[今日课程大纲]

电商项目介绍

开发环境搭建

完成后台商品查询

[知识点详解]

一.电商项目介绍

1.电商行业的几种模式.

1.1 B2B：企业到企业，商家到商家。代表：阿里巴巴、慧聪网。

1.2 B2C：商家到客户。代表：京东、淘宝商城（B2B2C）。

1.3 C2C：客户到客户。淘宝集市。

1.4 O2O：线上到线下。

2.技术选型

2.1 Spring、SpringMVC、Mybatis

2.2 JSP、JSTL、jQuery、jQuery plugin、EasyUI、KindEditor（富文本编辑器）、CSS+DIV

2.3 Redis（缓存服务器）

2.4 Solr（搜索）

2.5 Dubbo（调用系统服务）

2.6 Mysql

2.7 Nginx（web 服务器）

2.8 jsonp 跨域数据请求格式

2.9 nexus maven 私服

2.10 MyBatis 逆向工程

2.11 HttpClient 使用 java 完成请求及响应的技术.

2.12 MyCat mysql 分库分表技术

3. 开发工具和环境

3.1 Eclipse mars

3.2 Maven 3.3.3

3.3 Tomcat 7.0.79（Maven Tomcat Plugin）

3.4 JDK 1.7

3.5 Mysql 5.7

3.6 Nginx 1.8.0

3.7 Redis 3.0.0

3.8 Win7 操作系统

3.9 Linux(服务器系统)

4. 人员配置

4.1 产品经理：3 人，确定需求以及给出产品原型图。

4.2 项目经理：1 人，项目管理。

4.3 前端团队：5 人，根据产品经理给出的原型制作静态页面。

4.4 后端团队：20 人，实现产品功能。

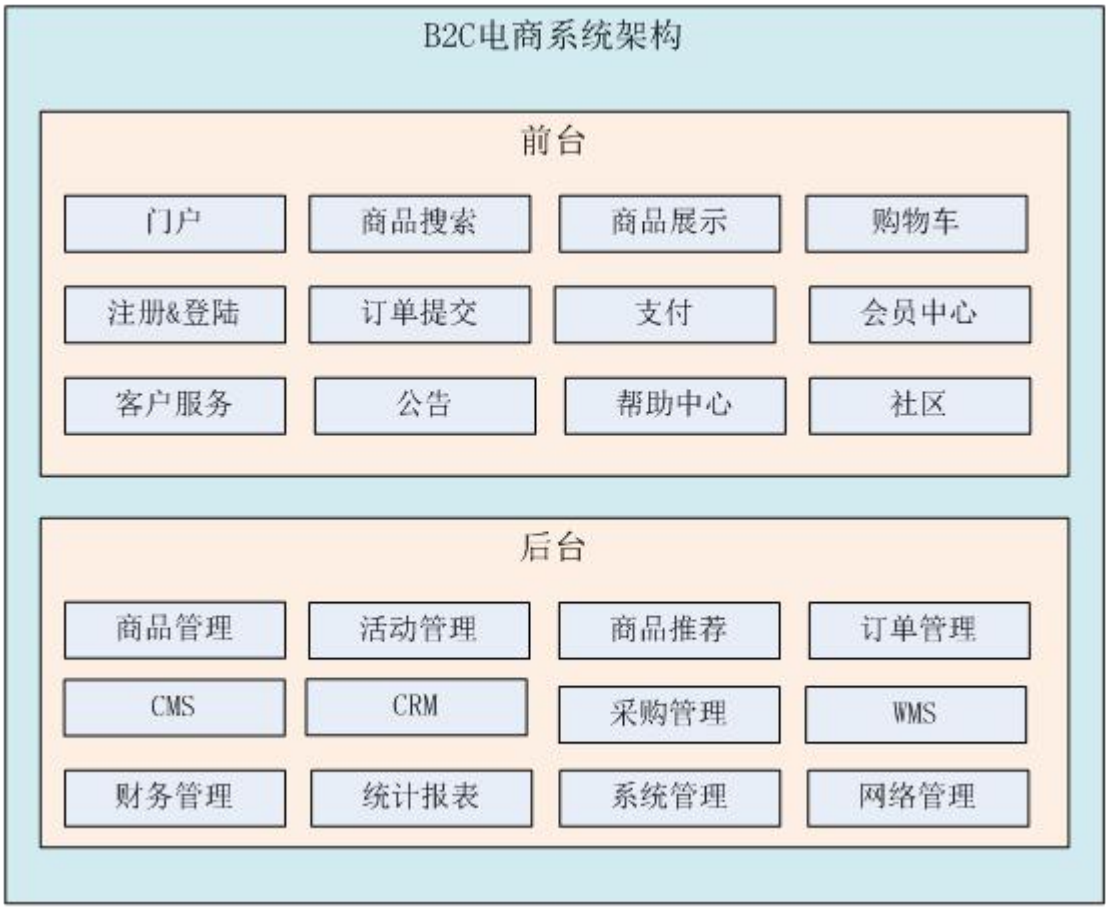
4.5 测试团队：5 人，测试所有的功能。

4.7 运维团队：3 人，项目的发布以及维护。

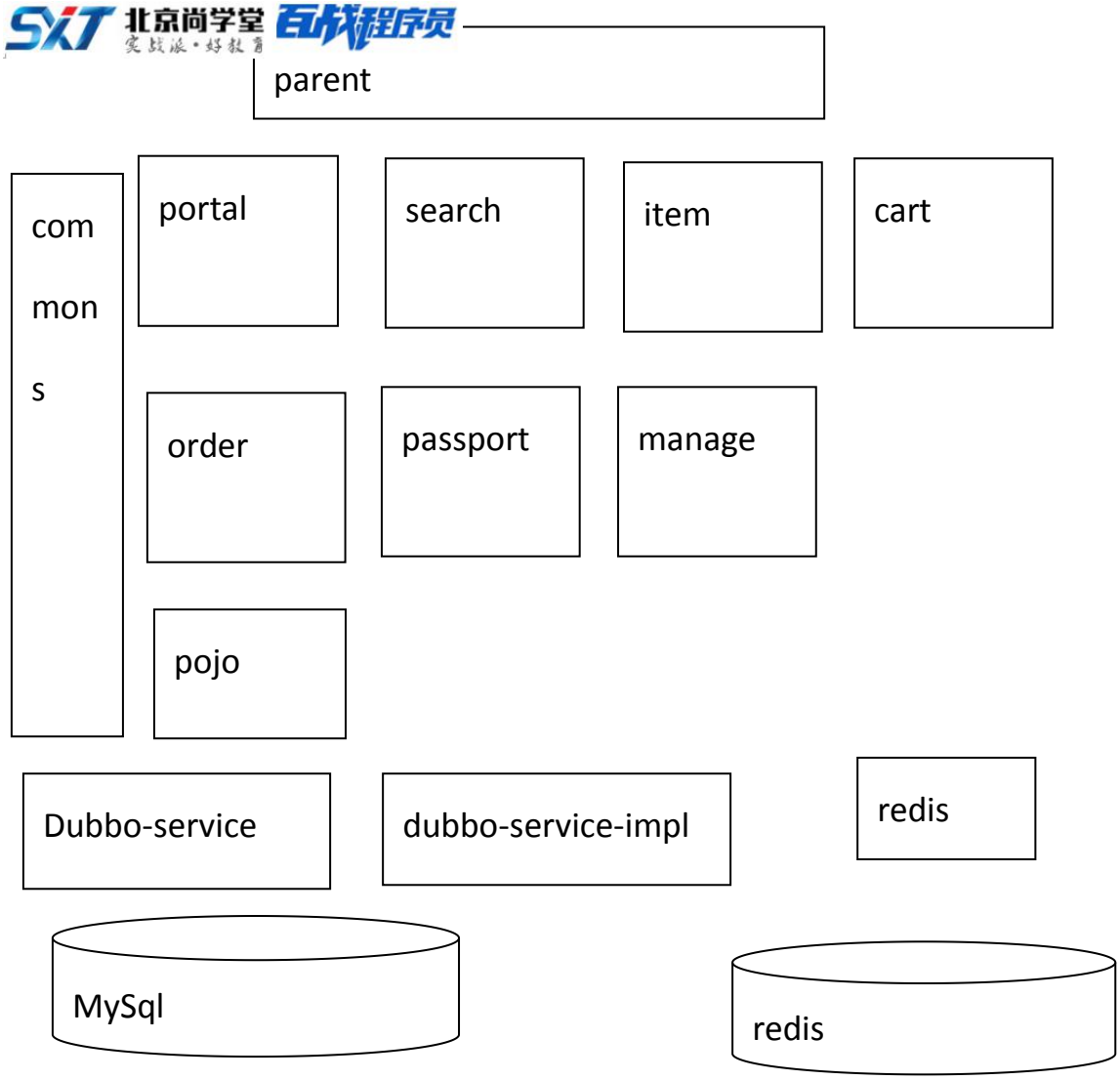
5.项目周期:

