刘昱江 中心名称 + 班级号 + 名字 527161772

刘老师: 错误描述 +照片

# 电商项目概述

## 电商的行业特点

### 高并发

概念:用户在单位时间内,有大量的用户请求访问服务器.

标准:tomcat服务器 150次/秒

日活量:单台tomcat 700万左右(中/大)台 一天

项目设计指标:

1.并发压力:2000个并发 15台

通用:软件不行加硬件

### 分布式

核心:根据特定的需求,将**大型项目**进行**拆分**

#### 分布式计算

概念:将一项任务,拆分为多份,由多个节点/线程同时执行(运算),最终获得结果的过程.  
例子: 一个人干一项工作需要10个小时,由10个人同时完成,需要1个小时.

处理的内容不同!!!!!

对比: 一个人完成一项任务10个小时完成,由10个人分别执行10个小时 (10个任务)

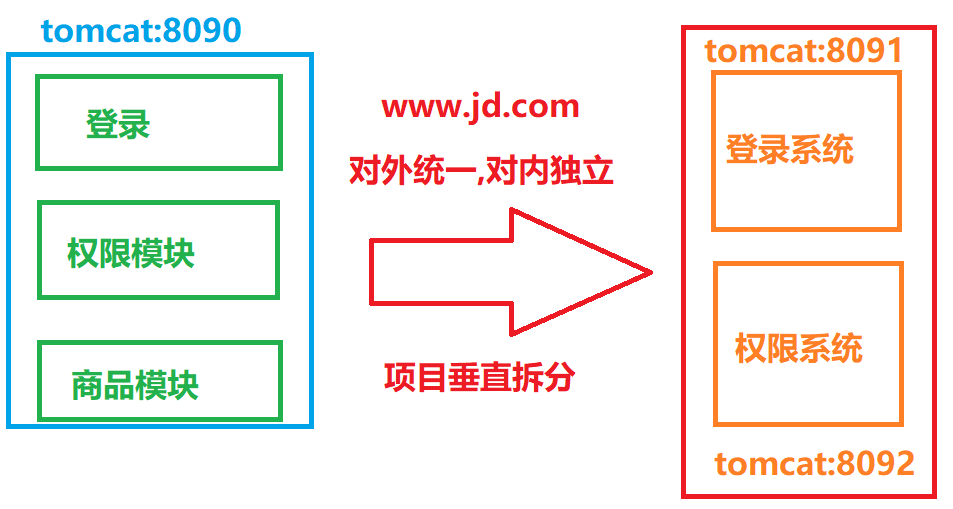
#### 分布式系统

前提:传统的单体项目,在当今已经不可使用.如果单体项目中某一个模块发生故障,则会影响整合服务(架构的耦合性高).

解决问题:架构中的耦合性

##### 项目垂直拆分

概念:将大型的项目,根据**功能模块**将项目进行拆分



##### 项目水平拆分

概念:将项目按照层级(controller/service/mapper/pojo/view)进行拆分

案例:

开发taobao-manage(pom) 父级

Taobao-manage-controller(war)

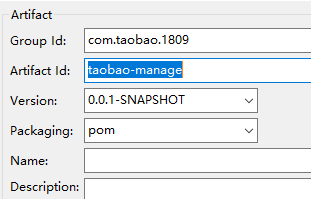
Taobao-manage-service(jar)

Taobao-manage-mapper(jar)

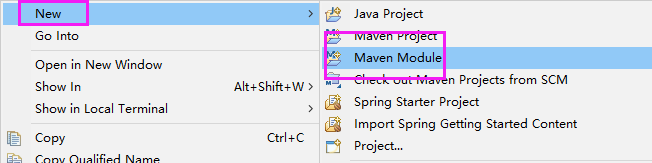
Taobao-manage-pojo(jar)

