# 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。

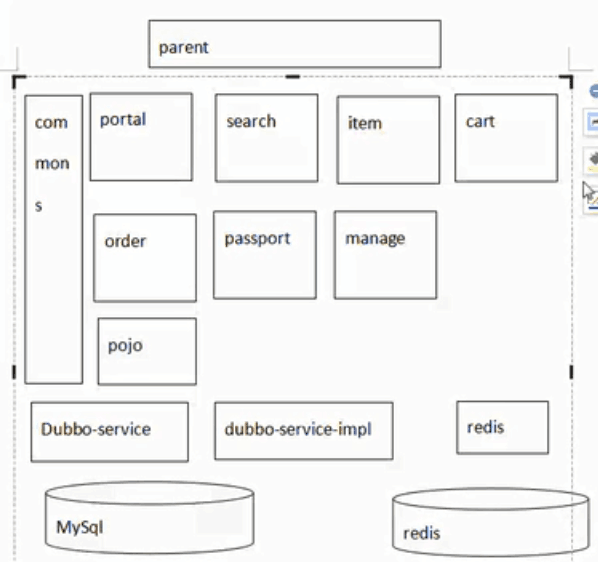
### 【域名】代表不同项目

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**

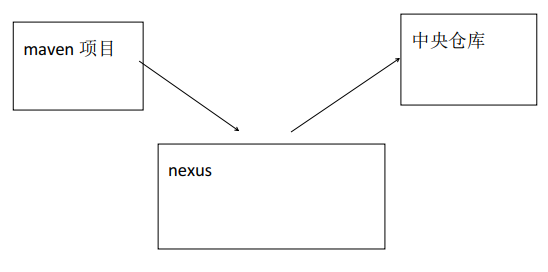
parent

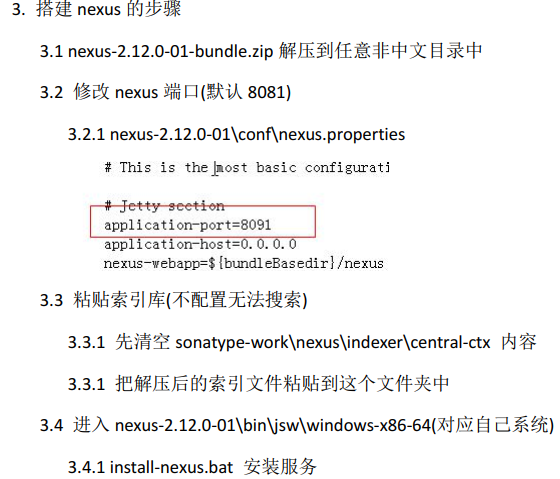
## 第一天 后台

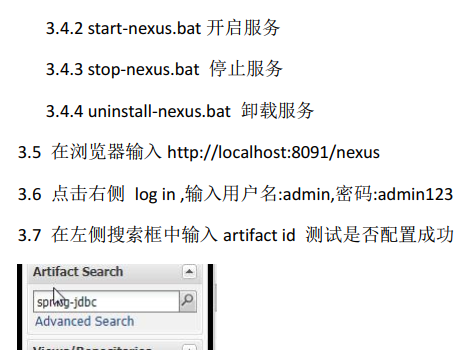
Parent, commons, pojo, dubbo, manage.

### 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**：maven管理，pom类型

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

**ego-service**：在pom中依赖ego-pojo。jar

**ego-service-impl** (ego-dubbo)：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。jar

**ego-manage:** 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

## 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。 |