# 分布式项目实战

## 技术栈大纲



+2.8 jsonp 跨域数据请求格式

2.9 nexus maven 私服

2.10 mybatis 逆向工程

2.11HttpClient 使用java完成请求及响应的技术

2.12mycat mysql分库分表技术

常用的jsr规范，HTTP协议，netty框架，消息中间件，分布式编程框架，并发编程

### 结构图



前台做前6个，后台做商品管理和CMS（消息管理）

后台使用esayui，前台用标准框架。

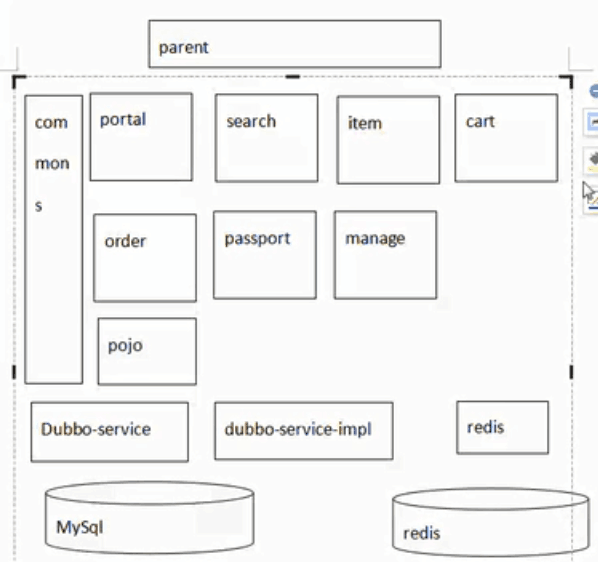
### 【域名】代表不同项目

域名：www.jd.com，随便搜索一个后变成search.jd.com/……，点开某个商品变成了item.jd.com/……，加入购物车cart.jd.com/……，结算order.jd.com/……，请登录passport.jd.com/……，前台就用了5个了。后台看不见

一个1级域名下有20个免费2级域名。比如www下面有search等。一旦域名换了说明项目换了。

因此www门户项目和search搜索项目是2个项目。

### 基于SOA架构。（面向服务）



* **服务：**

Dubbo：service，service实现，实体类pojo

* **前台：**

Portal：门户

Search:搜索

Item:商品

Cart:购物车

Order:订单

Passport：登录

* **后台：**

manage

* **公共项目：**

Commons

* **单独的:**

Redis

* **Maven-parent**

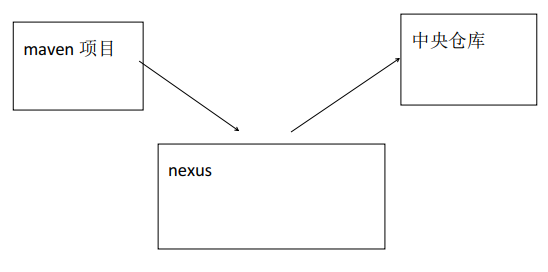
parent

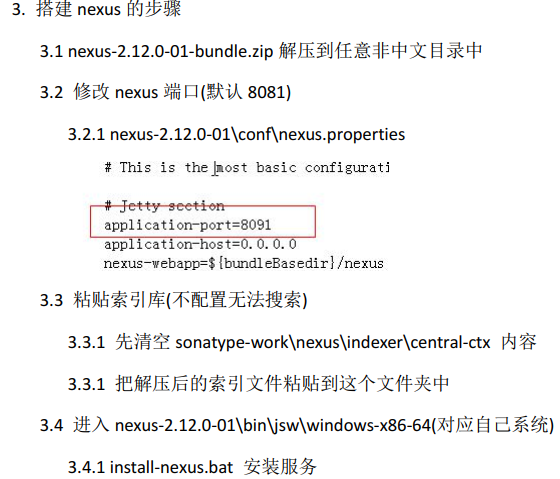
## 第一天 后台

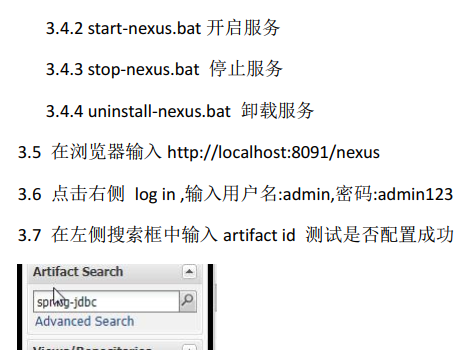
Parent, commons, pojo, dubbo, manage.

