刘昱江 QQ:527161772 中心 +班级 +名字

# 互联网公司特点

## 行业特点

### 分布式

#### 分布式计算

说明:将一项任务,交给多台计算机共同处理.

例子:一项任务一个人单独完成需要10个小时.如果10个人同时干活,理论值1个小时做完了.(大数据分布式计算hdfs)

#### 分布式系统(微服务)

概念:将项目按照功能模块和层级进行拆分.拆分为多个项目

##### 项目的垂直拆分

说明:按照功能模块,将项目拆分为多个项目.

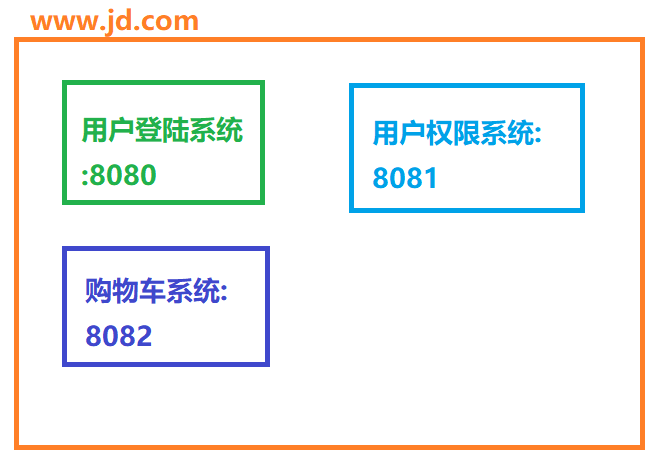
传统项目开发:



存在的问题:

1. 代码开发时,耦合性较高
2. 如果有一个模块出现问题,则可能影响整个项目.
3. 项目升级维护不易.

升级:



说明:

将项目按照功能模块的划分,拆分到多个服务器中.如果其中有某些服务器发生宕机的现象,则不会影响整个项目的运行.

##### 项目的水平拆分

晚上说…..

#### 关于分布式总结

**特点:对外是统一的.对内是独立的!!!!**

### 高并发

说明:在单位时间内,有大量用户同时访问服务器.

常识:理论值tomcat 300并发/s 实际150并发/s weblogic

支付宝:12万/s 数据库 4200万/s

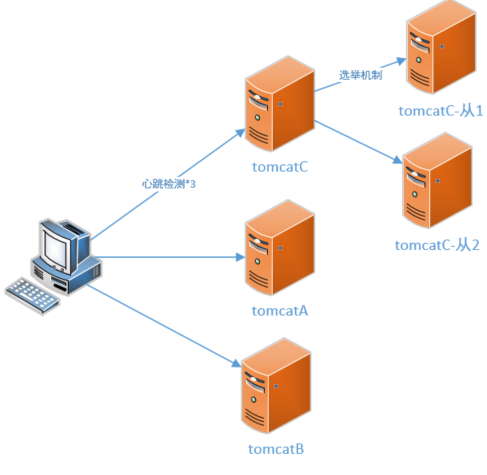
特点:高并发是现在电商网站中面临的主要的问题.

### 集群

说明:由多台tomcat共同来抗击用户大量的访问.搭建集群是抗击高并发最为有效的手段.

高级用法:

搭建集群可以有效的抗击高并发,如果出现了服务器宕机的现象,则集群内部有**高可用(HA)**的机制,可以实现故障的动态的迁移.保证服务正常的运行.



### 海量数据

说明:公司随着业务的不断的扩大,公司中的数据量逐年递增.如何在庞大的数据量中快速的检索.是现在企业中面临的主要的问题.

大数据处理.

### 网络安全

说明:保证公司服务器不被恶意的攻击和篡改.

例子:阿里(神盾局) 13亿次攻击.

