# 作业

## 作业实现

### 编辑页面JS

<%@ page language=*"java"* contentType=*"text/html; charset=utf-8"* pageEncoding=*"utf-8"*%>

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>您好Springboot</title>

<!--ajax使用步骤

1.jQuery是js的函数类库(API),必须导入js文件

2.让页面加载完成之后执行

3.发起ajax $.ajax $.get $.post $.getJSON $.load

4.获取服务端数据,利用js实现页面列表展现

-->

<script type=*"text/javascript"* src=*"/js/jquery-3.4.1.min.js"*></script>

<script type=*"text/javascript"*>

//2.让页面加载完成之后执行js

$(**function**(){

//3.发起ajax请求 key:value

$.ajax({

type: "get",

url: "http://localhost:8090/userList-ajax",

//data: ajax向服务器提交的数据 {key:value},

dataType: "json", //标识返回值结果类型

success: **function**(result){ //result是服务器返回的数据

**var** trs = getUserTrs(result);

$("#userTable1").append(trs);

}

});

/\*2.使用$.getJSON获取数据\*/

$.getJSON("http://localhost:8090/userList-ajax",

**function**(result){

**var** trs = getUserTrs(result);

$("#userTable2").append(trs);

});

//封装js函数

**function** getUserTrs(result){

**var** tr = "";

$.each(result,**function**(index,user){

**var** id = user.id;

**var** name = user.name;

**var** age = user.age;

**var** sex = user.sex;

tr += "<tr align='center'><td>"+id+"</td><td>"+name+"</td><td>"+age+"</td><td>"+sex+"</td></tr>"

});

**return** tr;

}

});

</script>

</head>

<body>

<table id=*"userTable1"* border=*"1px"* width=*"65%"* align=*"center"*>

<tr>

<td colspan=*"6"* align=*"center"*><h3>学生信息</h3></td>

</tr>

<tr>

<th>编号</th>

<th>姓名</th>

<th>年龄</th>

<th>性别</th>

</tr>

</table>

</html>

### 编辑业务层Controller

在userController中编辑方法.实现页面跳转.以及userListJSON数据返回.

/\*\*

\* 1.用户输入localhost:8090/userList时跳转页面

\*/

@RequestMapping("/userList")

**public** String toJSP() {

//调用视图解析器转向页面

**return** "userList-ajax";

}

/\*\*

\* 2.获取数据之后返回给页面 userListJSON

\*/

@RequestMapping("/userList-ajax")

@ResponseBody

**public** List<User> findList\_ajax(){

List<User> userList =

userService.findAll();

**return** userList;

}

# 电商行业特点

## 高并发

### 概念

说明:用户单位时间内访问服务器的次数.如果次数很多超过服务器响应能力.

### tomcat能力认知

官方说明:单台tomcat服务器处理能力 220次/秒

实际测试:实际测试与硬件有关系 150次/秒

一台tomcat全天提供响应能力: 折中算法 600万-800万之间

### 高并发场景说明

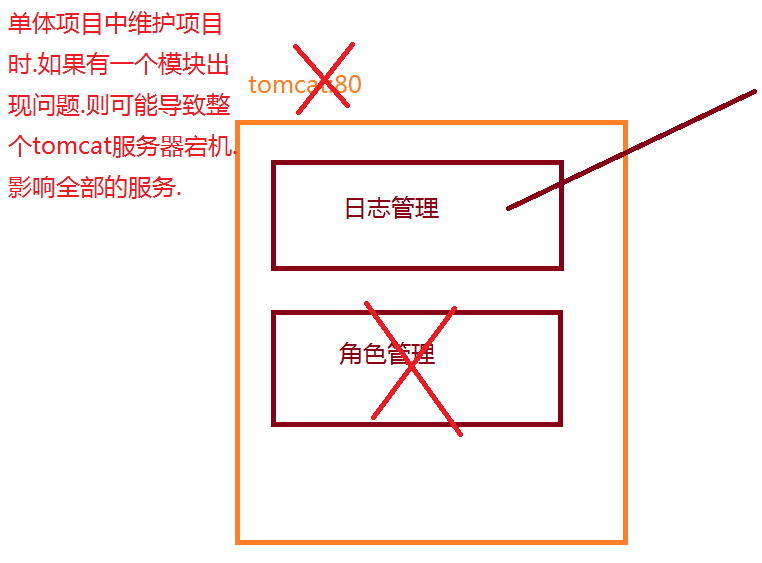
如果电商做活动时可能会有很高的短期的高的访问.例如秒杀.单台tomcat服务器能否应对.答案:肯定不行.