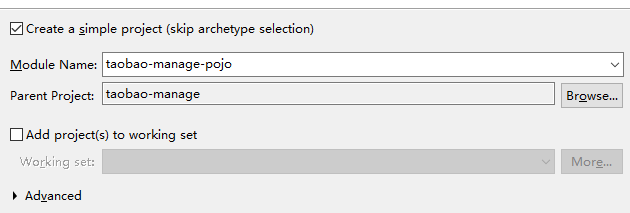
搭建步骤:

1. 新建pom类型的项目

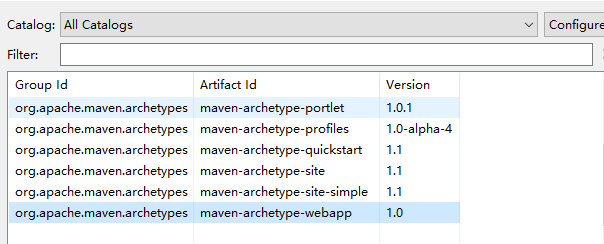


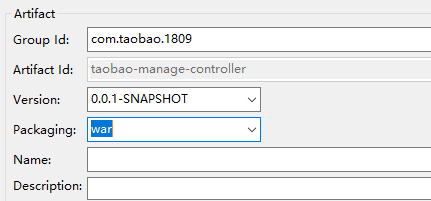
1. 创建maven-module





1. 分别创建mapper/service
2. 创建controller 选择webapp



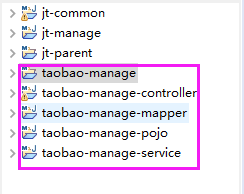


1. 项目文件结构



##### 聚合项目程序调用

Controller---依赖---🡪Service---🡪---Mapper---🡪pojo



只需要将taobao-manage打包,全部项目自动打包

[INFO] taobao-manage SUCCESS [ 0.318 s]

[INFO] taobao-manage-pojo SUCCESS [ 0.946 s]

[INFO] taobao-manage-mapper SUCCESS [ 0.042 s]

[INFO] taobao-manage-service SUCCESS [ 0.040 s]

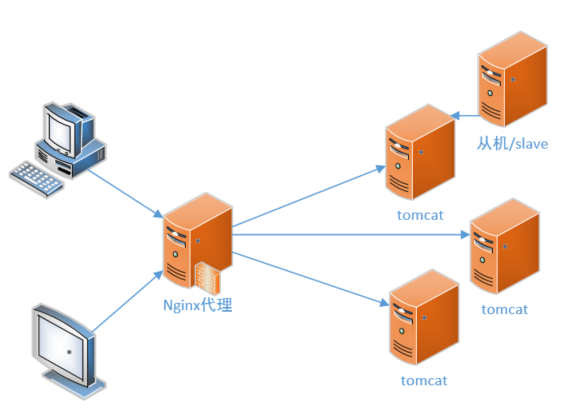
[INFO] taobao-manage-controller SUCCESS [ 0.330 s]

[INFO] BUILD SUCCESS

### 集群(高可用)

概述:集群是抗击高并发最有效的手段,为了为用户提供可持续的服务,一般都会实现高可用(HA是High Available缩写)

总结:**根据分布式思想,将项目进行拆分.拆分后一般会搭建多个集群,共同抗击高并发.**



### 安全性

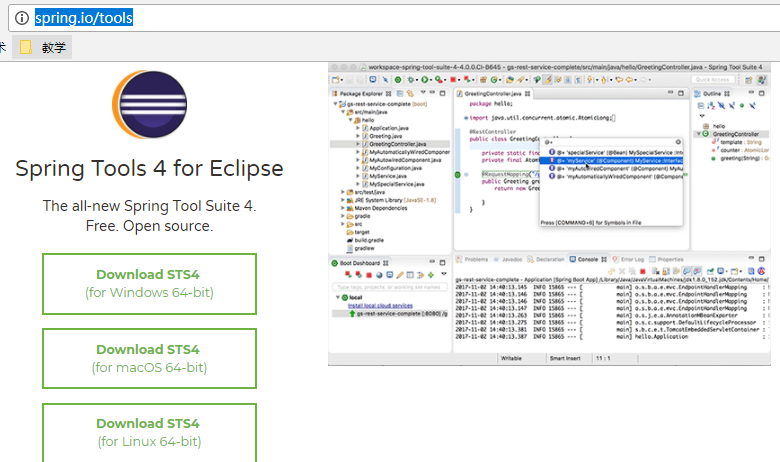
说明:软件设计时,必须要有安全性的考虑.公司网络安全部署.

