding" value="utf-8"></property>
        <!-- 1024*1024*20 = 20M -->
        <property name="maxUploadSize" value="20971520"></property>
        <property name="maxInMemorySize" value="20971520"></property>
    </bean>
    <!-- 4.扫描web相关的bean -->
    <context:component-scan base-package="com.imooc.o2o.web" />
</beans>
```

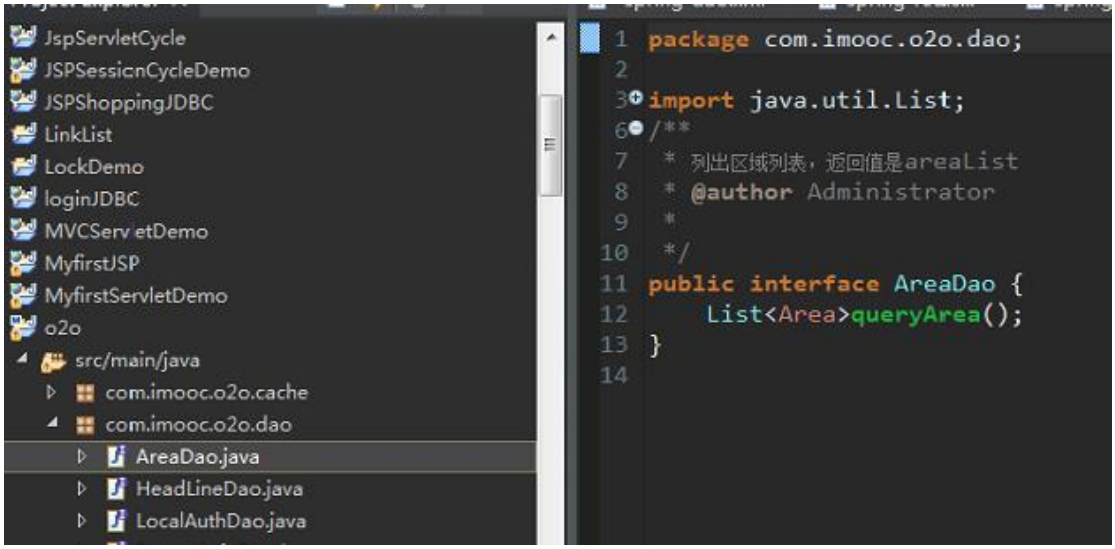
在 web.xml 中把三个 Spring 配置文件整合在一起，并配置 DispatcherServlet 来响应前端的请求

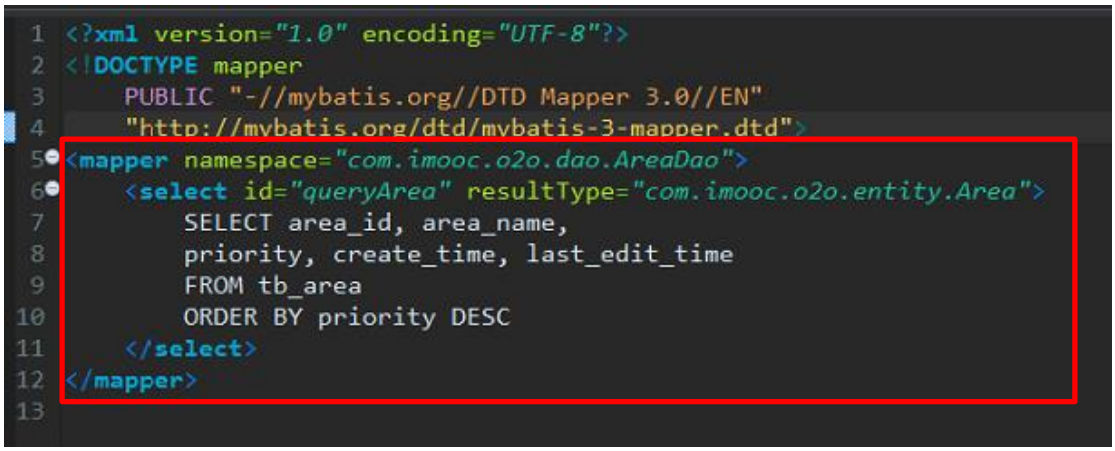
```xml
<servlet>
    <servlet-name>spring-dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring/spring-*.xml</param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>spring-dispatcher</servlet-name>
    <!-- 默认匹配所有的请求 -->
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