### 步骤

1. **建立数据库**
2. **逆向工程**
3. **搭建maven环境—使用Nexus搭建maven私服。私服的作用：**
   1. 公司所有开发成员没有外网，通过局域网链接nexus私服，由私服链接外网。
   2. 把项目发送到私服，其他人员从私服下周。
   3. 使用私服之后的架构图：







[Linux nexus 搭建私服和安装索引](https://blog.csdn.net/lzwjavaphp/article/details/43057047)

1. **使用 maven 连接私服**

只要在maven中配置：私服构件（链接私服用到的jar等）、jdk、镜像-直接连私服、本地仓库、激活私服构件

1. **搭建项目**

**新建maven工程：**

**ego-parent**：父项目，maven管理，pom类型

**ego-pojo**：实体类，把逆向工程生成的com.ego.pojo包拷贝进来。Jar

**ego-service**：服务接口，在pom中依赖ego-pojo。jar

**ego-service-impl** (ego-dubbo)：dubbo的provider，在pom中依赖spring、dubbo、mybatis、mysql、log4j、ego-service，版本去掉。把逆向工程生成的com.ego.dao包和mapper文件拷贝进来。再在resources中配置基本文件applicationContext.xml，log4j.properties，applicationContext-spring.xml，db.properties，dubbo(必须准备好本地dubbo.xsd和spring-context.xsd，在xml catalog中设置关联，key type为Schema location,否则项目会卡)，（从别处拷贝），再配置相关文件。jar

**ego-commons:** 空对象,放工具类。jar

**ego-manage:** 后台项目，在webapp下设置web.xml；设置pom：pluginmanagement+dependencymanagement+版本号删掉，保留zookeeper、dubbo、文件上传、jackson、log4j、springwebmvc、jstl、jredis、依赖接口（搭私服的主要目的）；设置配置文件 war

1. [**在linux上配置zookeeper、dubbo**](dubbo-zookeeper安装.docx)
2. **修改注册地址为目标linuxip:2181**
   1. **ego-service-impl resources/META-INF.spring下applicationContext-dubbo.xml.**
   2. **ego-manage resources/applicationContext-dubbo.xml**
3. **导入后台页面**

将第一天的后台页面放入ego-manage的WEB-INF下

修改springmvc.xml。配置js、image、css地址；配置视图解析器jsp地址。

写controller:com.ego.manage.controller

pageController.java

配置tomcat。

运行测试。

* **出现问题：**

springmvc controller调用不了、出现404。解决方法：1、检查配置；2、检查project structure的spring lib依赖；3、添加web facet，注意路径。

前端页面没有效果。解决：修改tomcat直接访问项目主页，如下。

**直接访问项目主页：**

/\* java tomcat中service.xml中<context>的path一般都是默认项目名的，你把它直接改成/就可以不用加项目名了\*/

其实把那个有冲突的index.jsp删除就可以了

至此项目运行成功！

1. **mybatis分页插件**
   1. **配置**

粘贴mybatis.xml到ego-service-impl resources下并配置本地dtd \*systemid

配置plugin

添加依赖

在applicationContext.xml中factory下添加configLocation属性，value=”classpath:mybatis.xml”。

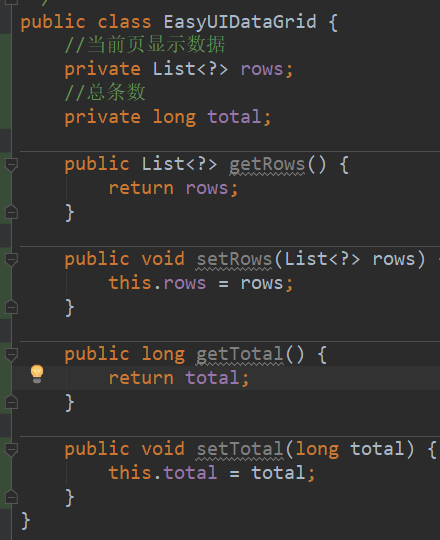
* 1. **使用**

**在ego-service下新建接口：com.ego.dubbo.service.TbItemDubboService**，主要操作tb\_item表。

Esayui返回的都在commons中，rows,total。

**在ego-commons中新建com.ego.commons.pojo包**

**包下新建EasyUIDataGrid.java**



**回到service接口中继续编写：**

**添加commons依赖**

编写Service

public interface TbItemDubboService {

/\*\*

\* 商品分页查询

\* @param page

\* @param rows

\* @return

\*/

EasyUIDataGrid show(int page, int rows);