## 分布式

### 核心概念

将单独的项目根据特定的规则进行项目的**拆分**,减少架构的耦合性.

### 项目耦合性说明



### 垂直拆分

概念:按照功能模块(**业务**)进行拆分.将项目分别部署到不同的服务器中.



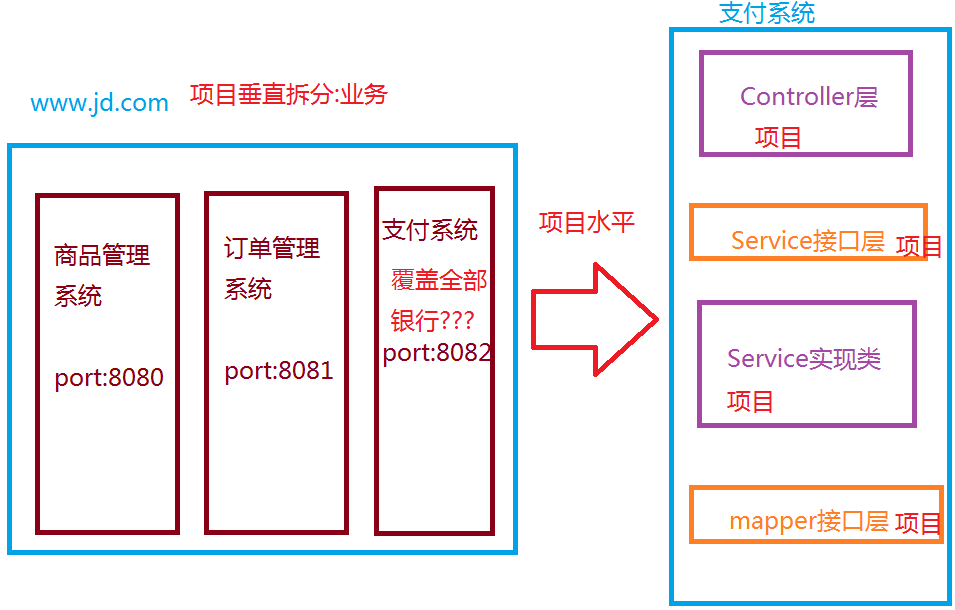
说明:垂直拆分之后,项目实现了松耦合.当一个模块出现问题.不会影响整个项目运行.

### 水平拆分

说明:

在垂直拆分的基础之上进行水平拆分,按照代码的**层级**进行拆分,将层级拆分为具体项目.

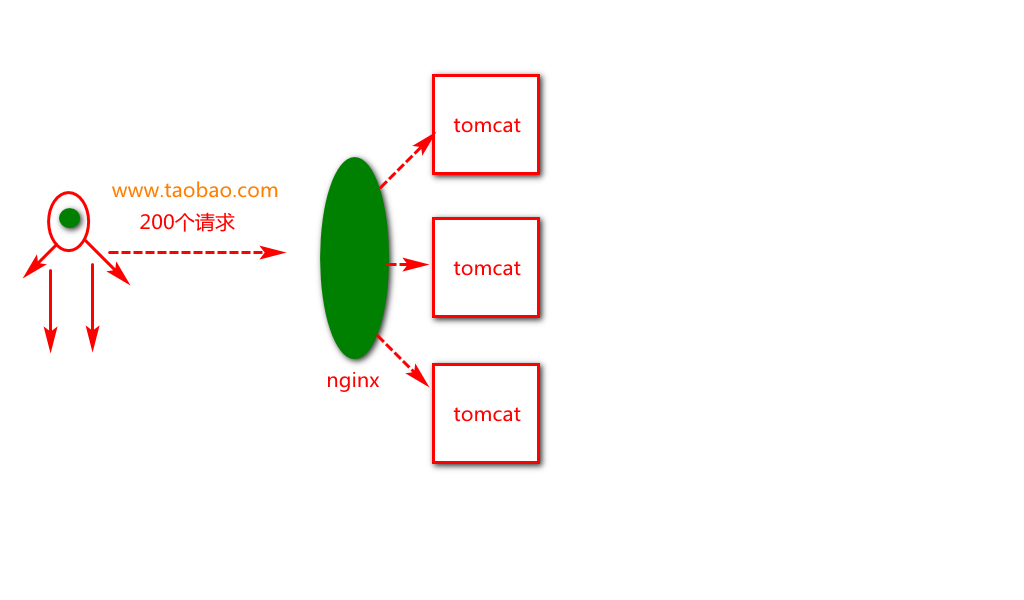
一般大项目才会进行水平拆分.因为多人维护同一个项目时容易出错.