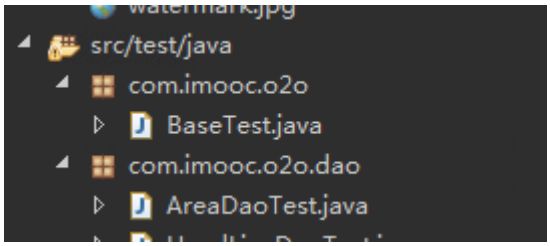
现在就自下而上的完成的 SSM 框架

8.通过代码验证 SSM 框架配置的正确性（选择 Area，它和其它类关系不紧密），自下而上，在 DAO 层创建 AreaDao 接口，由于 Mybatis 不需要 Dao 实现 impl，在 mapper 配置文件 AreaDao.xml 中配置





这就是Mybatis的好处,直接把这些复制到Mysql,在Mysql中添加数据就可以了,namesapce对应的是相应的方法，接着进行单元验证（src/test/java）



```java
package com.imooc.o2o.dao;

import static org.junit.Assert.assertEquals;

public class AreaDaoTest extends BaseTest {
    @Autowired
    private AreaDao areaDao;
    @Test
    public void testQueryArea() {
        List<Area> areaList=areaDao.queryArea();
        assertEquals(2,areaList.size());
    }

}
```

之前创建好了 2 条记录，进行 junit 验证是绿色的就说明没错了

比赛：

背景:云平台为了满足不同租户的需求,提供了一种可随时自助获取、可弹性伸缩的云服务器,即弹性云服务器(Elastic Cloud Server,ECS)。为容纳更多的租户请求、并尽可能提高资源利用率、降低成本,自动化、智能化的资源调度管理系统非常关键。

由于租户对 ECS 实例(虚拟机,VM)请求的行为具有一定规律,可以通过对历史 ECS 实例请求的分析,预测到未来一段时间的 ECS 实例请求,然后对预测的请求分配资源,这样可以找到一个接近最优的分配策略,实现资源最大化利用,同时也能参考预测的结果制定云数据中心的建设计划。

责任描述：

设计一个程序能够精确预测未来某个时间段内的虚拟机的请求情况,并寻找最佳的资源分配方案:对输入的虚拟机历史请求数据进行建模分析以及训练,确定系数给出预测模型,然后对给定预测时间段内不同规格的虚拟机数量进行预测,最后根据预测结果把虚拟机部署到物理服务器上,使得服务器的资源利用率最大化。

华为软件精英挑战赛

1. 比赛问题定义

背景:云平台为了满足不同租户的需求,提供了一种可随时自助获取、可弹性伸缩的云服务器,即弹性云服务器(Elastic Cloud Server,ECS)。为容纳更多的租户请求、并尽可能提高资源利用率、降低成本,自动化、智能化的资源调度管理系统非常关键