阿里:神盾局 每天要防止16亿次的网络攻击!!!!

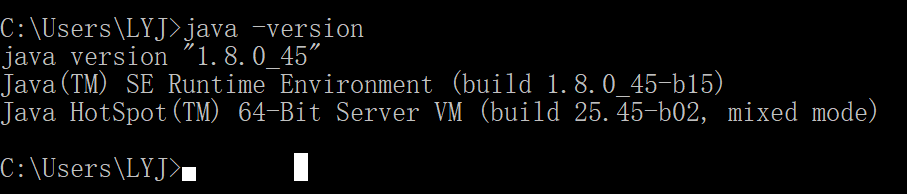
## 环境搭建

### IDE下载

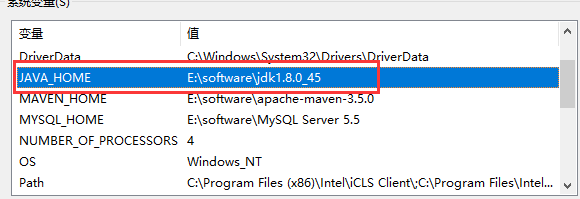
<URL:http://spring.io/tools>

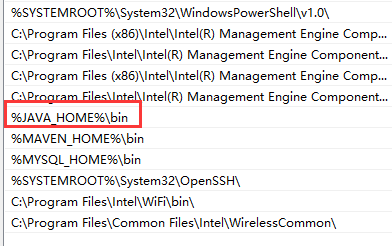


### JDK配置



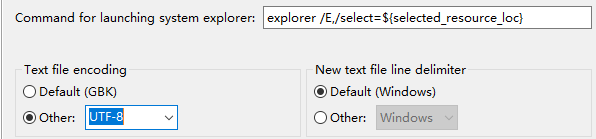
2.检查环境变量配置



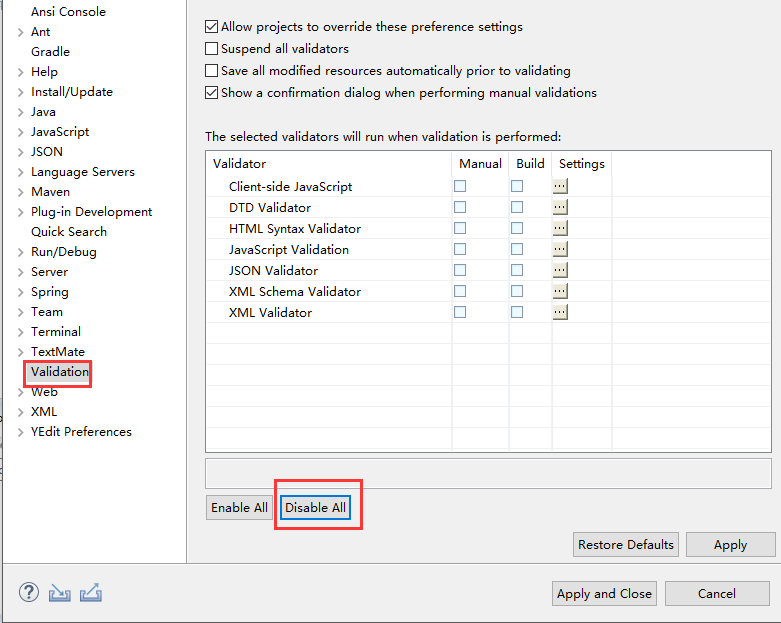


WIN7: path:“;%JAVA\_HOME%/bin” 如果不生效,将java配置放到path的第一个.