}

**写实现类，在ego-service-impl**

public class TbItemDubboServiceImpl implements TbItemDubboService {

@Resource

private TbItemMapper tbItemMapper;

@Override

public EasyUIDataGrid show(int page, int rows) {

//查询全部

List<TbItem> list = tbItemMapper.selectByExample(new TbItemExample());

//分页代码

//设置分页条件

PageHelper.startPage(page, rows);

PageInfo<TbItem> pi = new PageInfo<>(list);

//此时分页插件生效，pi中包含所有分页相关信息。

//放入到实体类

EasyUIDataGrid dataGrid = new EasyUIDataGrid();

dataGrid.setRows(pi.getList());

dataGrid.setTotal(pi.getTotal());

return dataGrid;

}

}

**写test类**

package com.ego.test;

import com.alibaba.dubbo.container.Main;

/\*\*

\* @author zdd

\* @date 2019-05-09 0:53

\*/

public class Test {

public static void main(String[] args) {

Main.main(args);

}

}

* 会遇到各种问题，需要进行如下步骤的操作：

1. 将spring包的版本统一到4.3.16（在pom和某些iml中，通过double shift来查找哪些包依赖了其他的版本）
2. 导入spring-beans包
3. 检查applicationContext发现扫描器定位。
4. 有些依赖在子项目pom中需要，而有些却不需要，不要随便加依赖
5. 改用c3p0加载数据源
6. 把分页插件改到5.0.0，在mybatis.xml中的配置com.github.pagehelper.PageHelper，去掉dialect。

7、登录dubbo服务器发现没有结果。

试错，测试applicationContext-dubbo.xml是不是存在问题。

没加载？随便写一段话，运行，报错，说明加载了。

最终发现接口没有注册进去。注册即可。

8、导入netty -all和curator-framework依赖

注意：导入curator后右报错，系版本冲突所致，因此要么zookeeper3.5+curator4.0要么curator改成2.11.1。zookeeper重装有点麻烦。

官方文档如下：

Versions

The are currently two released versions of Curator, 2.x.x and 3.x.x:

Curator 2.x.x - compatible with both ZooKeeper 3.4.x and ZooKeeper 3.5.x

Curator 3.x.x - compatible only with ZooKeeper 3.5.x and includes support for new features such as dynamic reconfiguration, etc

至此dubbo test运行成功！

服务方代码DONE

1. **在manage中调dubbo**

Jsp写完了，项目中就只需要写控制器跟service

Service:

新建接口：com.ego.manage.service.TbItemService

public interface TbItemService {

/\*\*

\* 显示商品

\*/

EasyUIDataGrid show(int page, int rows);

}

新建实现类：com.ego.manage.service.impl.TbItemServiceImpl

@Service

public class TbItemServiceImpl implements TbItemService {

@ Reference

private TbItemDubboService tbItemDubboServiceImpl;

@Override

public EasyUIDataGrid show(int page, int rows) {

return tbItemDubboServiceImpl.show(page, rows);

}

}

Controller:

在Controller中新建TbItemController用于管理商品。

@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);

}

}

到此商品查询功能完成。

500错误：

1、applicationContext-dubbo.xml中把注解扫描中的\*改成manage

2、使用dubbo必须序列化，每个非Example的pojo实现Serializable接口

其他问题：spring4.3.16会使得tomcat启动报错，改回4.2.6，则会出现之前总遇到的错误。正确方式是在project structure中fix一些问题。后面遇到其他的class not found问题则在添加依赖后在iml中也加上依赖再加到artifact中。再解决一些版本问题就可以运行出来了。

NullPointerException:仔细检查配置文件还有project-structure中的玄学依赖包。

还有就是@Reference应该用dubbo的，要不然会注入不了。

Reference会创建该类的一个代理对象。

3、mysql驱动包找不到：project-structure中发现unknown版本的依赖，换掉，然后pom中也进行一些操作。

4、java.lang.IllegalArgumentException: No converter found for return value of type: class com.ego.commons.pojo.EasyUIDataGrid —5小时

