# 刘昱江

要求:

1.听思路

2.根据自己的思路敲代码

3.项目阶段单点登录.购物车/订单系统代码特别的多

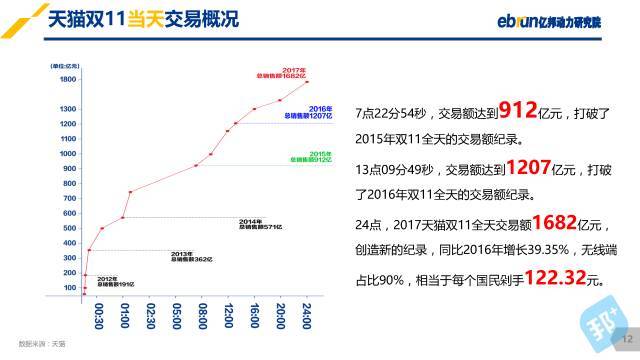
4.00-7000(1500+2000+信用卡) 1300左右 500-1000

5.527161772 地区+班级+姓名

现场互动良好

# 京淘项目概述

京淘项目是将当下电商网站中主流的项目拿过来,学习当下最新最热的电商网站技术



## 电商网站特点

### 高并发

交易总价为1682亿元，5分22秒支付峰值25.6万笔/秒。数据库的处理峰值4200万/秒。

概念:单位时间内服务器处理的大量的数据

说明:淘宝网的支付数量在25.6/万笔每秒这个压力已经很大.项目如何搭建面临着很大的考验.

### 分布式

分布式分为2大方面

1. 分布式计算

说明:假设一个人完成一项任务需要10个小时.如果由10个人**公共**完成,则需要1个小时.

实际应用:大数据

效果:能够有效的缩短任务的执行时间

1. 分布式系统

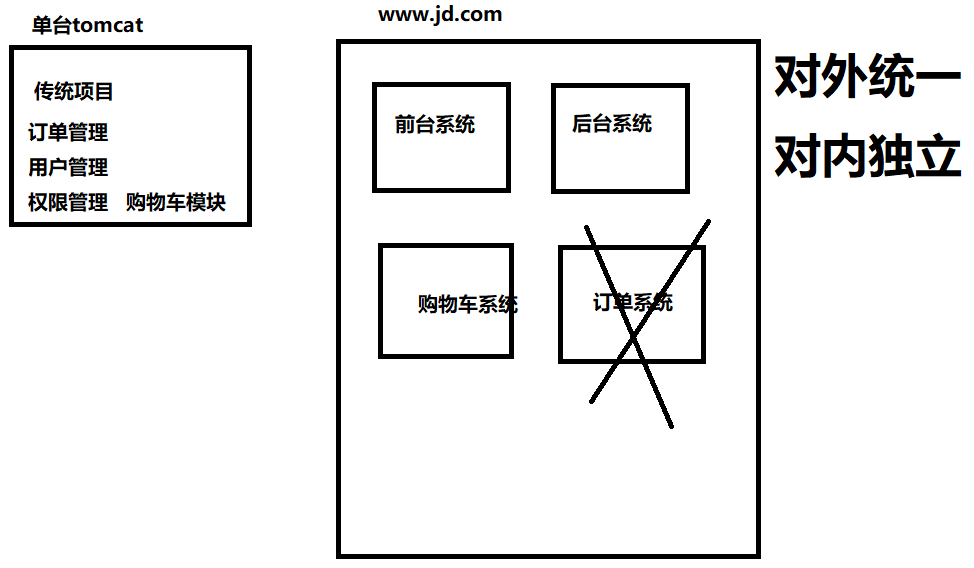
大型项目的开发中有很多的子系统 /支付系统/订单系统/后台系统/购物车系统

传统系统:将一个项目用单台tomcat进行部署.如果出现问题将影响整个网站.

分布式系统:

将传统的项目的模块进行系统化的处理.让每一个模块行形成一个单独的系统.便于部署.如果某个系统出现问题不会影响整个项目运行.