### 切换字符集



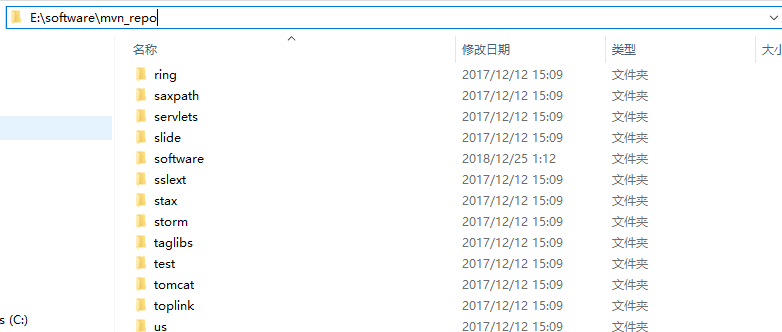
### 关闭校验



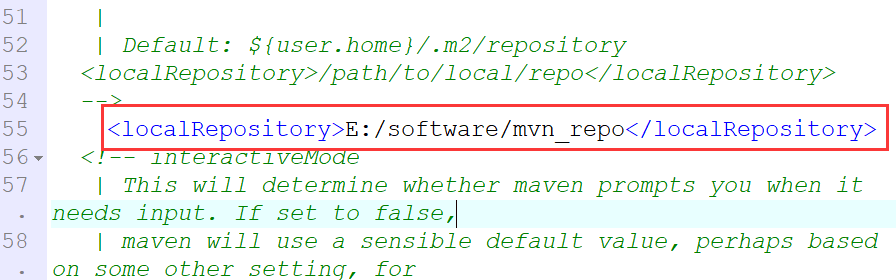
### Maven配置

1. 导入本地仓库

在本地磁盘中添加英文路径,之后解压课前资料中的maven仓库



1. 修改settings中的本地仓库路径



1. 配置镜像

<!-- 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>-->

<!--阿里云镜像-->

<mirror>

<id>nexus-aliyun</id>

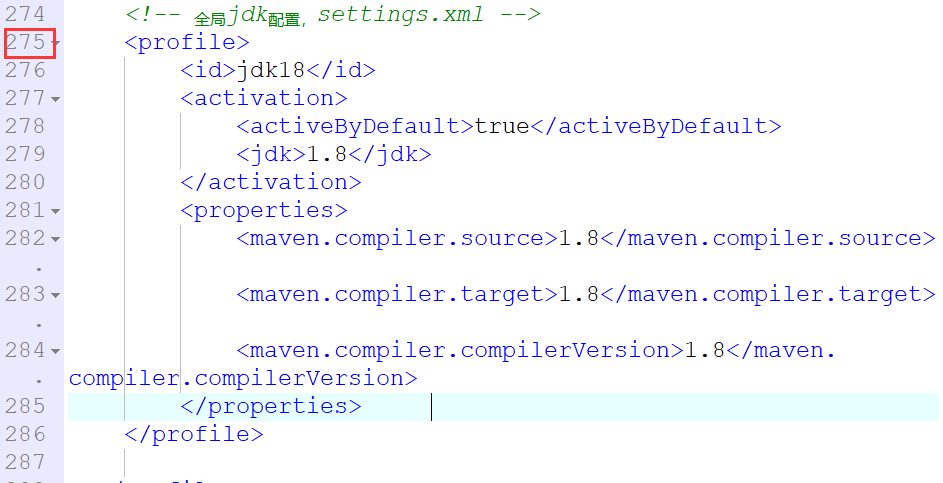
<mirrorOf>\*</mirrorOf>

<name>Nexus aliyun</name>

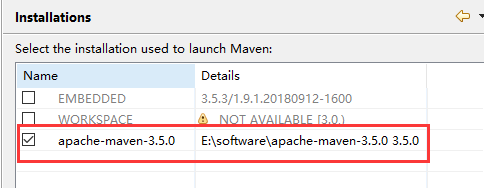
<url>http://maven.aliyun.com/nexus/content/groups/public</url>