## 集群

### 集群说明

抗击高并发最为有效的手段搭建服务器集群.

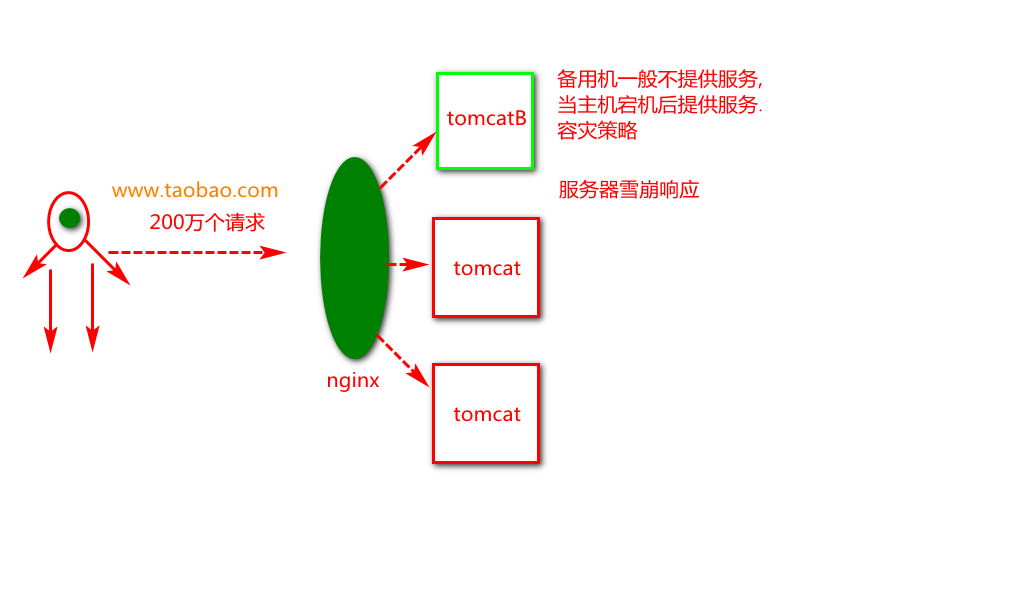


### 集群高可用

计算机系统的可用性用[1]  [平均无故障时间](https://baike.baidu.com/item/%E5%B9%B3%E5%9D%87%E6%97%A0%E6%95%85%E9%9A%9C%E6%97%B6%E9%97%B4)（**MTTF**）来度量，**即计算机系统平均能够正常运行多长时间，才发生一次故障**。系统的可用性越高，平均无故障时间越长。可维护性用平均维修时间（MTTR）来度量，即系统发生故障后维修和重新恢复正常运行平均花费的时间。系统的可维护性越好，平均维修时间越短。计算机系统的可用性定义为：MTTF/(MTTF+MTTR) \* 100%。由此可见，计算机系统的可用性定义为系统保持正常运行时间的百分比。

“高可用性”（High Availability）

总结:**高可用**表示当服务器发生故障.后台服务器能够**自动的实现故障的迁移.**



## 海量数据处理(了解)

### 概念说明

随着公司业务积累,数据量成倍增长.如果没有有效的管理方案.会导致用户长时间检索.用户体验差.

行业:大数据.

核心思想:拆大化小,分别执行.最后合并

## 网络安全(了解)

公司中一般都有网管部门,保证公司网络安全.不被恶意攻击.