特点:对外统一 对内独立



### 集群部署

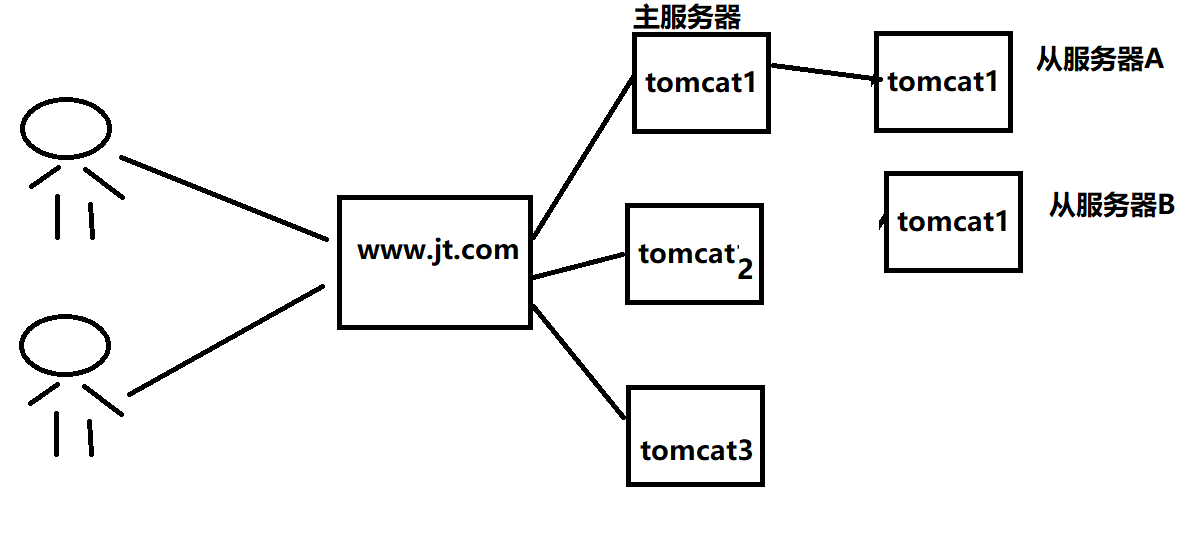
概念:将**相同**的项目部署到多台服务器中

需求:

现在要求实现20万/秒并发

现在准备10000台服务器.那么每台服务器的并发压力在20.服务器很容易完成这项任务.

衍生:集群的高可用技术(HA)



说明:采用了高克用的技术后.能够有效的防治服务器宕机.能够采用后备服务器.自动的实现服务器切换.用户体验较好

问题:

集群和分布式区别:

1. 集群:

是将**相同的项目**部署到多台服务器,完成相同任务(业务)

2分布式:

将**不同的项目**部署到不同的服务器中.完成的是不同的业务逻辑.让项目松耦合.

### 海量数据

描述:

随着互联网企业的发展数据增长速度很快.例如商品是有限的.订单是无限的

问题:电商网站中如何实现数据的快速检索.

方法:分库分表 主表中只存储近2周到一年的数据.将之间的购买记录存储到历史表中.

衍生:大数据方向

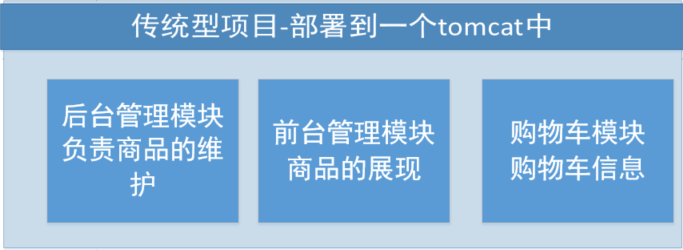
说明:随着软件的不断的发展,数据不断的积累,如果快速进行处理.以及从海量数据中获取有价值的信息.是大数据只要研究的内容.

# 大型项目拆分方式

说明:

大型项目通常是由团队开发.并且其中的业务逻辑相对复杂.并且工程数量较多.需要整个团队协同配合才能完成

## 传统项目开发的问题



说明:传统型项目中,将全部的业务都写到一个项目中,用一台tomcat部署.这时如果某一项业务操作时造成了服务器宕机.则全部项目都不能正常访问.用户体验较差.

## 项目的垂直拆分

概念:将原有项目的模块系统化,这样的拆分方式叫做垂直拆分.

图例:



说明:大型项目将模块系统化之后方便了系统的开发和管理.如果某一个业务有问题.不会对整个项目造成太大的影响.实现了软件架构之间的松耦合.

项目的垂直拆分就是分布式思想的应用.

问题1:

如果是多个项目共同开发,jar包如何管理

某些项目中的jar包文件是自己所特有的,某些jiar包文件是公共的.那么jar包如何管理成为分布式系统的主要难题?