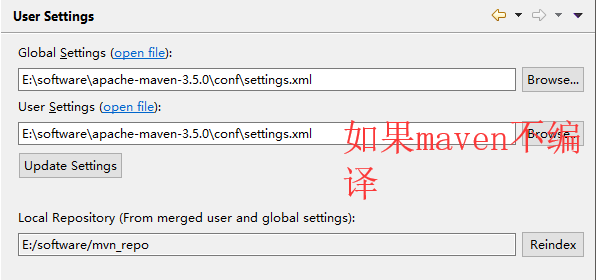
</mirror>

1. 修改JDK配置



5.引入maven 动态更新即可





## 京淘项目划分

### 项目划分

|  |  |  |
| --- | --- | --- |
| 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 |

### 项目拆分后-jar包如何管理

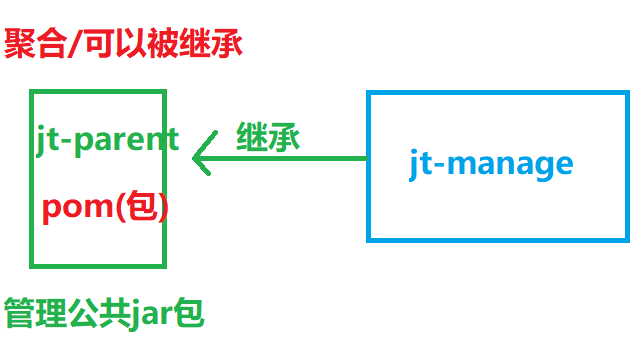
问题:如果项目采用分布式的方式进行构建,则每个模块都能够正常的独立运行,所需的jar包都是独立的.如果将来jar包升级,则需要多次修改.

1. jar包冲突.
2. Jar包更新版本控制,代码的稳定性弱

想法:能否让jar包统一管理!!!

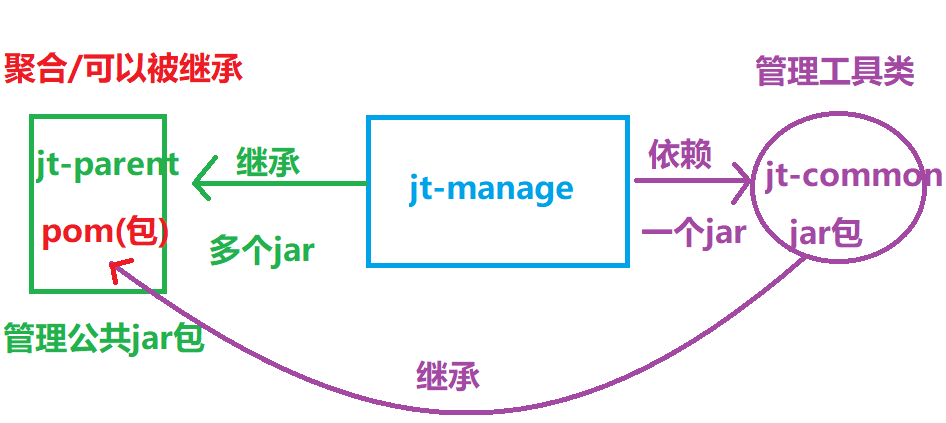
实现:在项目中创建一个pom类型的项目,该项目管理全部的公用jar包.如果将来项目升级,只需要修改pom中的jar包的版本即可.