360安全管家

# 搭建京淘环境

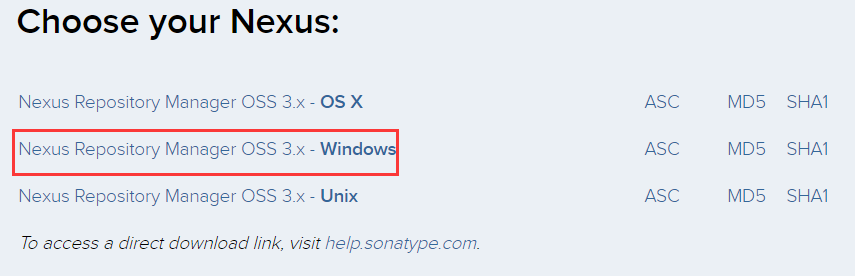
## 配置Maven

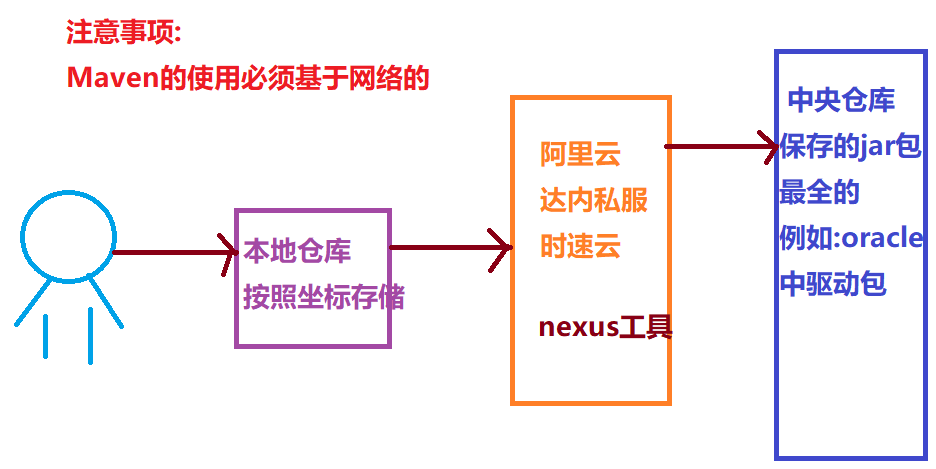
### Maven介绍

Maven是一站式的项目管理工具.可以实现jar包的自动的依赖和项目的自动的打包和部署.同时支持多种类型项目的创建.(war/jar/pom)