5.1 6 个月

6. 整个电商结构图



7. 基于 SOA 架构



二. 数据库准备和逆向工程

1. 直接运行 SQL 脚本.
2. 使用逆向工程生成 mapper 和 pojo

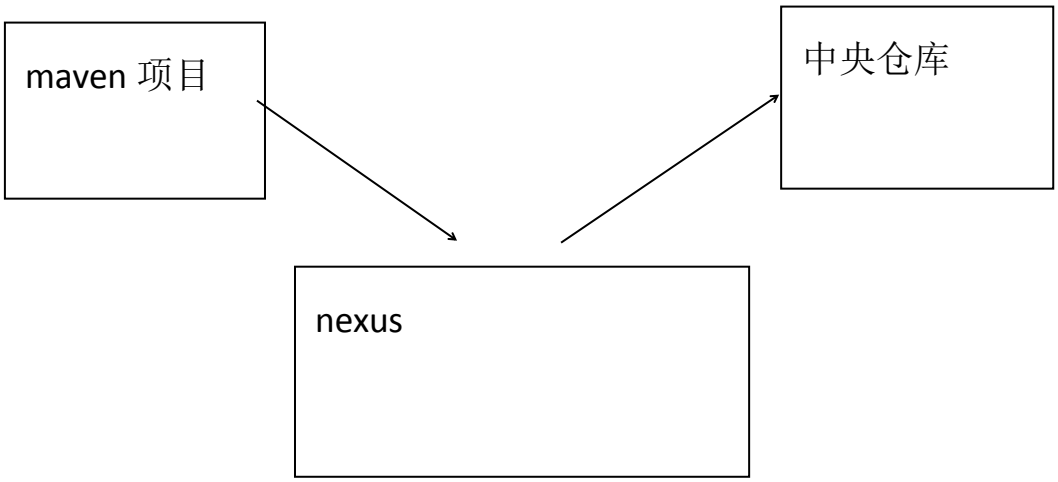
三.搭建 maven 环境

1. 为什么使用 Nexus 搭建 maven 私服(私服的作用)

1.1 公司所有开发成员没有外网,通过局域网连接 nexus 私服,由私服连接外网

1.2 把项目发布到私服.其他人员从私服下载.

2. 使用私服之后架构图



3. 搭建 nexus 的步骤

3.1 nexus-2.12.0-01-bundle.zip 解压到任意非中文目录中

3.2 修改 nexus 端口(默认 8081)

3.2.1 nexus-2.12.0-01\conf\nexus.properties

```
# This is the most basic configurati

# Jetty section
application-port=8091
application-host=0.0.0.0
nexus-webapp=${bundleBasedir}/nexus
```

3.3 粘贴索引库(不配置无法搜索)

3.3.1 先清空 sonatype-work\nexus\indexer\central-ctx 内容

3.3.1 把解压后的索引文件粘贴到这个文件夹中

3.4 进入 nexus-2.12.0-01\bin\jsw\windows-x86-64(对应自己系统)