其他项目只需要**继承**该**parent**即可



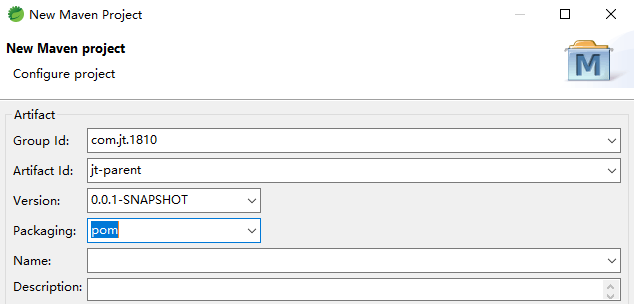
### 项目拆分后-工具类如何管理

工具类由架构师,或者高级程序员编辑,将工具类打成jar包后,被其他项目依赖即可.



## 项目创建-jt-parent

### 创建项目



### 引入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>

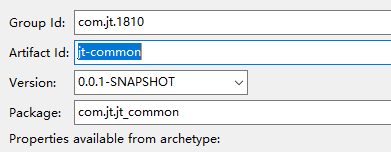
<version>5.1.3.Final</version>

</dependency>

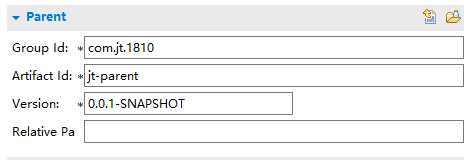
</dependencies>

## 创建工具类jt-common

### 创建项目

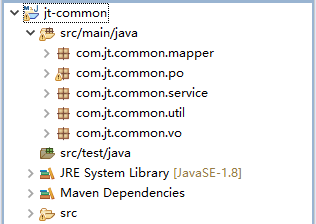


### 添加继承



### 导入工具类

说明:将课前资料中的jt-common的src文件导入项目.



### 将common打成jar包

1. 将jt-parent打包

Installing E:\workspaceAll\1810\jt-parent\pom.xml to E:\software\mvn\_repo\com\jt\1810\jt-parent\0.0.1-SNAPSHOT\jt-parent-0.0.1-SNAPSHOT.pom

根据jt-parent的坐标,将打包后的文件保存到执行的坐标路径下.

1. 将jt-common打包

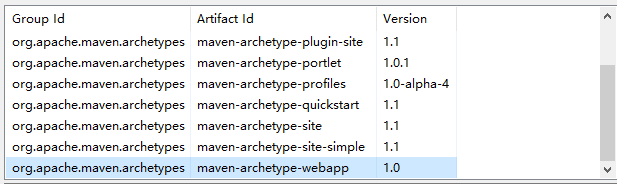
Installing E:\workspaceAll\1810\jt-common\target\jt-common-0.0.1-SNAPSHOT.jar to E:\software\mvn\_repo\com\jt\1810\jt-common\0.0.1-SNAPSHOT\jt-common-0.0.1-SNAPSHOT.jar

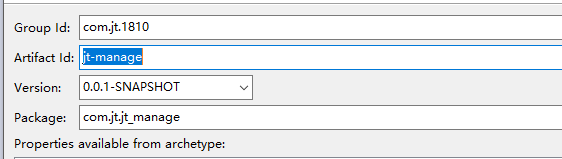
[INFO] Installing E:\workspaceAll\1810\jt-common\pom.xml to E:\software\mvn\_repo\com\jt\1810\jt-common\0.0.1-SNAPSHOT\jt-common-0.0.1-SNAPSHOT.pom

[INFO]

## 创建jt-manage

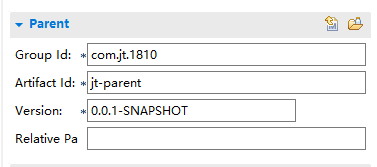
### 选择骨架





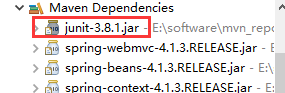
### 继承父级项目

继承父级项目实现依赖



### 继承引发问题

1. 说明自己引入jar包,与父级中jar包版本不一致!!!



1. maven中的jar有依赖的传递性.