1. 阿里巴巴 支付宝 每天恶意攻击13亿次 神盾局(全世界招聘黑客)

达摩院 技术研发部

法律规定: 3-7 抢劫金融机构 15年以上

1. 360安全卫士.

## 分布式项目构建

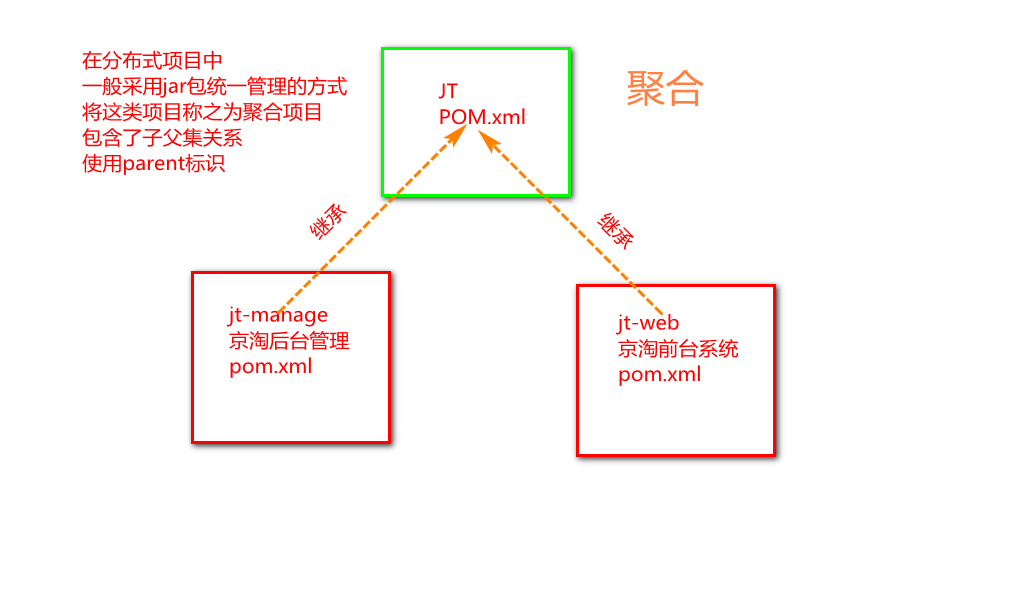
### 分布式说明

经过垂直拆分/水平拆分.将单独的项目进行分布式部署(多个项目)

### 分布式项目jar包管理

问题:由于是分布式部署.jar包通过pom.xml文件进行管理时.如果修改一个jar包.则应该修改全部项目的pom.xml.这样方式耦合性高.

