# 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。

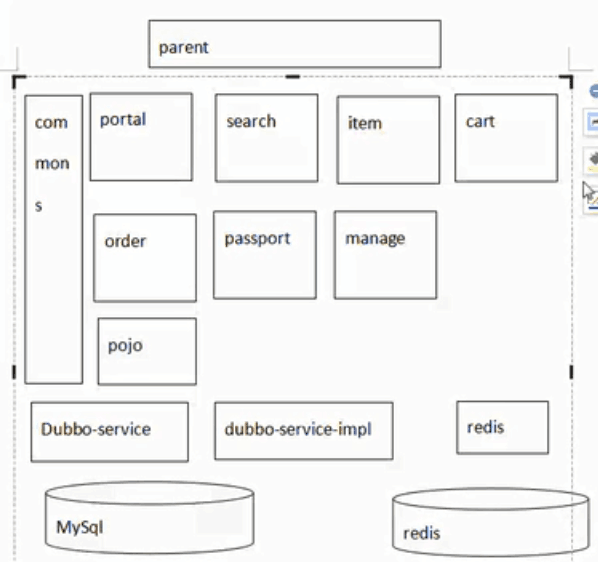
### 【域名】代表不同项目

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**

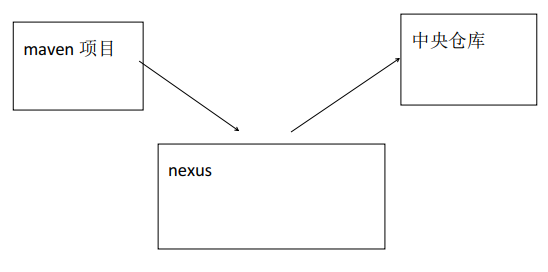
parent

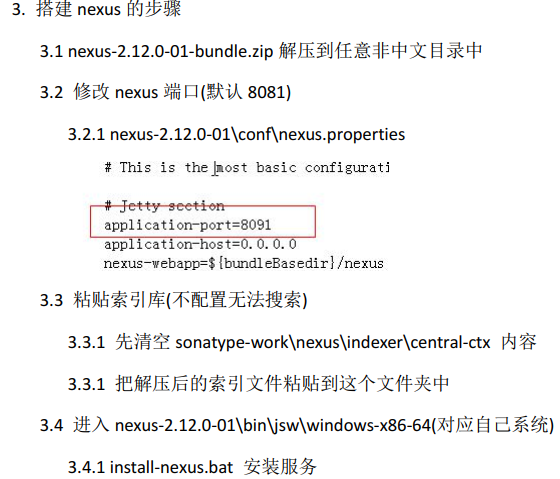
## 第一天 后台

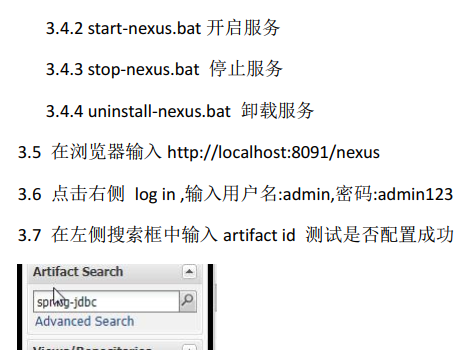
Parent, commons, pojo, dubbo, manage.

### 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”。

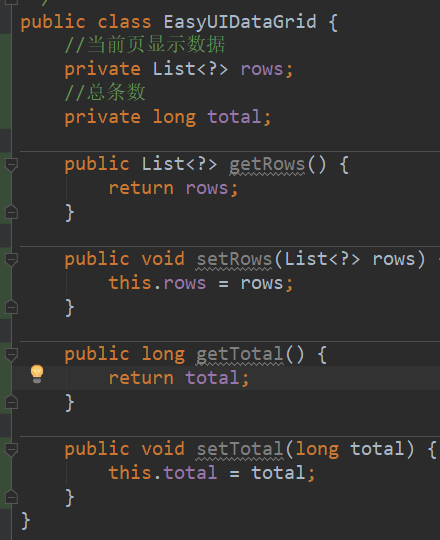
* 1. **使用**

**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

## 名词总结

|  |  |  |
| --- | --- | --- |
| 术语 | 英文单词 | 术语描述 |
| 内存屏障 | Memory barriers | 是一组处理器指令，用于实现对内存操作的顺序限制。 |
| 缓冲行 | Cache line | CPU高速缓存中可以分配的最小存储单位，处理器填写缓存行时会加载整个内存行，现代CPU要执行几百次CPU指令 |
| 原子操作 | Stomic operations | 不可中断的一个或一系列操作 |
| 缓存行填充 | Cache line fill | 当处理器识别到内存中读取操作数是可缓存的，处理器读取整个高速缓存行到适当的缓存(L1,L2,L3的或所有) |
| 缓存命中 | Cache hit | 高速缓存行填充操作的内存位置任然是下次处理器访问的地址时，处理器从缓存中读操作数而不是内存中。 |
| 写命中 | Write hit | 处理器将操作数写回到一个内存缓存的区域时，它首先会检查这个缓存地址是否在缓存行中，存在则写回到缓存而不是内存中。 |
| 写缺失 | Write misses the cache | 一个有效的缓存行被写入到不存在的内存区域。 |
| =================================================================== | | |
| 缓存行 | Cache line | 缓存的最小操作单位 |
| 比较并交换 | Compare and swap | CAS操作需要输入两个数值，一个旧值（期望操作前的值）和一个新值，比较旧值是否变化（与期望值相等），如果没有就用新值替换。 |
| CPU流水线 | CPU pipeline | CPU流水线的工作方式就像工业生产上的装配流水线，在CPU中由5-6个不同功能的电路单元组成一条指令处理流水线，然后将一条X86指令分成5-6步后由这些电路单元分别执行，这样就能实现在一个CPU时钟周期完成一条指令，因此提高CPU的运算速度。 |
| 内存顺序冲突 | Memory order violation | 内存顺序冲突一般是由假共享引起的，假共享是指多个CPU同时修改一个缓存行的不同部分而引起其中一个CPU的操作无效，这种情况CPU必须清空流水线。 |