A---🡪B----🡪C A的jar包同时依赖B和C

### Maven加载jar包的原理

例子:

<dependency>

<groupId>junit</groupId>

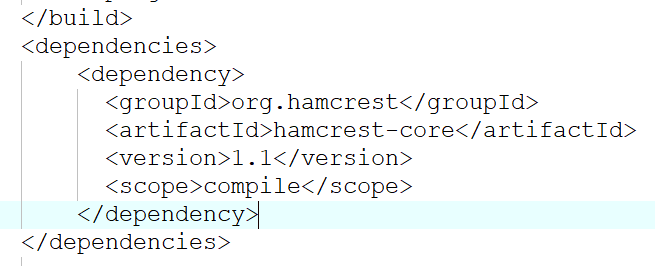
<artifactId>junit</artifactId>

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

<scope>test</scope>

</dependency>

当通过pom.xml文件添加jar包时,根据依赖指定的坐标进行查找.查找到jar包时,再次解析当前jar包的pom.xml文件,之后根据解析再次加载依赖包.

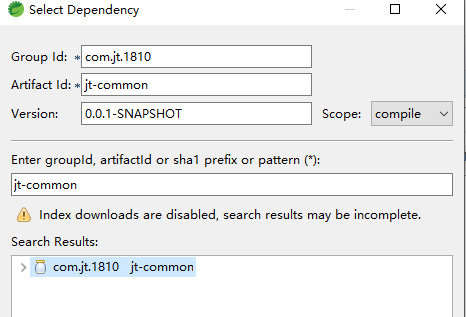


### Maven如何保证数据有效性

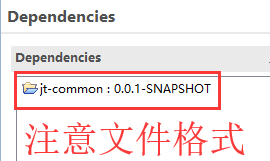
1. MD5 不可逆 +加盐值 计算能力---区块链(局域网)
2. SHA1

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)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)。**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)。

### 添加依赖



引入后效果:



## Tomcat插件介绍

### 原生tomcat缺点

1. 如果需要搭建tomcat集群,需要多台tomcat,这时需要修改配置文件

8005/8080/8009(ajp) 使用是不方便

1. 将项目发布到tomcat/webapps中,如果部署多个服务项,那么则tomcat越用越卡(临时文件太多)

### 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>

<!--项目发布根目录 -->

<path>/jt</path>

</configuration>

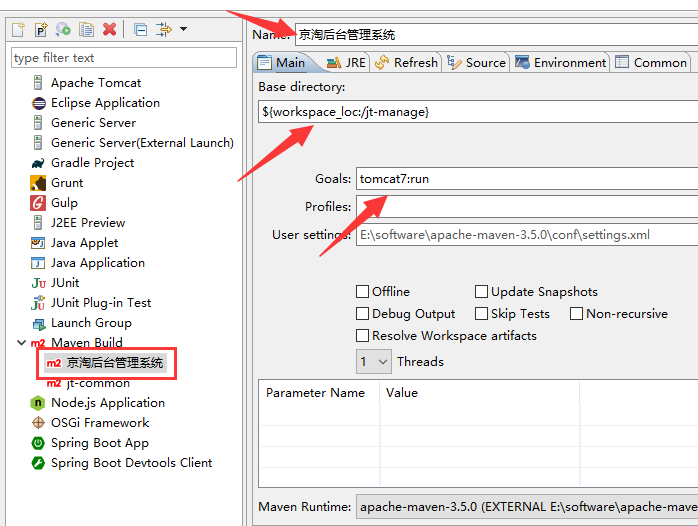
</plugin>

</plugins>

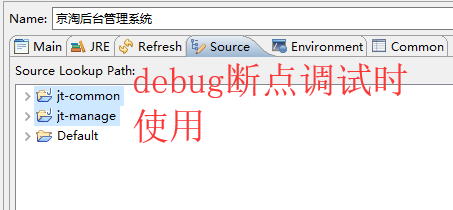
</build>

### 插件配置项

1.配置插件



2.配置source



1. 项目启动说明

通过RUNAS启动tomcat后,程序即可正常运行,如果要关闭tomcat,只需要终止进程即可.