优化:采用聚合项目构建方式,需要添加parent标签

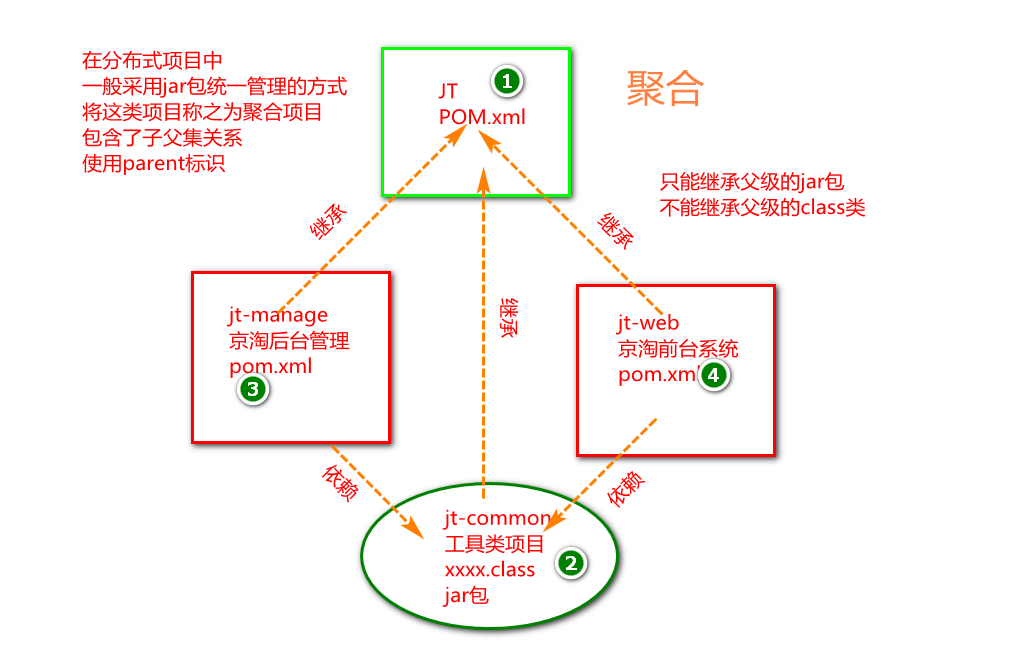


### 分布式项目工具类管理

问题说明:

由于分布式项目.工具类如果每个项目中都写一份.耦合性高.

解决方案:



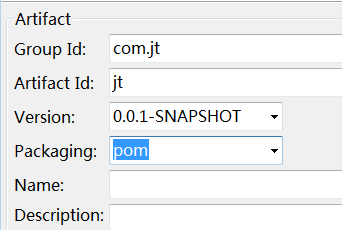
## 京淘项目构建

### 构建说明

1. 聚合项目jt 打包为pom 表示可以被别人继承.
2. jt-comon 打包为jar 可以被其他项目依赖调用
3. jt-manage 打包为war 继承jt 依赖common

### 构建jt项目

#### 选择类型



#### 添加pom.xml文件

<!--parent表示定义公共的jar包 -->

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!--springBoot动态的引入springMVC全部的配置 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- 引入aop支持 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-aop</artifactId>

</dependency>

<!--引入测试类 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!--添加属性注入依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

<!--支持热部署 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>springloaded</artifactId>

<version>1.2.8.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

</dependency>

<!--引入插件lombok 自动的set/get/构造方法插件 -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</dependency>

<!--引入数据库驱动 -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.32</version>

<scope>runtime</scope>

</dependency>

<!--引入druid数据源 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.1.12</version>

</dependency>

<!--spring整合mybatis-plus -->

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus-boot-starter</artifactId>

<version>3.0.6</version>

</dependency>

<!--spring整合redis -->

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<!--springBoot整合JSP添加依赖 -->

<!--servlet依赖 注意与eureka整合时的问题 -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

</dependency>

<!--jstl依赖 -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<!--使jsp页面生效 -->

<dependency>

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

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

<!--添加httpClient jar包 -->

<dependency>

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

<artifactId>httpclient</artifactId>

</dependency>

<!--引入dubbo配置 -->

<!--<dependency> <groupId>com.alibaba.boot</groupId> <artifactId>dubbo-spring-boot-starter</artifactId>

<version>0.2.0</version> </dependency> -->

<!--添加Quartz的支持 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-quartz</artifactId>

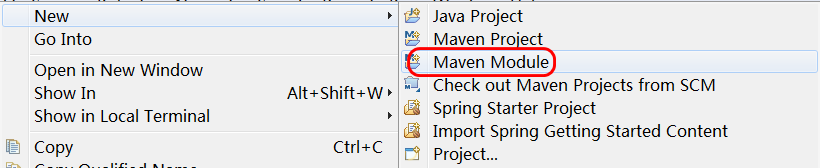
</dependency>

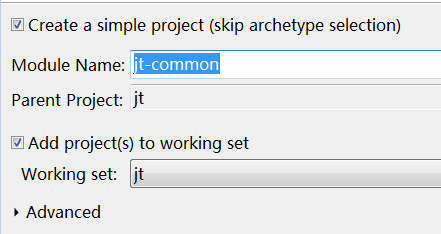
</dependencies>

## 创建jt-common

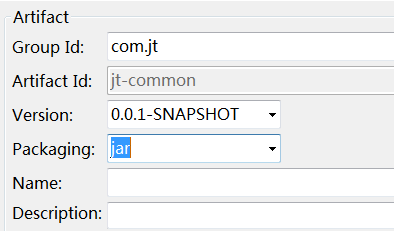
### 选择项目创建方式

选择该模式表示.创建的项目是聚合项目的一部分.





图中可以看出.已经设定子父集关系.