解决方案:

将一些公共的jar包文件进行抽取,抽取为jt-parent项目.其他的系统只需要继承jt-parent项目后,就可以获取jar包文件.这样做方便jar包文件的管理和升级

问题2:

大型项目开发其中需要用到一共工具类,如果每个项目中都单独引入工具类,那么会造成代码维护的不方便.

解决方案:

将工具类进行抽取.抽取为单独的项目.如果开发时需要用到工具类.只需要依赖工具类jar包文件即可.

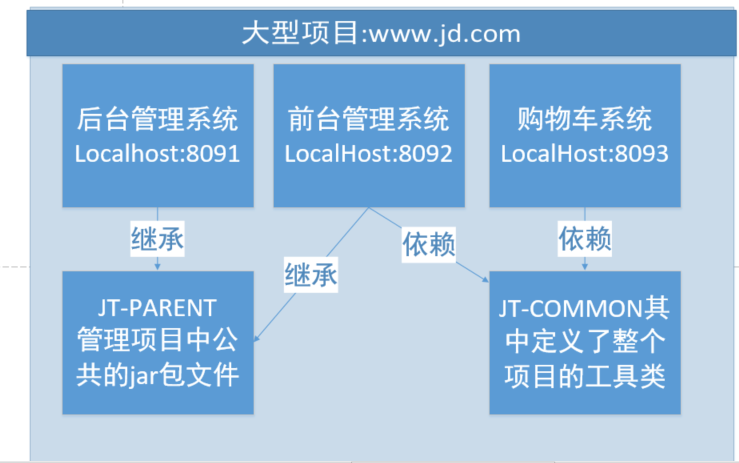
面试问题:

请简述大型项目构建的注意事项?

答案:

大型项目构建时,需要将特定的模块进行系统化.这样做可以将系统性能大大提升.即使其中一个项目出现了问题,也不影响其他项目的使用.方便系统的管理和维护.(开发的耦合性)

但是项目拆分完成之后会有jar包和工具类的管理性问题.需要再次进行抽取.将工具类项目抽取到jt-commons项目中.将jar包文件抽取到jt-parent项目中.这样管理管理和维护.



# 京淘项目的架构设计

## 项目划分

京淘项目一共分为8大子系统,其中有6大系统是业务模块拆分后的结果.jt-parent和jt-common是业务的支撑系统.

|  |  |  |
| --- | --- | --- |
| 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-parent | 负责jar包的管理和依赖 |
| 8 | jt-common | 管理工具类文件 |

## 京淘技能点

作业:利用课余时间查询技术到底是做什么的?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 知识点 | 重要程度 | 难度系数 | | 掌握程度 |
| 1. 1 | 业务：商品分类、商品、前台、权限单点登录、购物车、订单、商品全文检索 | ★★★★★ | | 1 | 熟练 |
|  | Spring、SpringMVC、MyBatis主流框架 | ★★★ | | 1 | 熟练 |
|  | Maven 一键构建继承和聚合 | ★★ | | 1 | 熟练 |
|  | PowerDesinger表设计及优化 | ★★★★ | | 3 | 会用 |
|  | 富客户端EasyUI、KindEditor图文控件 | ★ | | 1 | 会用 |
|  | RESTFul 访问方式 | ★★★★★ | | 1 | 熟练 |
|  | 通用Mapper插件，自动生成调用代码 | ★ | | 1 | 会用 |
|  | Nginx 负载均衡/反响代理 | ★★★★★ | | 1 | 精通 |
|  | Tomcat集群 | ★★★ | | 1 | 熟练 |
|  | Linux/CentOS 操作系统 | ★★★ | | 1 | 熟练 |
|  | MySQL主从复制，Amoeba读写分离 | ★★★★★ | | 2 | 精通 |
|  | Redis 缓存服务器，分片，哨兵，集群高可用 | ★★★★★ | | 1 | 精通 |
|  | JSONP浏览器跨域 | ★★ | | 2 | 会用 |
|  | HttpClient系统间调用 | ★★★ | | 1 | 熟练 |
|  | 石英钟定时任务 | ★ | | 1 | 会用 |
|  | RabbitMQ 消息队列 | ★★★ | | 3 | 熟练 |
|  | Lucene、Solr搜索 | ★★★★ | | 3 | 熟练 |
|  | Jsoup爬虫技术 | ★★★★ | | 3 | 熟练 |
|  | Dubbo框架 | ★★★★ | | 4 | 熟练 |
|  | Docker容器技术 | ★★★★ | | 4 | 熟练 |

# 京淘网络拓扑图



说明:由于业务逻辑众多,课上不一一介绍,后期一一学习.