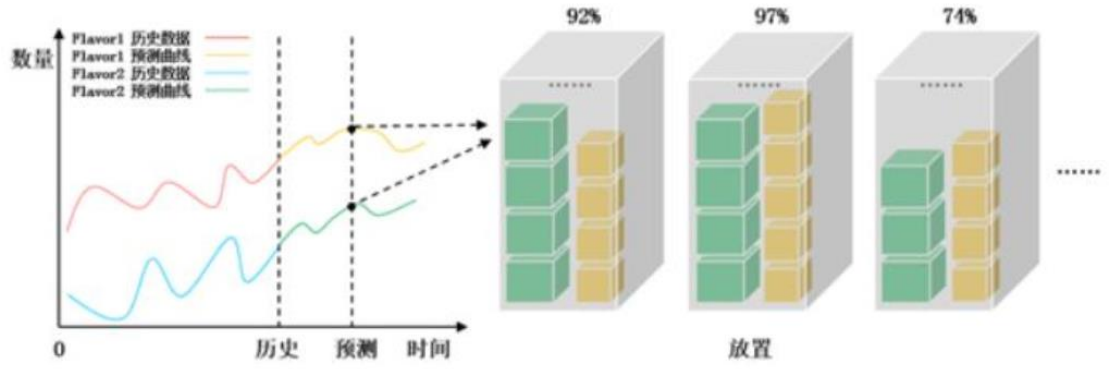


图1 智能预测与资源调度示例

本次赛题基本描述

由于租户对 ECS 实例(虚拟机,VM)请求的行为具有一定规律,可以通过对历史 ECS 实例请求的分析,预测到未来一段时间的 ECS 实例请求,然后对预测的请求分配资源(如图 1 所示),这样可以找到一个接近最优的分配策略,实现资源最大化利用,同时也能参考预测的结果制定云数据中心的建设计划。

本次赛题通用性描述

物理服务器:云数据中心通常由大规模的物理服务器集群所组成,通过虚拟化技术,可以对每个物理节点的资源(如 CPU、内存、硬盘等)进行隔离,使得每台物理机上可以同时容纳

多个虚拟机,这些虚拟机即共享该物理服务器上的所有资源。为了保证每台虚拟机的运行性能,通常物理资源不能“超分”,即每台物理服务器上所有虚拟机的虚拟资源总量不能超过其物
理资源总量。这里假设物理服务器只有一种规格大小。
虚拟机规格:云平台通常预先定义了各种类型虚拟机的规格(flavor),便于租户选择购买。例如,如果租户对计算要求低,而内存要求相对高,可以选择 4 个 vCPU、 16GB 内存的配置。由于不考虑物理资源的“超分”,虚拟机的资源占用可与物理资源进行一比一的映射。当然,不同规格的虚拟机有不同的性能,价格也不一。
资源维度:如上述,虚拟机正常工作需要多个维度的资源同时配合,如 CPU、内存、硬盘、网络等等,每种资源都可能成为瓶颈,并且每种资源分配不合理都可能产生碎片。这里假设只需要考虑单个维度资源的使用,即尽可能最大化每台物理服务器的 CPU 或者内存(Mem)的利用率,但在考虑某个维度资源优化的同时,要保证其他资源不能超分。
历史请求数据:租户在云平台上每申请一台虚拟机都会在后台的数据库产生一条数据,每条数据包含了虚拟机的 ID、虚拟机的规格大小以及创建时间等。如果我们可以获取到云平台在过去一段时间的所有虚拟机请求数据,通过训练这些数据特征,可以预测下一个时间段可能到来的虚拟机请求分布。
比赛程序内容:请你设计一个程序能够精确预测未来某个时间段内的虚拟机的请求情况,并寻找最佳的资源分配方案:对输入的虚拟机历史请求数据进行建模分析以及训练,确定系数给出预测模型,然后对给定预测时间段内不同规格的虚拟机数量进行预测,最后根据预测结果把虚拟机部署到物理服务器上,使得服务器的资源利用率最大化。