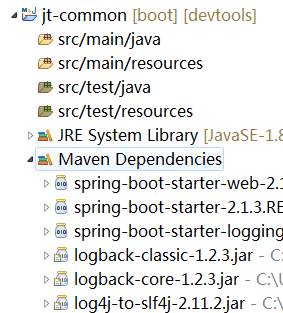


选择打包方式jar包

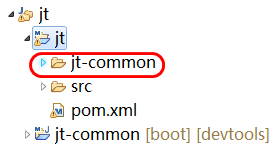
### 项目层级关系

#### jt-common介绍

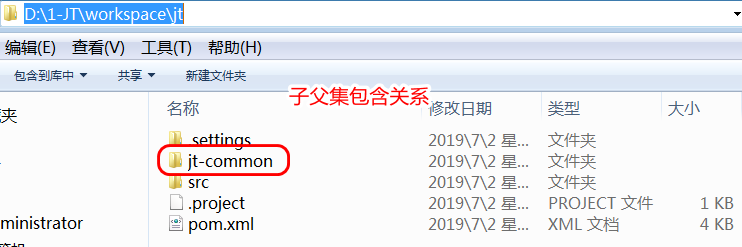
说明:jt-common中拥有了jt中的全部jar包



#### 检查jt是否包含common

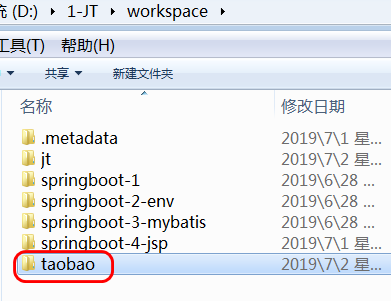


#### 检查工作空间信息

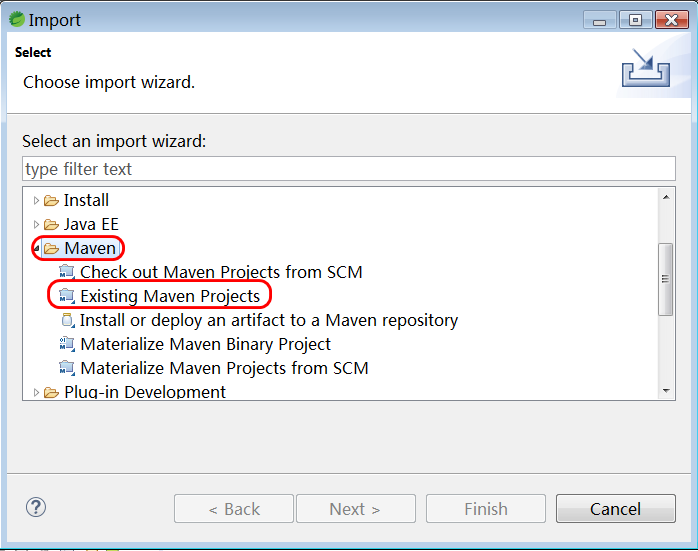


### 聚合项目导入方式

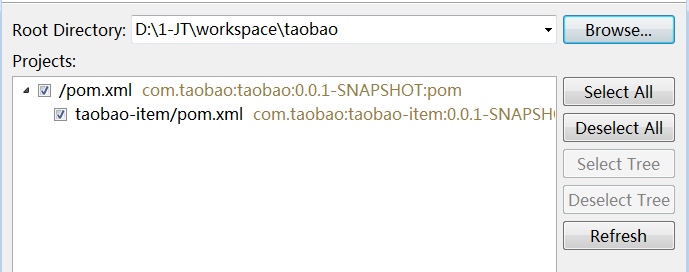
1. 将项目拷贝到本地的工作空间中.