3.4.1 install-nexus.bat 安装服务

3.4.2 start-nexus.bat 开启服务

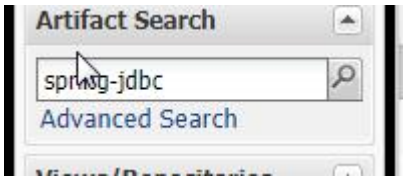
3.4.3 stop-nexus.bat 停止服务

3.4.4 uninstall-nexus.bat 卸载服务

3.5 在浏览器输入 http://localhost:8091/nexus

3.6 点击右侧 log in ,输入用户名:admin,密码:admin123

3.7 在左侧搜索框中输入 artifact id 测试是否配置成功



4. 使用 maven 连接私服

4.1 前提:把 maven 环境搭建,并设置 users settings 引用 settings.xml

4.2 在 settings.xml 配置

4.2.1 本地仓库路径

```
<localRepository>D:/maven/myrepository</localReposito
ry>
```

4.2.2 配置 jdk

```
<profile>
        <id>jdk-1.7</id>
        <activation>
          <activeByDefault>true</activeByDefault>
          <jdk>1.7</jdk>
        </activation>
```

```
        <properties>

   <maven.compiler.source>1.7</maven.compiler.source>

   <maven.compiler.target>1.7</maven.compiler.target>

   <maven.compiler.compilerVersion>1.7</maven.compiler
.compilerVersion>
        </properties>
      </profile>
```

4.2.3 配置私服构建(连接私服用的到 jar 等内容)

```
    <profile>
<id>nexusTest</id>
<repositories>
<repository>
<id>local-nexus</id>
<url>http://127.0.0.1:8091/nexus/content/groups/publi
c/</url>
<releases>
<enabled>true</enabled>
</releases>
<snapshots>
```

```
<enabled>true</enabled>
</snapshots>
</repository>
</repositories>
</profile>
```

4.2.4 配置让私服构建生效

4.2.4.1 nexusTest 上面<profile>的<id>

```
  <activeProfiles> <!--激活 id 为 nexusTest 的 profile-->
<activeProfile>nexusTest</activeProfile>
</activeProfiles>
```

4.2.5 配置镜像,maven 连接私服

```
     <mirror>
      <id>nexus-releases</id>
      <mirrorOf>*</mirrorOf>

<url>http://localhost:8091/nexus/content/groups/publi
c</url>
    </mirror>
     <mirror>
       <id>nexus-snapshots</id>
      <mirrorOf>*</mirrorOf>
```

```
<url>http://localhost:8091/nexus/content/repositories
/apache-snapshots/</url>
    </mirror>
```

5. 把项目发布到私服的步骤

5.1 在 pom.xml 中配置私服路径

```
  <distributionManagement>
    <repository>
      <id>releases</id>

  <url>http://localhost:8091/nexus/content/repositori
es/releases</url>
    </repository>
    <snapshotRepository>
      <id>snapshots</id>

  <url>http://localhost:8091/nexus/content/repositori
es/snapshots</url>
    </snapshotRepository>
  </distributionManagement>
```

5.2 在 settings.xml 中配置连接私服仓库的用户名和密码

5.2.1 <server>中<id>和 pom.xml 中<repository>中<id>对应

```
<server>
```

```
    <id>releases</id>
    <username>admin</username>
    <password>admin123</password>
  </server>
  <server>
    <id>snapshots</id>
    <username>admin</username>
    <password>admin123</password>
  </server>
```