# 搭建京淘后台架构

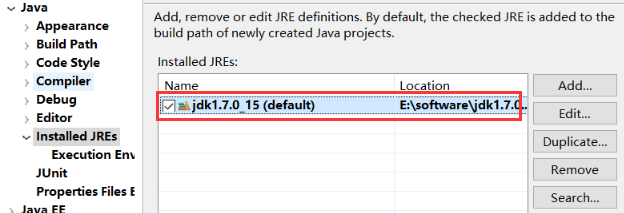
说明:

京淘后台项目的搭建需要搭建一个业务系统和2个业务支撑系统

1. jt-manage 构建web项目
2. jt-parent 构建pom项目
3. jt-common 构建java项目

## 切换工作空间

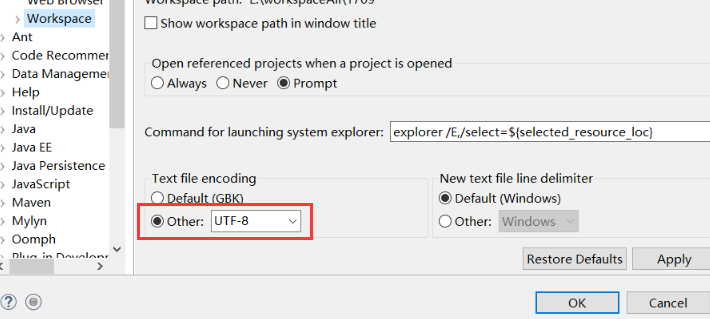
### 配置系统JDK



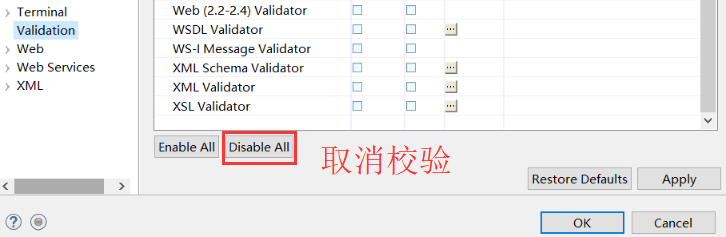
将系统的环境变量配置为JDK1.7 防治后期由于版本问题冲突

### 修改字符集

将系统的字符集修改为UTF-8,方便后期导入代码时不会乱码



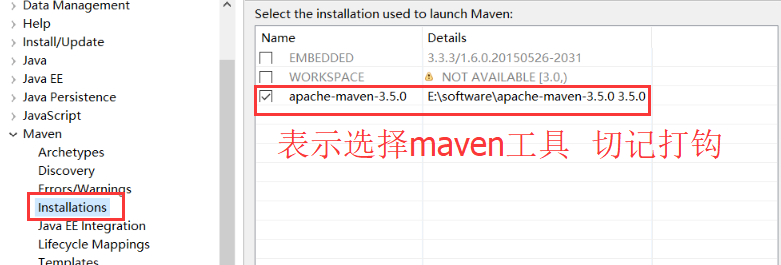
### 关闭校验



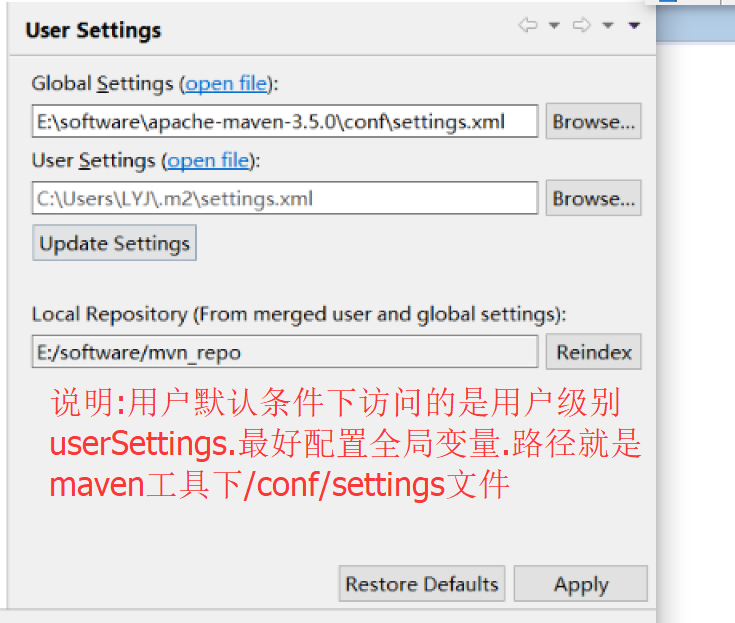
说明:关闭校验的目的是为了让编译器快速的编译.减少等待的时间.