1. 通过maven工具直接导入项目

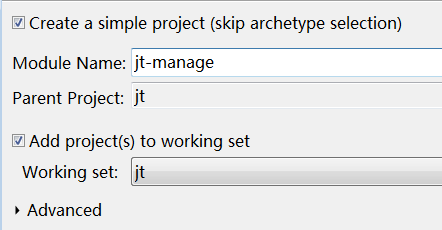


1. 选择项目导入

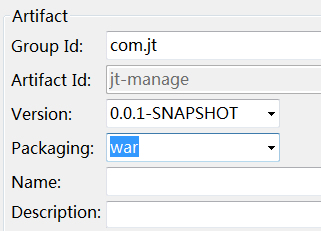


## 创建jt-manage

### 项目创建

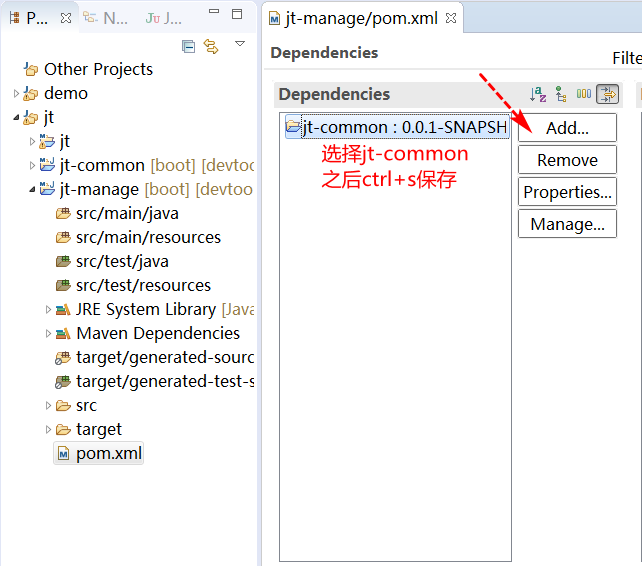


2.选择war包



### 添加依赖项

在jt-manage的pom.xml文件中 添加依赖项



易错项:如果项目中报错.不能依赖common则需要将jt-common单独打包 install

### 编辑jt-manage

#### 编辑pom.xml

说明:在jt-manage中添加插件.

<build>

<plugins>

<!--跳过测试类打包 -->

<plugin>

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

<artifactId>maven-surefire-plugin</artifactId>

<configuration>

<skip>true</skip>

</configuration>

</plugin>

<!--表示maven打包的插件 -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

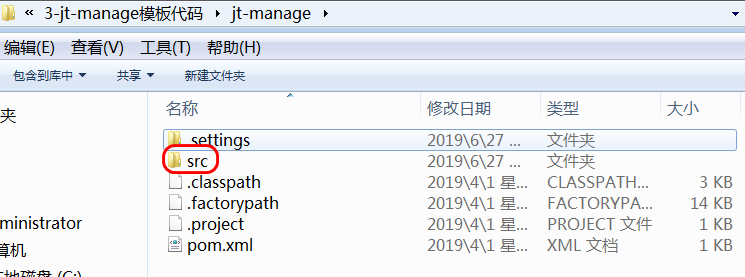
</plugin>

</plugins>

</build>

#### 导入静态资源文件

说明:将课前资料中的src文件导入项目



# 京淘项目后台管理系统

## PowerDesinger

### 介绍

**PowerDesigner**最初由Xiao-Yun Wang（[王晓昀](https://baike.baidu.com/item/%E7%8E%8B%E6%99%93%E6%98%80/9236799)）在SDP Technologies公司开发完成。PowerDesigner是[Sybase](https://baike.baidu.com/item/Sybase)的企业建模和设计解决方案，采用模型驱动方法，将业务与IT结合起来，可帮助部署有效的企业体系架构，并为研发生命周期管理提供强大的分析与设计技术。PowerDesigner独具匠心地将多种标准数据建模技术（UML、业务流程建模以及市场领先的[数据建模](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BB%BA%E6%A8%A1/720111)）集成一体，并与 .NET、WorkSpace、PowerBuilder、Java™、Eclipse 等主流开发平台集成起来，从而为传统的软件开发周期管理提供业务分析和规范的数据库设计解决方案。此外，它支持60多种关系数据库管理系统（RDBMS）/版本。PowerDesigner运行在Microsoft Windows平台上，并提供了Eclipse插件.