Jackson核心包没有导入。包括父项目和子项目

配置springmvc.xml，添加转换器。

\*在springmvc.xml中配置了转换器后java.lang.NoClassDefFoundError: com/fasterxml/jackson/core/util/DefaultIndenter

Jackson包和spring版本不匹配。可是没用，再从project structure入手，手动引入jackson包。

再不行就把本地maven仓库的删除重下

**还是不行，使用clean大法，发现好多报错，一个一个清理，然后clean，然后install。**

还是不行，检查settings->build,exception…->build tools->maven->ignored,是不是把某些pom忽略了。把前面的勾去掉。

还是不行，检查项目的pom，发现重复依赖了jackson，去掉。成功运行！！！

总结：归根接地这些乱起八糟的问题给根源就是Maven胡乱导入包。

总结2：

Step1:分页插件，在commons中创建esayUIDataGrid类并序列化

Step2:在ego-service中创建接口

Step3:在ego-service-impl中编写功能

Step4:在ego-service-impl中app…-dubbo.xml中配置接口

Step5:编写Test类运行dubbo服务

Step6:在ego-manage中添加TbItemService及实现类

Step7:在ego-manage中新建TbItemController

注意：改了dubbo服务端代码一定要重启

## 第二天 后台

### 一、基本步骤框架：以及添加类目功能

1. **启动zookeeper** ./zkServer.sh start
2. **启动dubbo服务(ego-service-impl test)**
3. **启动manage**
4. **编写dubbo服务端：**
   1. **新建dubbo-service：TbItemCatDubboService接口及实现类**
   2. **在-dubbo.xml中配置该实现。注册发布服务接口**

<!--商品类目服务-->

<dubbo:service interface="com.ego.dubbo.service.TbItemCatDubboService" ref="TbItemCatDubboServiceImpl" />

<bean id="TbItemCatDubboServiceImpl" class="com.ego.dubbo.service.impl.TbItemCatDubboServiceImpl" />

* 1. **重启dubbo服务**

1. **编写commons端**
   1. **新建EasyUITree(菜单树)**
   2. **按条件（官方文档）格式化类Tree**
2. **编写manage端**
   1. **新建manage-service: TbItemCatService及其实现类**
   2. **新建manage-controller: TbItemCatController**
3. **小结**

**Dubbo的service负责查询sql，而manage中的service负责封装结果。**

**也就是说dubbo端和mapper打交道，manage端和service、dubbo-service打交道。**

### 二、nginx实现商品新增-图片上传 涉及到nginx

**准备工作：**

1、linux需要安装vsftpd，安装后实现使用ftpclient完成图片上传功能。

2、在vsftpd所在服务器安装nginx，实现图片回显

**步骤：**

1. 克隆一个linux虚拟机：右键🡪管理🡪克隆，第一步选现有快照（需要提前新建快照），第二步选择创建完整克隆。克隆时候需要对网卡设置新的ip和NM\_CONTROLLED=no
2. 查看ip、或设置ip，使用Xshell连接。
3. [安装vsftpd](vsftpd安装手册(搭建ftp清晰版).doc)
4. [安装、启动nginx](nginx安装手册.doc)，
   1. 先把压缩包上传
   2. 转到存放压缩包的目录下，解压tar zxvf nginx-1.8.0.tar.gz
   3. 转到解压目录下，创建目录mkdir -p /var/temp/nginx。
   4. 转到存放压缩包的目录下，创建目录mkdir -p a/b
   5. 转到nginx目录下，配置参数

./configure \

--prefix=/usr/local/nginx \

--pid-path=/var/run/nginx/nginx.pid \

--lock-path=/var/lock/nginx.lock \

--error-log-path=/var/log/nginx/error.log \

--http-log-path=/var/log/nginx/access.log \

--with-http\_gzip\_static\_module \

--http-client-body-temp-path=/var/temp/nginx/client \

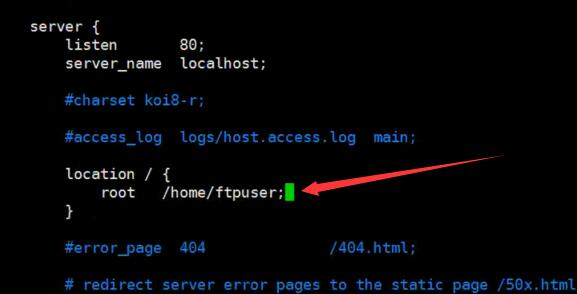
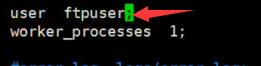
--http-proxy-temp-path=/var/temp/nginx/proxy \