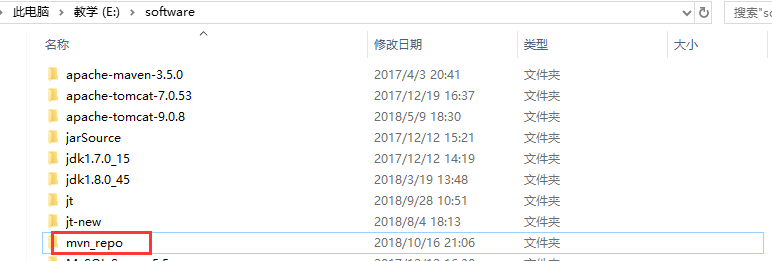
### Maven工作原理

私服网址: <https://www.sonatype.com/download-oss-sonatype>



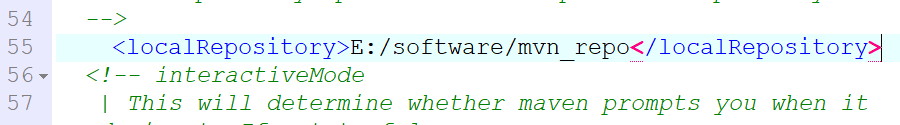


### 解压本地仓库



### 修改本地仓库

1.修改本地仓库路径



2.配置私服镜像地址

<!-- 达内内网 Maven 镜像服务器配置 -->

<mirror>

<id>nexus</id>

<name>Tedu Maven</name>

<mirrorOf>\*</mirrorOf>

<url>http://maven.tedu.cn/nexus/content/groups/public/</url>

</mirror>

3.修改jdk版本

修改配置文件中275行

<!-- 全局jdk配置，settings.xml -->

<profile>

<id>jdk18</id>

<activation>

<activeByDefault>true</activeByDefault>

<jdk>1.8</jdk>

</activation>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

<maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>

</properties>

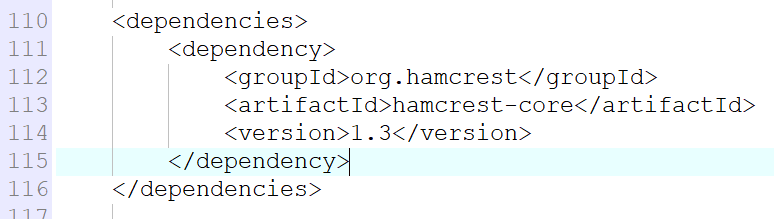
</profile>

### Maven中jar包如何依赖

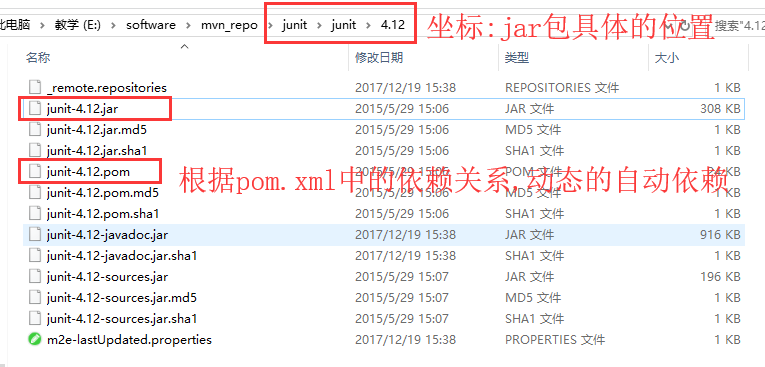
说明:Maven项目在进行添加jar包时,会自动的引入依赖的jar.

原理:

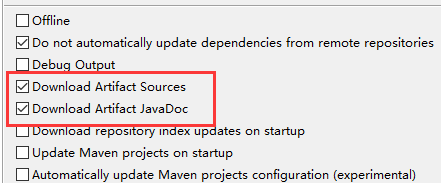
1. pom.xml文件



1. jar包未知



1. maven下载源码和doc文件



说明:当项目中引入具体的jar包时,之后Maven工具通过坐标定位jar包的位置.如果本地仓库中没有改jar包则会通过网络,到镜像中进行下载.之后将下载后的jar包引入项目.同时解析当前jar包的pom.xml文件,检测当前的jar是否有自动依赖的jar包文件,如果有则再次执行引入jar包的过程.

### Maven中引入jar包如何保证不被篡改??

1. MD5加密

特点:采用MD5加密后,结果是不可逆的. 明文--🡪密文

1. MD5Hash

特点:在MD5的基础上添加solt盐值,之后进行hash.

1. sha1 数字签名

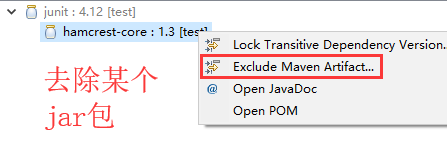
安全[哈希算法](https://baike.baidu.com/item/%E5%93%88%E5%B8%8C%E7%AE%97%E6%B3%95)（Secure Hash Algorithm）主要适用于**[数字签名](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E7%AD%BE%E5%90%8D" \t "_blank)**标准 （Digital Signature Standard DSS）里面定义的数字签名算法（Digital Signature Algorithm DSA）。对于长度小于2^64位的消息，SHA1会产生一个160位的[消息摘要](https://baike.baidu.com/item/%E6%B6%88%E6%81%AF%E6%91%98%E8%A6%81/4547744" \t "_blank)。**当接收到消息的时候，这个消息摘要可以用来验证数据的完整性**。在传输的过程中，数据很可能会发生变化，那么这时候就会产生不同的消息摘要。 SHA1有如下特性：不可以从消息摘要中复原信息；两个不同的消息不会产生同样的消息摘要,(但会有1x10 ^ 48分之一的机率出现相同的消息摘要,一般使用时忽略)。