最终得分和排名机制

比赛平台会使用多个训练数据集,每个训练数据集又构造出多个测试用例判题,测试用例分为初级、中级、高级三个等级,每个等级的难度主要根据预测的时间长短以及预测的虚拟机规格数量两个指标来区分。每个等级各 3 个用例,共 9 个用例。参赛者对于每个测试用
例都会得到一个百分制分数,使用加权总分(初级权重为 0.2,中级权重为 0.3,高级权重为 0.5)
作为该参赛者的最终得分。
最终排名机制如下:
Step1:最终得分为所有测试用例得分的加权,根据得分高低进行排名;
Step2:最终得分相同的情况,比较所有测试用例的总运行时间,运行时间越短,排名越靠前;
Step3: 如以上均相同的情况,则根据提交的先后顺序区分排名。
特别说明:在比赛初期(练习阶段),比赛平台只放出初级、中级的测试用例各 2 个,故加权后满分为 100 分(初级权重为 0.2,中级权重为 0.3,高级权重为 0.5;下同);在正式比赛阶段,
才会放出高级测试用例(具体发放时间会在网站公告通知),初、中、高级用例各 3 个,加权后
满分为 300 分。练习阶段每支参赛队伍每天最多只能提交 100 次,正式比赛阶段每支参赛队伍每天最多只能提交 5 次。
特别注意:最终成绩以最后一次提交的答案为准。请各位参赛者务必注意。

虚拟机部署到物理服务器上，使得服务器资源利用最大化

用户租赁服务器的服务			物理服务器（有CPU和内存两个维度）
flavor1	1	1024=1G	
flavor2	1	2048=2G	
flavor3	1	4096=4G	
flavor4	2	8192=8G	
flavor5	2	4096=4G	
flavor6	2	8192=8G	
flavor7	4	16384=16G	
flavor8	4	8192=8G	
flavor9	4	16384=16G	
flavor10	8	8192=8G	
flavor11	8	16384=16G	
flavor12	8	32768=32G	
flavour13	8	16384=16G	
flavour14	16	32768=32G	
flavour15	16	65536=64G	

cpu	内存
56G	128G

假设给定的物理服务器的cpu为56核，内存是128核G

因为用户租赁的虚拟机在之前已经被预测出来了，所以我们直接取虚拟机的个数及对应的规格

flavor	5	(120)
flavor	8	(40)
flavor	15	(10)

总的cpu $= 2\times 120 + 4\times 40 + 16\times 10 = 360$

总的内存数 $= 4\times 120 + 8\times 40 + 64\times 10 = 1040$

计算物理服务器的总数

$n_1 = \frac{360}{56} \approx 6.43$

$n_2 = \frac{1040}{128} \approx 8.125$

为存放所有的虚拟机，则至少需要9台物理服务器

方法：先将cpu大的虚拟机放到服务器内，再用cpu小的来填补

[16|64] [56|128] ⇒可放两台flavor15，

第一台物理服务器，放flavor15，

此时内存全部用完，cpu剩24

事实上flavor15全部用完需要5台物理服务器；

对第六台物理服务器，放flavor8 [4|8]，比较 $\{ 56/4=14,\ 128/8=16 \}$，取14

所以第六台放14个flavor8，内存剩16G，

同样第七台情况和第6台一样，剩16G

此时flavor8还剩12台，flavor5还未使用 [2|4]

第8台先将flavor8放完，内存剩32，cpu剩8 $\{ 8/2=4,\ 32/4=8 \}$ ⇒选4

再放4台flavor5

现在还剩116台flavor5，全部放入第9台物理服务器中，剩余cpu=24，剩余内存64.

最后检查剩余内存是否超过了原来内存和cpu的一半，则将最后一台虚拟机服务器删除；若剩余内存小于原来内存和cpu的一半，则我们会向服务器中添加虚拟机，并改变相应预测虚拟机的数量，这样做的作用是：损失一些预测精度来提高服务器的资源利用率。

其他队员的分工：①数据预处理：用中位数来去除异常

②算法预测：用局部加权线性回归预测虚拟机一到两周的台数

③对数据进行装箱：对预测出来的一段时间的虚拟机台数进行合理的分配，将不同种类的虚拟机分配到相应的物理服务器上以达到最优方式