### 添加Maven

1.添加maven工具



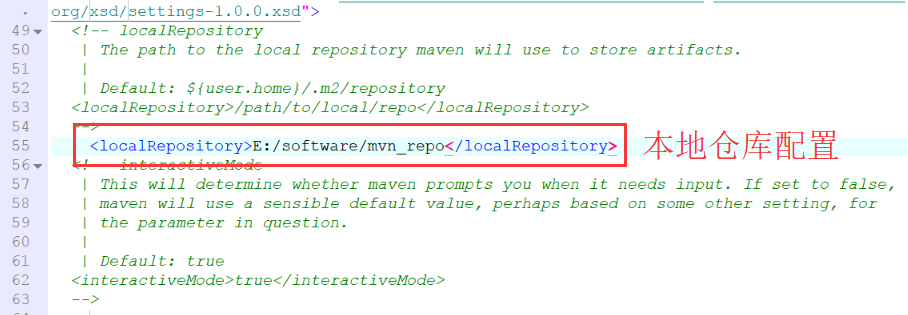
2.配置Settings文件



### 修改本地仓库路径

1.修改本地仓库路径

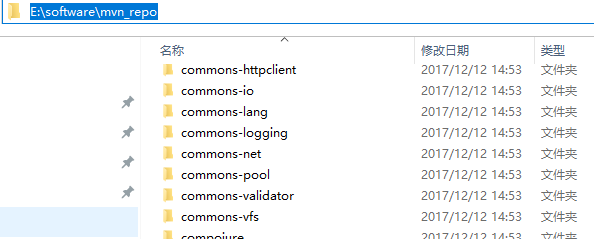
默认条件下maven访问的是m2下的repository.如果本地仓库切换,需要进行人为的修改



2.导入本地仓库

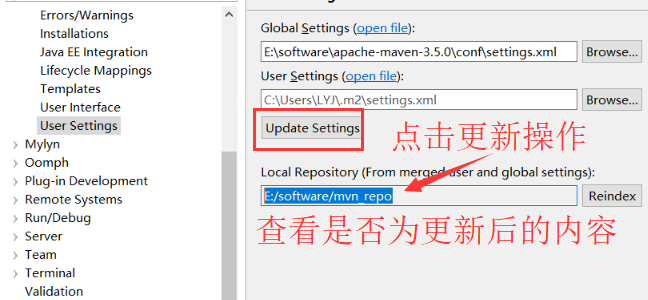
将课前资料中的mvn\_repo导入本地磁盘路径下.

注意文件夹位置.是否在根目录



3.更新Maven

打开Maven的本地仓库路径.点击更新操作.检测配置中是否更换为我们京淘的本地仓库

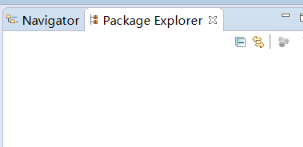


## 骨架构建jt-parent

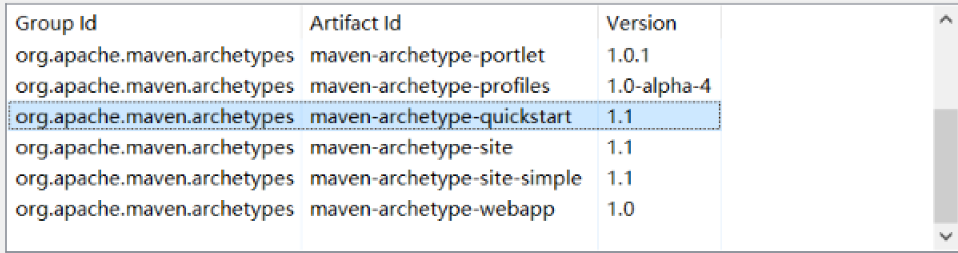
说明:jt-parent的项目类型为POM

### 视图选择

1. Navigator视图一般以文件夹的形式进行展现,比较直观.方便代码拷贝
2. Package视图 进行代码编辑时使用的视图



### 选择Maven骨架



GroupId:组Id

一般描述的是公司的具体的网站和分组Id

ArtifaceId

表示项目的名称.不能重复

Version:版本号

默认都是0.0.1

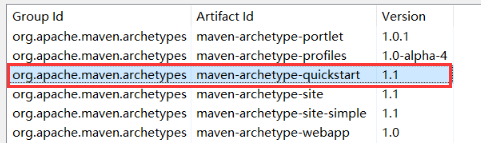
上述三项内容共同组成的的项目的坐标

坐标的作用:

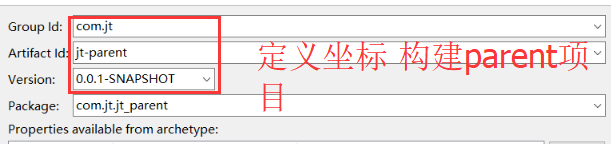
1. 方便定位jar包的具体路径
2. 方便小组内部实现jar包共享

3.方便项目编译后会根据坐标之间生成在本地仓库中,方便别人调用

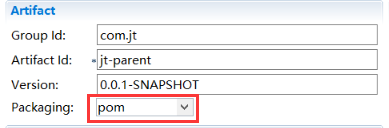
1.创建java项目



2.编辑坐标



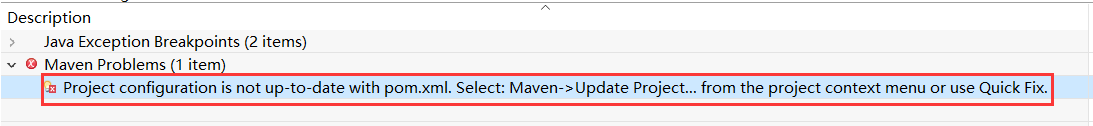
3.生成项目修改为pom



将原来的jar改为POM.表示我是一个父级工程可以被继承.

4.更新Maven项目

如果项目为POM文件之后文件会出现报错信息



将maven进行update即可.

步骤:在项目中右键🡪Maven-🡪update Project即可

5.导入jar包文件

将京淘项目课前资料中Day01的jt-parent中的pom.xml拷贝

<properties>和<dependencies>中的内容

导入完成之后点击保存



### 创建jt-common

说明:工具类项目的名称jt-common 是一个jar包项目

1. 构建项目

说明:选择quickStart骨架构建项目

编辑坐标创建项目

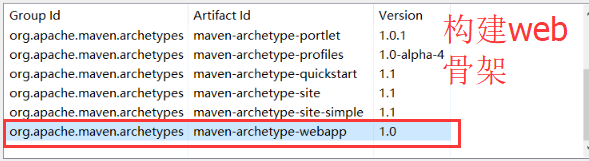
## 

# 构建jt-manage

## 构建步骤

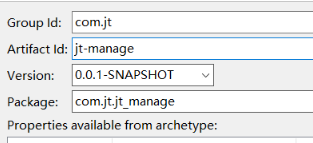
### 选择骨架

说明:由于jt-manage是一个web项目.所以通过web骨架webapp新建项目



### 修改坐标

说明:编辑坐标 项目名称不能重复



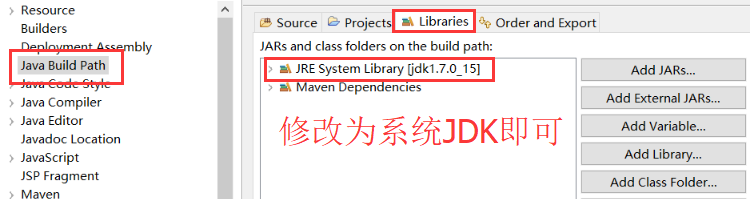
### 修改JDK

说明:

由于eclipse于Maven兼容性的问题,使创建web项目时,没有生成main/java,main/text文件夹

解决办法:

需要手动的修改JDK即可

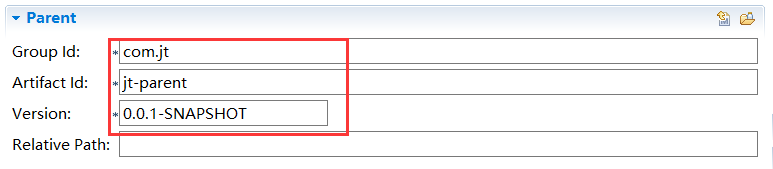


## 实现依赖和继承

### 继承父级项目

由于jt-manage项目是业务模块.所以需要继承jt-parent项目.具体操作如下.