--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \

--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \

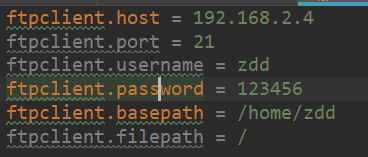
--http-scgi-temp-path=/var/temp/nginx/scgi

* 1. make
  2. make install，默认装在了/usr/local/ngnix下。
  3. 进入安装目录下，进入conf目录，vim nginx.conf



* 1. cd .. ; ./nginx 启动！
  2. 停止：cd /usr/local/nginx/sbin；./nginx -s stop
  3. 重启：./nginx -s quit；./nginx；或./nginx -s reload
  4. 设置开机启动: vim /etc/init.d/nginx
  5. 设置文件访问权限
  6. 加入到rc.local中。
  7. 开放80端口
  8. 浏览器访问<http://192.168.2.4/>工具.txt

1. 实现图片上传的代码
   1. 在manage中新建PicService及其实现
   2. 在commons包中新建com.ego.commons.utils，把FtpUtil.java和IDUtil.java复制进去。，在pom中导入commons-net3.3、commons-fileupload 1.3.1
   3. 在commons包中的resources中新建commons.properties



再在manage的applicationContext-spring.xml中扫描属性文件。

<context:property-placeholder location="classpath:commons.properties" />

* 1. 在impl中处理这些参数，[用@Value(“${ftpclient.host}”)](mailto:用@Value(“$%7bftpclient.host%7d”))来注入string对象。
  2. 使用kindEditor
  3. 新建PicController
  4. 在springmvc.xml中加载Multipart解析器multipartResolver

1. 实现新增功能
   1. 先在浏览器中新增，然后从f12🡪network🡪查看form data
   2. 第一件事：dubbo-service：在ego-service中新建TbItemDescDubboService接口，在ego-service-impl中定义其实现。
   3. 在TbItemDubboService中定义方法insTbItem，并实现
   4. 在dubbo.xml中注册服务
   5. ============================DUBBO DONE===========
   6. Manage端：在TbItemController新建返回值为EgoResult的方法insert()
   7. 在TbItemService中编写方法save()及其实现类
   8. 不具备回滚功能，要具备就要写多个dubbo添加类。
2. 实现回滚
   1. 在TbItemDubboService中添加方法insTbItemDesc及其实现。使用捕获异常的方法。异常就回滚。throw new Exception("新增失败,数据还原");
   2. 在applicationContext.xml中<声明式事务>添加一句话：<tx:method name=”ins” rollback-for=”java.lang.Excption”/>
   3. 回到manage中，只需修改TbItemServiceImpl中的save()方法
   4. 测试回滚：让商品表添加成功，描述表添加失败：在save()方法中不设置itemDesc的主键。-注释掉itemDesc.setItemId(id);
   5. 提示异常信息：在EgoResult中添加Object属性data，然后Controller捕获到下层不断抛的异常后把原因字符串放到data中，前端item-add.jsp获取字符串提示给用户，方法不会捕获异常，只有控制器捕获。er.setData(e.getMessage());

## 第三天 后台

一、规格参数管理

1. 介绍
   1. 规格参数为组名、属性名、属性信息。2张表，1张group表，一张子项表，每个子项引用一个group，每个子项后面放着内容
   2. 同一个类型的商品下的规格参数模板差不多，不同类型灵活性比较高。
   3. TbItem表，下面有一个cat表，每个分类下有不同的规格参数，每个不同的规格参数有个group，还有小的子项。
2. 优化
   1. 需求：每个分类有不同的分类模板；根据模板每个商品有具体信息。
   2. 传统数据库设计：