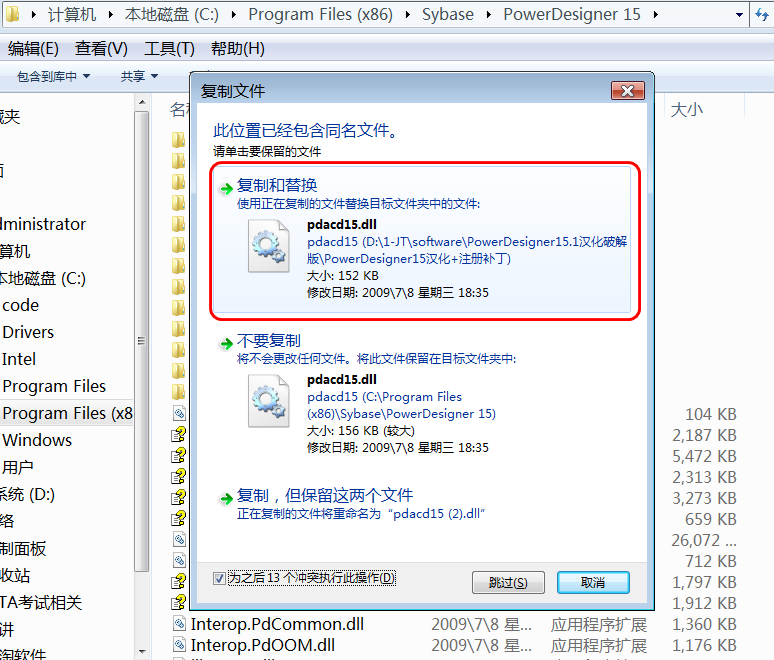
概括:提供可视化的图形界面操作数据库.并且非常直观的能够表达数据库的关联关系.

针对不同的数据库会自动的生成sql语句.

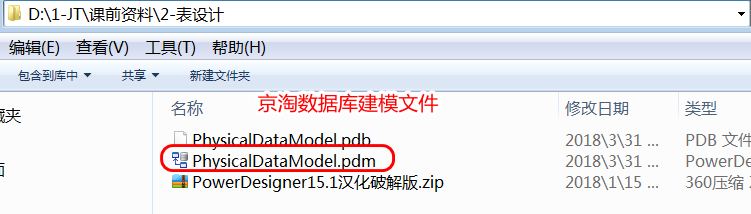
### PD安装使用

1. 汉化

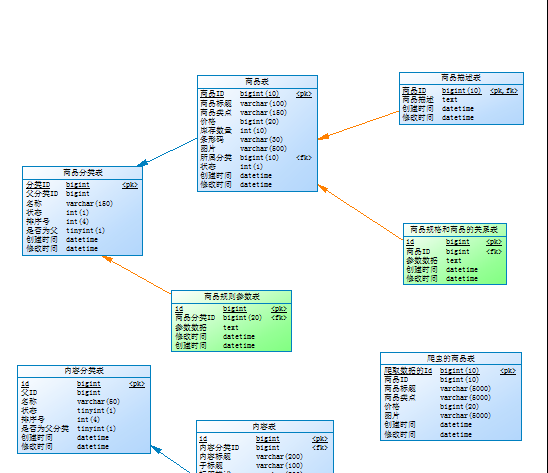
将课前资料的汉化补丁复制,覆盖PD安装根目录的文件即可完成破解.



1. 将课前资料文件中的PDM文件使用PD打开即可



1. 破解成功如图所示



## 商品查询实现(一)

### 定义BasePOJO

//定义公共的POJO对象

@Data

@Accessors(chain = **true**)

**public** **class** BasePojo **implements** Serializable{

**private** Date created;

**private** Date updated;

}

### 定义商品POJO对象

@TableName("tb\_item")

@Data

@Accessors(chain = **true**)

**public** **class** Item **extends** BasePojo{

@TableId(type = IdType.***AUTO***)

**private** Long id; //定义主键

**private** String title; //标题

**private** String sellPoint; //卖点信息

**private** Long price; //商品价格 1.dubbo有精度问题 0.88888..+0.1111112=0.9999.... //2.速度 int > long > dubbo 将商品扩大100倍保存

**private** Integer num; //商品数量

**private** String barcode; //二维码

**private** String image; //商品品图片信息

**private** Long cid; //商品分类信息

**private** Integer status; //商品状态 1正常，2下架，3删除'

}

### 作业

预习EasyUI 12个案例.