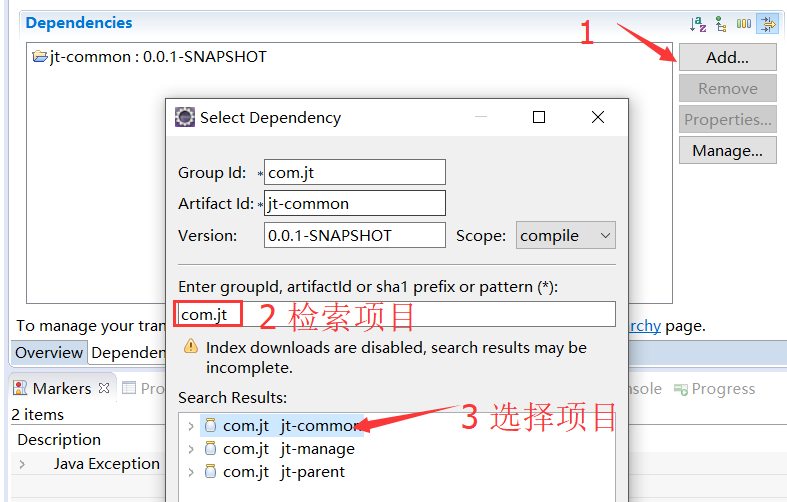
在pom.xml文件中添加parent选项



### 依赖工具类项目

说明:

Jt-manage项目中需要添加工具类文件后方便方法的执行.所以要依赖项目



引入完成之后 记得点击保存

# Tomcat插件应用

使用tomcat实例的缺点:

1. 随着项目数量的增多,tomcat启动越来越慢
2. Tomcat在不同的编译器中配置方式不同.
3. 有些时候需要手动的改变tomcat发布路径

## Tomcat插件引入

### 引入插件

将插件文件加入到pom.xml中,后删除多余的bulid标签,不然会报错

<build>

<!--tomcat插件 -->

<plugins>

<plugin>

<!--tomcat插件的本地仓库路径 -->

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

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

<version>2.2</version>

<configuration>

<!--tomcat发布端口 -->

<port>8091</port>

<!--/表示根目录 /jt -->

<path>/</path>

</configuration>

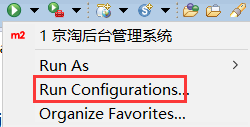
</plugin>

</plugins>

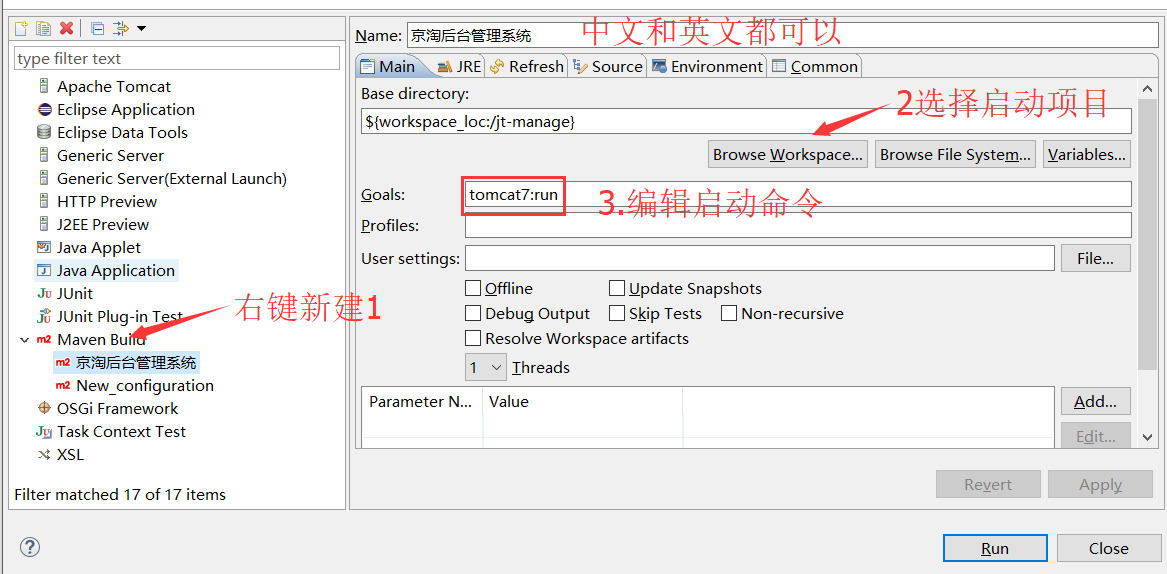
</build>

### 创建启动项

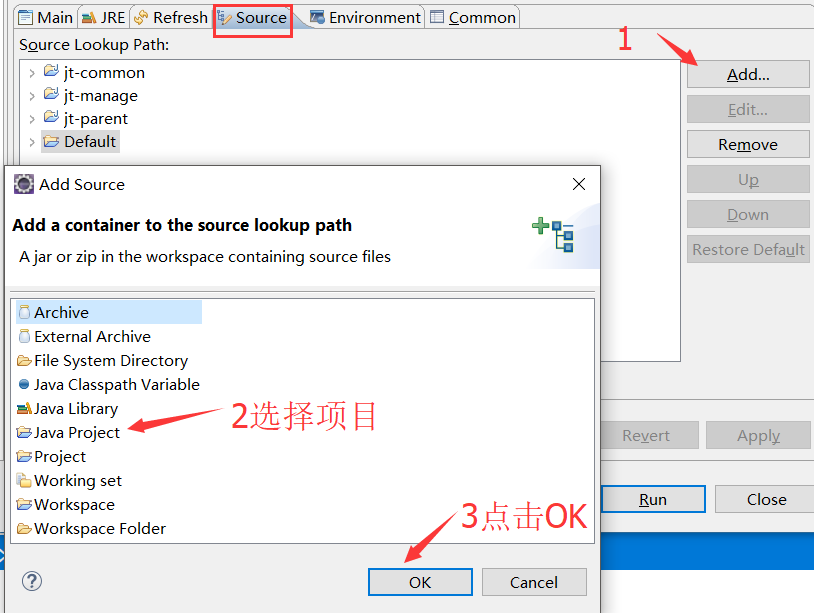
1.点击runAs下的run Configurations 进行配置



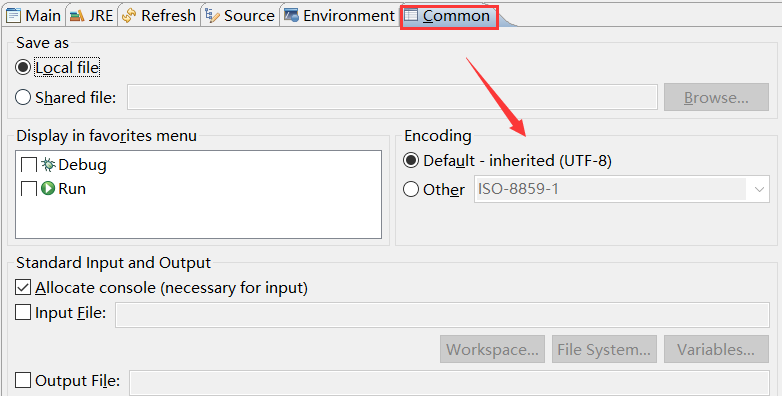
1. 配置启



1. 引入源码

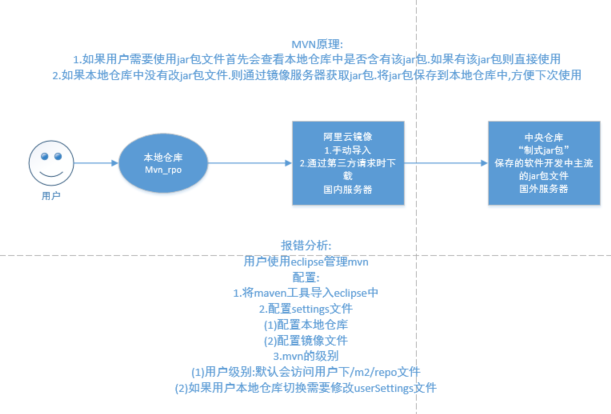


1. 检测字符集



# 补充知识

## Maven的工作原理



## Maven修改JDK

说明:通过下面的配置可以修改maven生成项目时所使用的jdk

<profile>

<id>jdk17</id>

<activation>

<activeByDefault>true</activeByDefault>

<jdk>1.7</jdk>

</activation>

<properties>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

<maven.compiler.compilerVersion>1.7</maven.compiler.compilerVersion>

</properties>

</profile>