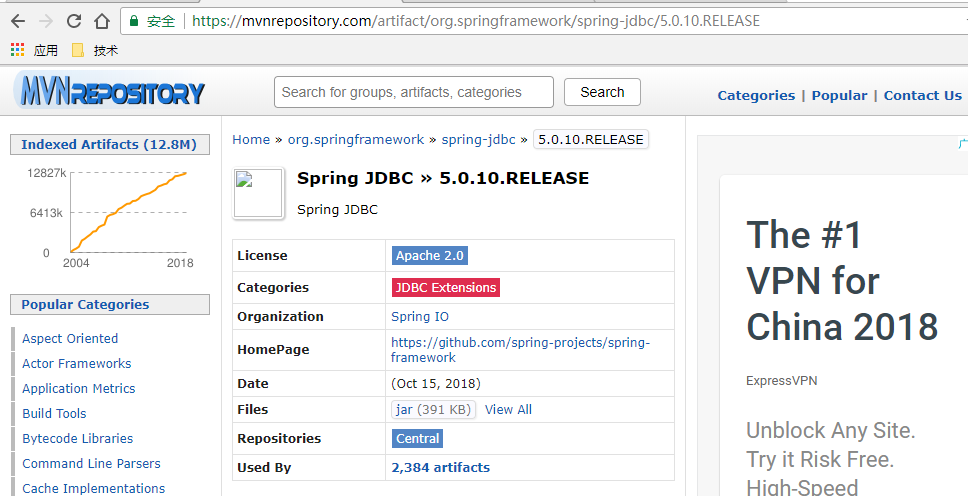
### 如何项目中jar包冲突如何解决?

1. 如果由于版本不一致导致的jar包报错.保留高版本jar包,去除低版本jar包.

思路:

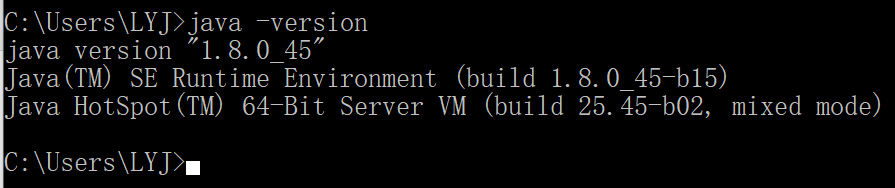


1. jar包版本如何保证稳定.

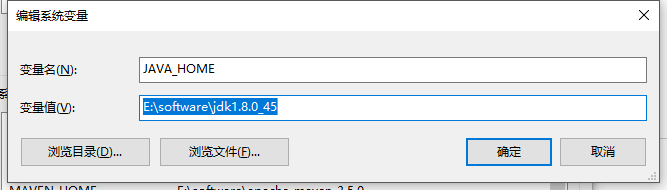


## 京淘环境要求

### JDK环境变量

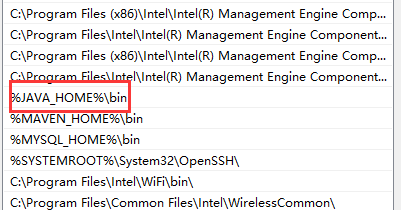


### 修改JDK环境变量



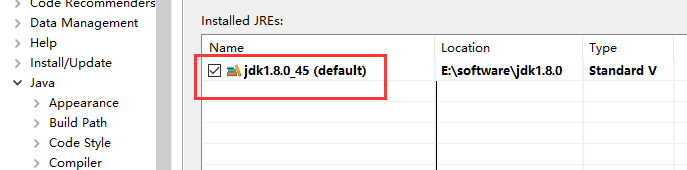
必须配置path:

;%JAVA\_HOME%/bin

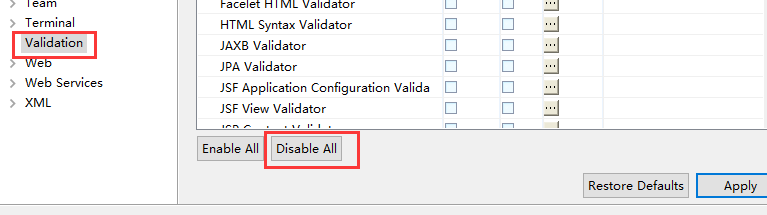


### 修改eclipse环境

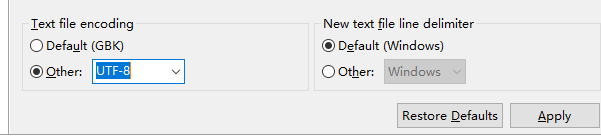
1. 配置JDK引入



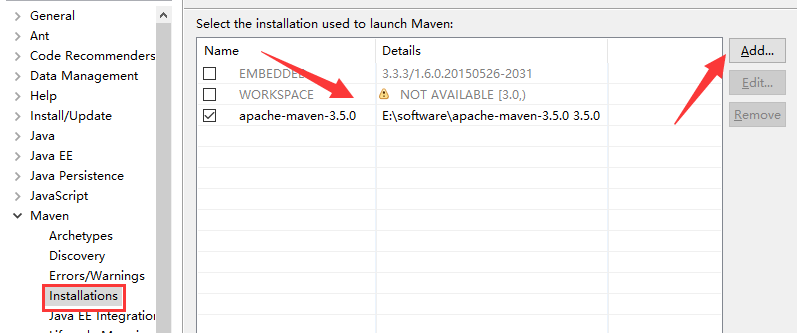
1. 关闭校验



1. 修改字符集

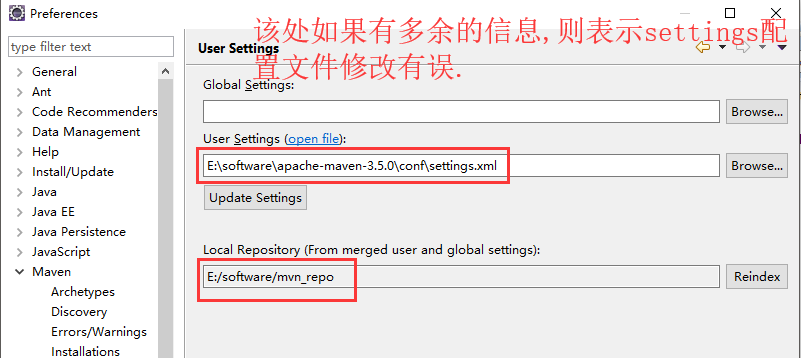


1. 配置Maven



1. 添加maven环境配置文件

说明:因为maven用户默认使用的级别是user settings



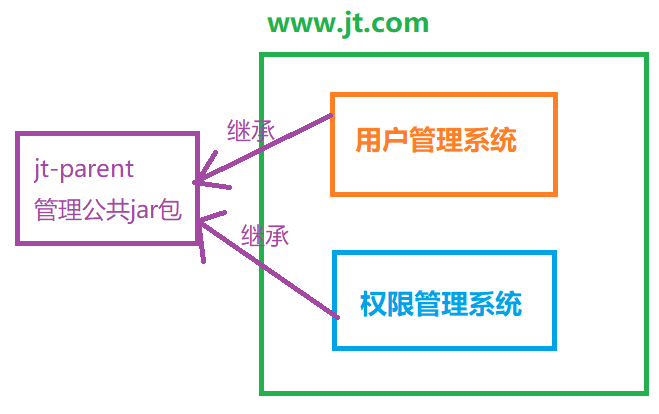
补充:

因为eclipse整合maven中存在问题.所以导入项目时,只拷贝src文件.和pom.xml即可.

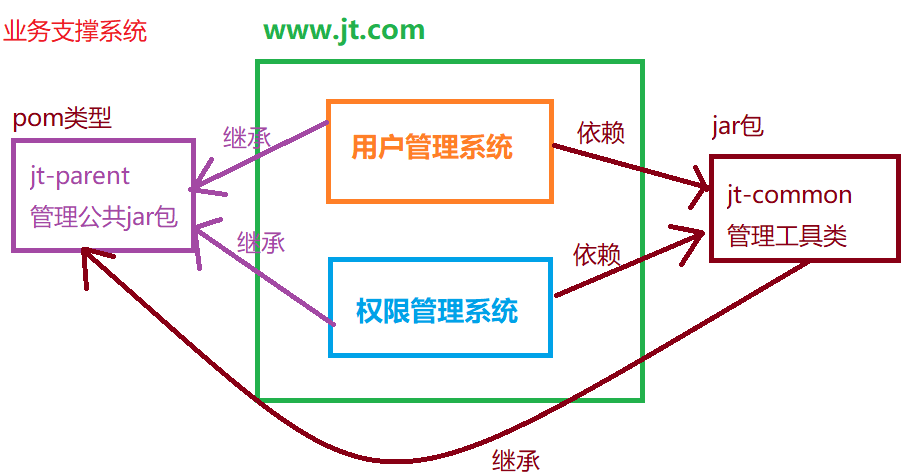
## 京淘环境搭建

### 项目拆分后jar包如何管理

说明:采用jt-parent项目管理jar包,通过一个项目维护jar包的版本.如果将来需要对jar包进行升级,只需修改jt-parent项目即可.