商品-子项表数据太庞大。

* 1. 优化数据库设计

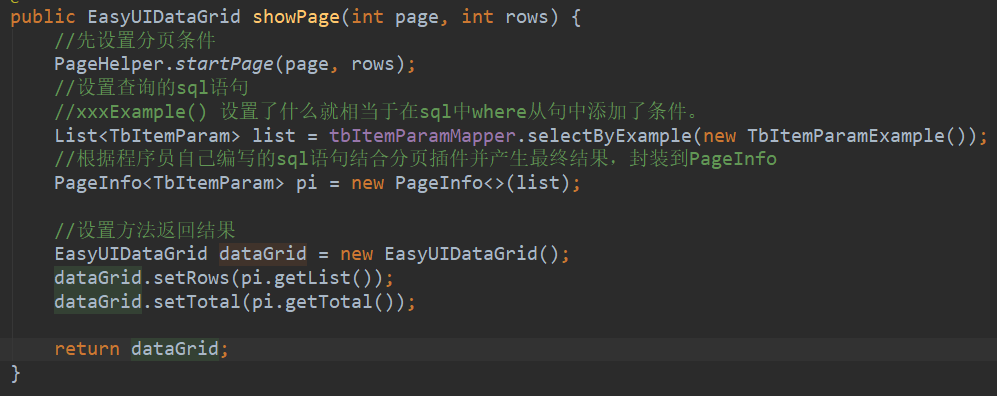
前提：group和子项的对应关系比较灵活。

商品-子项表数据比较多

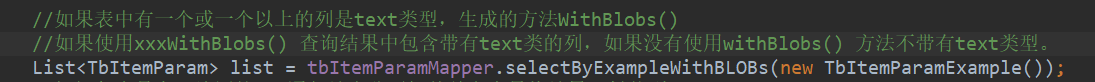
优化后：①所有的规格参数模板设置到一个列中，这个列中存储一个特定格式的数据。（看到格式，想到json），把数据存储为json数据格式。优点：传输时可以对格式快速进行解析，能取出一长段话中固定某一小块的内容。（为新增做模板准备，最少3列可实现）。**②**设计一个表，设置某个列中存储json数据格式，目的是：存储某个商品的具体规格信息。（最少3列可实现）

1. 实现
   1. 写接口及其实现：TbItemParamDubboService、TbItemParamDubboServiceImpl

查询代码模板如下：



* 1. 在dubbo.xml中注册服务。(dubbo三步骤：接口、实现类、注册服务)
  2. 在manage中写TbItemParamService及其实现
  3. 在controller中写TbItemParamController
  4. 会出错，paramData为null，数据库中的是Text类型，逆向工程有要求，所以在Dubbo中写的查询语句需要注意修改WithBlobs()。



* 1. 联合查询出类目，在ItemCatDubboService提供根据类目id查询的方法，及其实现。再在manage中的TbItemParamServiceImpl中改一下。新建pojo：com.ego.manage.pojo.TbItemParamChild继承TbItemParam。这样依赖就不影响原先的实体类了。然后同时调用两个Dubbo类来联合查询，set到子类中。

1. 删除功能
   1. 在TbItemParamDubboService中添加delByIds方法及其实现。//回滚，比如说，其他管理员已经删除一条，而你这里还存在。 因此要在applicationContext.xml中添加rollback声明。
   2. 于是在TbItemParamService中也添加对应的方法及其实现。
   3. 在Controller中处理
2. 商品类目新增
   1. 在TbItemParamDubboService中添加selByCatId()方法及其实现，重启
   2. 在TbItemParamService中添加showParam()方法及其实现，此处返回值直接为EgoResult。
   3. 在控制器中写show()方法
   4. 在TbItemParamDubboService中添加insParam()方法及其实现，实现中必须用insertSelective()，因为主键自增，重启。
   5. 在TbItemParamService中添加save()方法及其实现。
3. 商品新增-包含规格参数信息
   1. 直接在TbItemDubboServiceImpl的insTbItemDesc中添加操作。
   2. 在manage的TbItemServuce和实现以及Controller中也修改一下，多接收一个参数。Contorller中的参数名要和前端传过来的一致！

## 总结

1. Dubbo开发三步骤：接口、实现类、注册；启动：zookeeper、Test服务。
2. Manage开发三步骤：Service、Impl、Controller。（JSP）
3. @RequestMapping(“”) == @RequestMapping != @RequestMapping(“/”).前两个是截获任意地址的请求，最后一个是截获项目页面的请求.
4. @ResponseBody转换成json数据
5. 报404: springmvc问题，从头开始检查-jar包、spring配置文件;资源404-静态资源问题；控制器404-包扫描问题，扫描包的那句话和包名对应没。
6. EgoResult可以直接写在Service中返回，在增删改时候写在控制器中，如果写在Service中就要处理异常。
7. 经常maven install，因为依赖的项目可能有更新