5.3 右键项目--> run as 输入 deploy

四.创建项目

1. 创建六个项目

   1.1 ego-commons: 放工具类等

   1.2 ego-manage: 后台项目

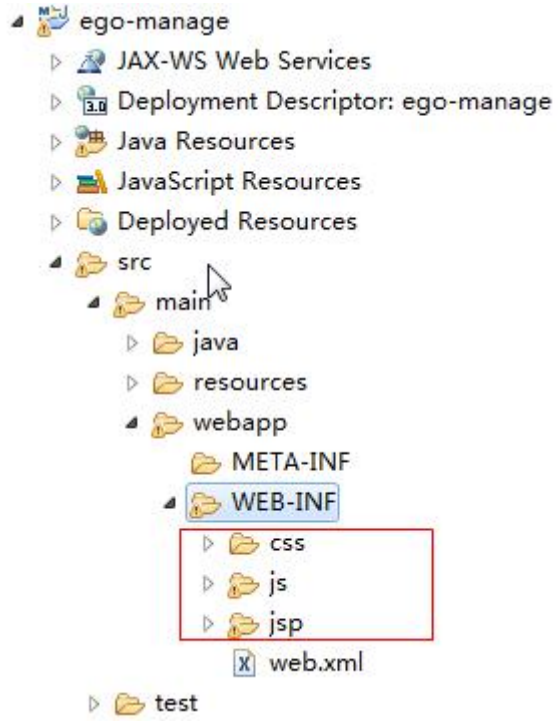
   1.3 ego-parent:父项目

   1.4 ego-pojo:实体类

   1.5 ego-service: 服务接口

   1.6 ego-service-impl: dubbo 的 provider

2. 把后台页面放在 ego-manage/WEB-INF 中



3. 在 ego-manage 编写控制器类

```
@Controller
public class PageController {
   @RequestMapping("/")
   public String welcome(){
      return "index";
   }
   @RequestMapping("{page}")
```

```
    public String showPage(@PathVariable String page){
        return page;
    }
}
```

五. MyBatis 分页插件

1. 在 mybatis.xml 中配置<plugin>标签,在程序员所编写的 sql 命令基础上添加一些内容.

2. 在 pom.xml 配置依赖

```
<!-- 分页插件 -->
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>4.1.6</version>
        </dependency>
```

3. 创建 mybatis.xml 并配置插件信息

```
<?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>
   <plugins>
      <plugin
interceptor="com.github.pagehelper.PageHelper">
         <!-- 告诉分页插件是哪个数据库 -->
         <property name="dialect" value="mysql"/>
      </plugin>
   </plugins>
</configuration>
```

4. 在 applicationContext.xml 中配置加载 mybatis.xml

```
    <!-- SqlSessionFactory -->
    <bean id="factory"
class="org.mybatis.spring.SqlSessionFactoryBean">
      <property name="dataSource"
ref="dataSource"></property>
      <property name="typeAliasesPackage"
value="com.ego.pojo"></property>
      <property name="configLocation"
value="classpath:mybatis.xml"></property>
    </bean>
```

5. 编写代码时注意: PageHelper.startPage()写在查询全部上面.

```
PageHelper.startPage(page, rows);
```

```
        //查询全部
        List<TbItem> list =
tbItemMapper.selectByExample(new TbItemExample());
        //分页代码
        //设置分页条件
        PageInfo<TbItem> pi = new PageInfo<>(list);
```

六. 实现商品分页显示功能

1. 在 ego-commons 中创建 easyuidatagrid 类

   1.1 把所有 ego-pojo 中类序列化

```
public class EasyUIDataGrid implements Serializable {
    //当前页显示数据
    private List<?> rows;
    //总条数
    private long total;
```

2. 在 ego-service 中创建接口

```
public interface TbItemDubboService {
    /**
     * 商品分页查询
     * @param page
```

```
    * @param rows
    * @return
    */
   EasyUIDataGrid show(int page,int rows);
}
```

3. 在 ego-service-impl 编写功能

```
public class TbItemDubboServiceImpl implements
TbItemDubboService {
   @Resource
   private TbItemMapper tbItemMapper;
   @Override
   public EasyUIDataGrid show(int page, int rows) {
      PageHelper.startPage(page, rows);
      //查询全部
      List<TbItem> list =
tbItemMapper.selectByExample(new TbItemExample());
      //分页代码
      //设置分页条件
      PageInfo<TbItem> pi = new PageInfo<>(list);

      //放入到实体类
      EasyUIDataGrid datagrid = new EasyUIDataGrid();
```

```
        datagrid.setRows(pi.getList());
        datagrid.setTotal(pi.getTotal());
        return datagrid;
    }
}
```