### 项目拆分后工具类如何管理



说明:根据分布式的思想,将项目进行垂直拆分,同时要想让整个服务调用松耦合,必须需要业务支撑系统的支持.一个jt-parent负载管理jar包,一个jt-common管理工具类.

### 京淘架构设计

|  |  |  |
| --- | --- | --- |
| 序号 | 项目名称 | 主要作用 |
| 1 | jt-manage | 京淘后台管理系统 负责商品的更新和维护 |
| 2 | jt-web | 京淘前台系统，主要负责商品的展现 |
| 3 | jt-cart | 京淘购物车系统，负责用户购物车信息维护 |
| 4 | jt-sso | 京淘单点登陆系统，实现session数据共享 |
| 5 | jt-order | 京淘订单系统，负责订单维护 |
| 6 | jt-search | 京淘项目的全文检索 |
| 7 | jt-rabbitMQ | 消息队列系统 |
| 8 | jt-parent | 负责jar包的管理和依赖 |
| 9 | jt-common | 管理工具类文件 |

## 构建jt-parent项目

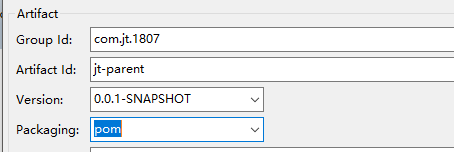
### 项目创建的规则

Jt-parent pom类型

Jt-common jar类型

Jt-manage war类型

### 创建项目



### 导入jar包

<!-- 集中定义依赖版本号 -->

<properties>

<junit.version>4.10</junit.version>

<spring.version>4.1.3.RELEASE</spring.version>

<mybatis.version>3.2.8</mybatis.version>

<mybatis.spring.version>1.2.2</mybatis.spring.version>

<mybatis.paginator.version>1.2.15</mybatis.paginator.version>

<mysql.version>5.1.32</mysql.version>

<bonecp-spring.version>0.8.0.RELEASE</bonecp-spring.version>

<druid.version>1.0.29</druid.version>

<mapper.version>2.3.2</mapper.version>

<pagehelper.version>3.4.2</pagehelper.version>

<jsqlparser.version>0.9.1</jsqlparser.version>

<slf4j.version>1.6.4</slf4j.version>

<jstl.version>1.2</jstl.version>

<servlet-api.version>2.5</servlet-api.version>

<jsp-api.version>2.0</jsp-api.version>

<joda-time.version>2.5</joda-time.version>

<commons-lang3.version>3.3.2</commons-lang3.version>

<commons-fileupload.version>1.3.1</commons-fileupload.version>

<jackson.version>2.4.2</jackson.version>

<httpclient.version>4.3.5</httpclient.version>

<jedis.version>2.6.2</jedis.version>

</properties>

<dependencies>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<!-- Spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>${mybatis.spring.version}</version>

</dependency>

<dependency>

<groupId>com.github.miemiedev</groupId>

<artifactId>mybatis-paginator</artifactId>

<version>${mybatis.paginator.version}</version>

</dependency>

<!-- MySql -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>${mysql.version}</version>

</dependency>

<!--引入阿里druid监控 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>${druid.version}</version>

</dependency>

<!-- 通用Mapper -->

<dependency>

<groupId>com.github.abel533</groupId>

<artifactId>mapper</artifactId>

<version>${mapper.version}</version>

</dependency>

<!-- 分页插件 -->

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>${pagehelper.version}</version>

</dependency>

<dependency>

<groupId>com.github.jsqlparser</groupId>

<artifactId>jsqlparser</artifactId>

<version>${jsqlparser.version}</version>

</dependency>

<!-- 连接池 -->

<dependency>

<groupId>com.jolbox</groupId>

<artifactId>bonecp-spring</artifactId>

<version>${bonecp-spring.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

<!-- Jackson Json处理工具包 -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>${jackson.version}</version>

</dependency>

<!-- httpclient -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>${httpclient.version}</version>

</dependency>

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpmime</artifactId>

<version>4.3.1</version>

</dependency>

<!-- 消息队列 -->

<dependency>

<groupId>com.rabbitmq</groupId>

<artifactId>amqp-client</artifactId>

<version>3.5.1</version>

</dependency>

<dependency>

<groupId>org.springframework.amqp</groupId>

<artifactId>spring-rabbit</artifactId>

<version>1.4.0.RELEASE</version>

</dependency>

<!-- JSP相关 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>${servlet-api.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<version>${jsp-api.version}</version>

<scope>provided</scope>

</dependency>

<!-- 时间操作组件 -->

<dependency>

<groupId>joda-time</groupId>

<artifactId>joda-time</artifactId>

<version>${joda-time.version}</version>

</dependency>

<!-- Apache工具组件 -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>${commons-lang3.version}</version>

</dependency>

<!-- 文件上传组件 -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>${commons-fileupload.version}</version>

</dependency>

<!-- jedis -->

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>${jedis.version}</version>

</dependency>

<!--添加spring-datajar包 -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-redis</artifactId>

<version>1.4.1.RELEASE</version>

</dependency>

<!-- 字符加密、解密 -->

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.9</version>

</dependency>

<!-- 数据校验 -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

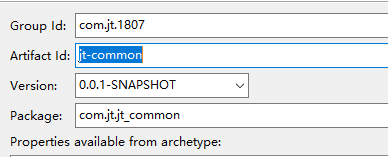
<version>5.1.3.Final</version>

</dependency>

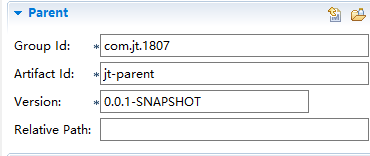
</dependencies>

## 构建jt-common

### 创建项目



### 添加继承



### 关于子父级依赖jar包问题

<dependencies>

<!--说明:

如果父级中定义的jar包的版本和子项目中定义的jar包版本冲突了,

则以自己定义的版本为准.该功能一般在解决复杂的jar包冲突时使用.

-->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

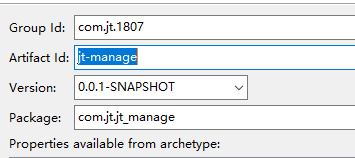
### 关于工具类修改问题

说明:



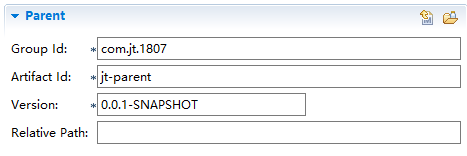
## 构建jt-manage

### 构建项目



问题说明:如果创建的项目中只有一个资源文件夹,缺少main/test则修改JDK即可.

### 添加继承



### 添加依赖



## Tomcat插件

### Tomcat服务器引入后缺点

1. tomcat使用一段时间后启动越来越慢.
2. Tomcat在运行期,会生成一些临时文件.这些文件一般保存到硬盘
3. 如果需要在window/Linux中启动多台tomcat服务器.必须修改对应的端口号 8080/8005/8009