4. 在 ego-service-impl 中 applicationContext-dubbo.xml 配置接口

```
<dubbo:service
interface="com.ego.dubbo.service.TbItemDubboService"
ref="tbItemDubboServiceImpl"></dubbo:service>
    <bean                    id="tbItemDubboServiceImpl"
class="com.ego.dubbo.service.impl.TbItemDubboServiceI
mpl"></bean>
```

5. 编写 Test 类运行 dubbo 服务

```
    public static void main(String[] args) {
        Main.main(args);
    }
```

6. 在 ego-manage 添加 TbItemService 及实现类

```
public interface TbItemService {
    /**
     * 显示商品
     * @param page
     * @param rows
```

```
    * @return
    */
   EasyUIDataGrid show(int page,int rows);
}
```

```
@Service
public class TbItemServiceImpl implements TbItemService{
   @Reference
   private TbItemDubboService tbItemDubboServiceImpl;
   @Override
   public EasyUIDataGrid show(int page, int rows) {
      return tbItemDubboServiceImpl.show(page, rows);
   }
}
```

7. 在 ego-manage 里新建 TbItemController

```
@Controller
public class TbItemController {
   @Resource
   private TbItemService tbItemServiceImpl;
   /**
    * 分页显示商品
    */
   @RequestMapping("item/list")
```

```
    @ResponseBody
    public EasyUIDataGrid show(int page,int rows){
        return tbItemServiceImpl.show(page, rows);
    }
}
```

七. 商品上架,下架,删除

1. 在 ego-service 中 TbItemDubboService 接口及实现类添加

```
    /**
     * 根据 id 修改状态
     * @param id
     * @param status
     * @return
     */
    int updItemStatus(TbItem tbItem);
```

```
    @Override
    public int updItemStatus(TbItem tbItem) {
        return
tbItemMapper.updateByPrimaryKeySelective(tbItem);
    }
```

2. 在 ego-commons 中添加 EgoResult,做为 java 代码和 jsp 叫做公共类

```
public class EgoResult {

   private int status;
```

3. 在 ego-manage 的 TbItemService 及实现类添加

```
   /**
    * 批量修改商品状态
    * @param ids
    * @param status
    * @return
    */
   int update(String ids,byte status);
```

```
@Override
   public int update(String ids, byte status) {
      int index = 0 ;
      TbItem item = new TbItem();
      String[] idsStr = ids.split(",");
      for (String id : idsStr) {
         item.setId(Long.parseLong(id));
         item.setStatus(status);
         index
+=tbItemDubboServiceImpl.updItemStatus(item);
      }
      if(index==idsStr.length){
```

```
            return 1;
        }
        return 0;
    }
```

4. 在 ego-manage 中 TbItemController 添加三个控制器方法

```
/**
     * 商品删除
     * @param ids
     * @return
     */
    @RequestMapping("rest/item/delete")
    @ResponseBody
    public EgoResult delete(String ids){
        EgoResult er = new EgoResult();
        int index = tbItemServiceImpl.update(ids,
(byte)3);
        if(index==1){
            er.setStatus(200);
        }
        return er;
    }
    /**
```

```
    * 商品下架
    * @param ids
    * @return
    */
   @RequestMapping("rest/item/instock")
   @ResponseBody
   public EgoResult instock(String ids){
      EgoResult er = new EgoResult();
      int index = tbItemServiceImpl.update(ids,
(byte)2);
      if(index==1){
         er.setStatus(200);
      }
      return er;
   }
   /**
    * 商品上架
    * @param ids
    * @return
    */
   @RequestMapping("rest/item/reshelf")
   @ResponseBody
```

```
   public EgoResult reshelf(String ids){
      EgoResult er = new EgoResult();
      int index = tbItemServiceImpl.update(ids,
(byte)1);
      if(index==1){
         er.setStatus(200);
      }
      return er;
   }
```