### 引入tomcat插件

<!--通过maven启动tomcat插件 -->

<build>

<plugins>

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<port>8091</port>

<path>/</path>

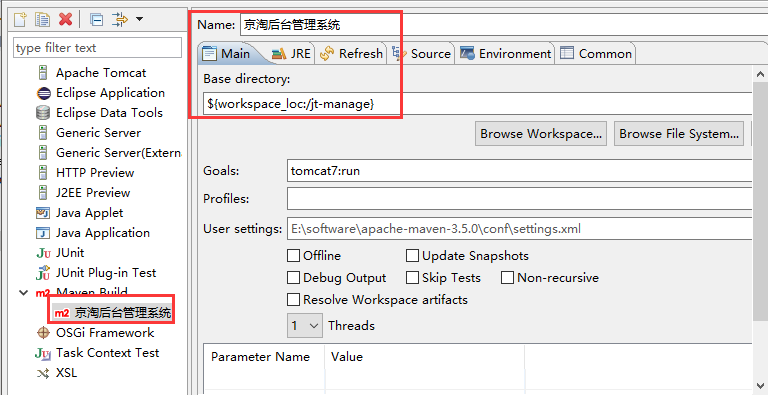
</configuration>

</plugin>

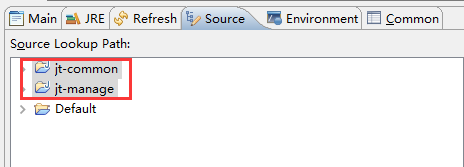
</plugins>

</build>

### 配置tomcat启动命令

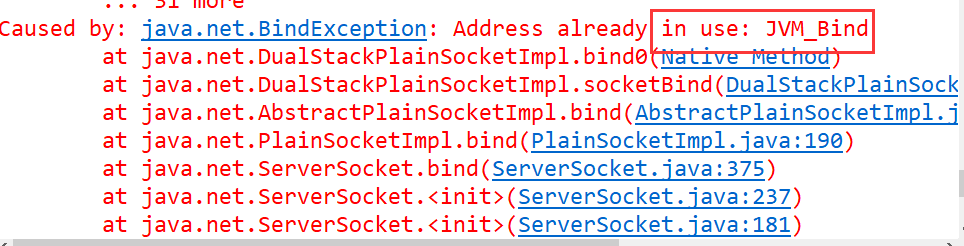


添加源码:



### 端口被占用问题

如果出错,表示端口被占用.需要重启服务.



## 水平拆分

### 原理

说明:因为编辑超大型项目时,开发时会造成一个业务逻辑可能有多个人共同开发的现象.因为这样的开发编程耦合性较高,开发不易,所以进行项目的水平拆分.

根据项目的层级进行拆分.

View----Controller(war)----接口/Service(jar)----dao(jar)/mapper----pojo(jar)

### 具体业务场景

Web-aa(聚合项目)

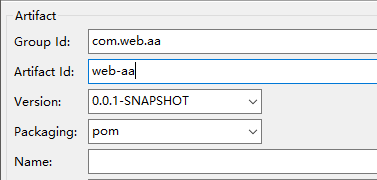
Web-aa-controller---war

Web-aa-service ---jar

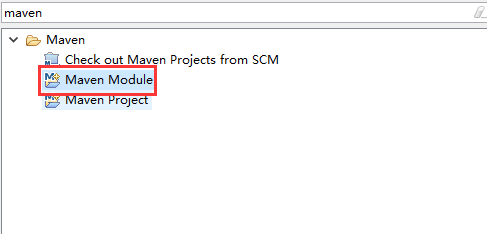
Web-aa-mapper ---jar

Web-aa-pojo ---jar

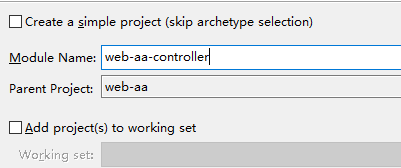
### 创建父级项目



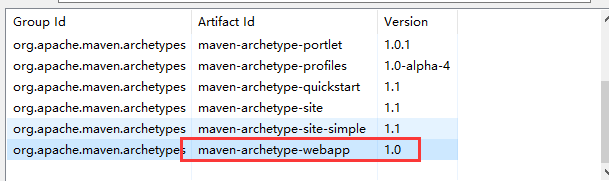
2.选择项目创建方式



3.新建项目名称



4.选择项目类型



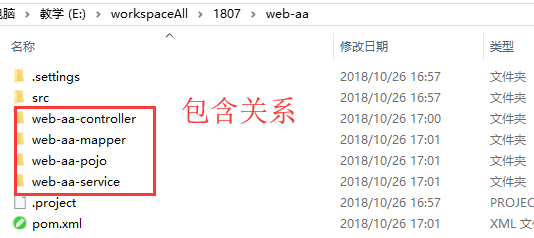
### 项目如何调用

依赖顺序:

Web-aa-controller---依赖-🡪Service—依赖--🡪mapper-依赖--🡪pojo

因为maven中依赖是由传递性的.

### 项目格式



打包时只需要将web-aa打包即可.

