

# 第一天：项目背景+搭建框架

## 一. 学习项目的心态：

1. 用什么方法来学习这些技术的?  
    是什么?  
    它的实现原理?  
    用在哪些地方?  
    怎么用?
2. 如何学习项目课程?
  1. 学什么?  
        主要是学习项目的业务！什么是业务？就是做事情的一些步骤。**能够跟面试官熟练的表达出你所熟悉的业务。**
  2. 表达能力？很重要
  3. 能将业务使用代码编写出来
  4. 学到新知识点，用多少学多少，只讲用少讲原理
  5. 提高自学能力，提高调试能力，提高表达沟通能力

## 二.项目背景

商务综合管理平台是国际物流行业一家专门从事进出口**玻璃器皿**贸易的公司。业务遍及欧美。随着公司不断发展壮大，旧的信息系统已无法满足公司的快速发展需求，妨碍公司成长，在此背景下，公司领导决定研发《商务综合管理平台》。

《商务综合管理平台》分三期完成。一期完成仓储管理（包括：采购单、仓库、货物、条形码、入库、出库、退货、盘点、库存、库存上限报警、统计查询）和展会管理（包括：展会管理、出单管理），形成货物统一数字化管理。二期完成货运全流程管理，包括购销合同、出货表统计、出口报运单、HOME 装箱单、装箱单、委托书、发票、财务统计等。三期完成决策分析（包括：成本分析图、销售情况统计、重点客户、经营情况同期比对统计、工作绩效），为公司经营决策提供数据支持。

如何从国外拿到订单？

通过一些展销会，拿到订单。到国内找生产厂家生产货物。在指定日期生产厂家要将生产的货物运到码头，同时一边到海关进行审批。

## 三.为什么选择国际物流项目？

- 1.市场
- 2.很火
- 3.业务

## 四.界面原型法

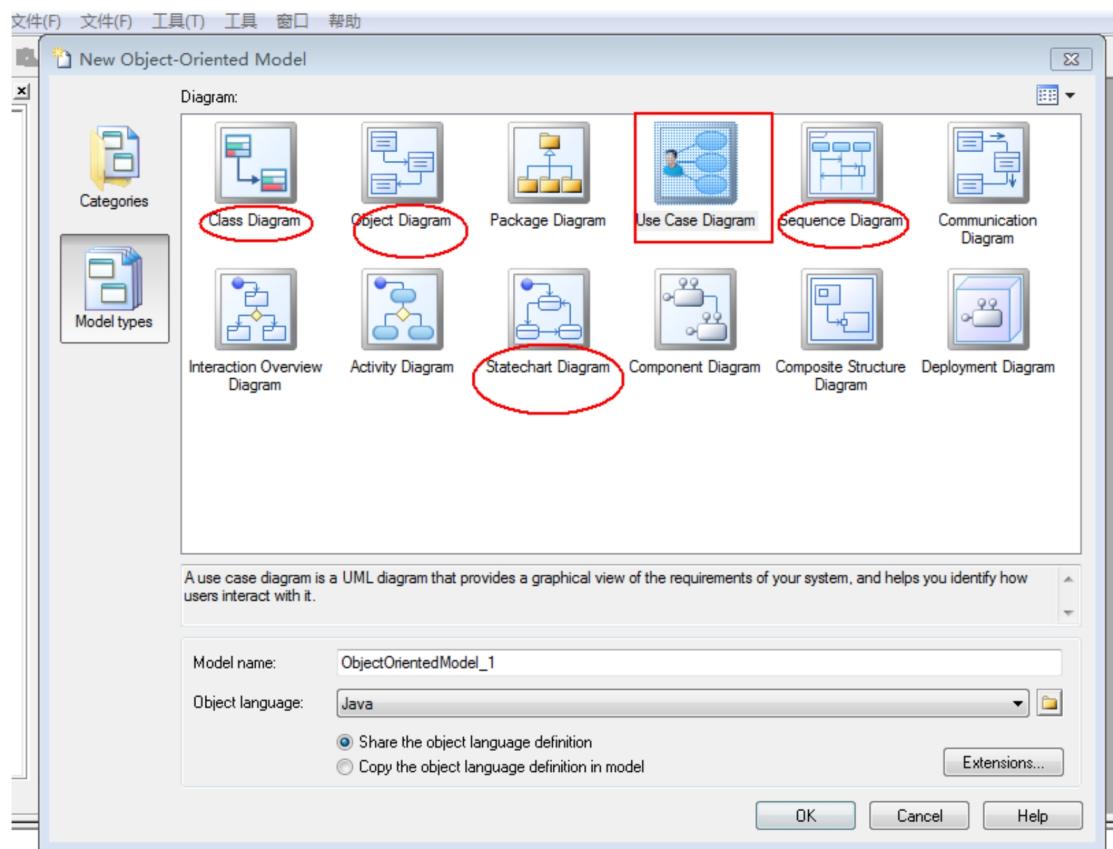
原型它的目的，它给客户提前看未来的系统长什么样子。客户就能有一个直观的印象。  
界面原型法，在实际需求调研阶段用的非常多。和用户谈需求并进行记录，跟 web 前端页面工程师沟通，让其设计出一套相关的页面原型，再拿过去与用户再次进行沟通，并修改进行记录，再回来进行页面的修改，如果确认，将来的页面就基本不再改动。

目的就是在最短的时间里，得到用户最真实的需求。

## 五.UML 的 UseCase 图

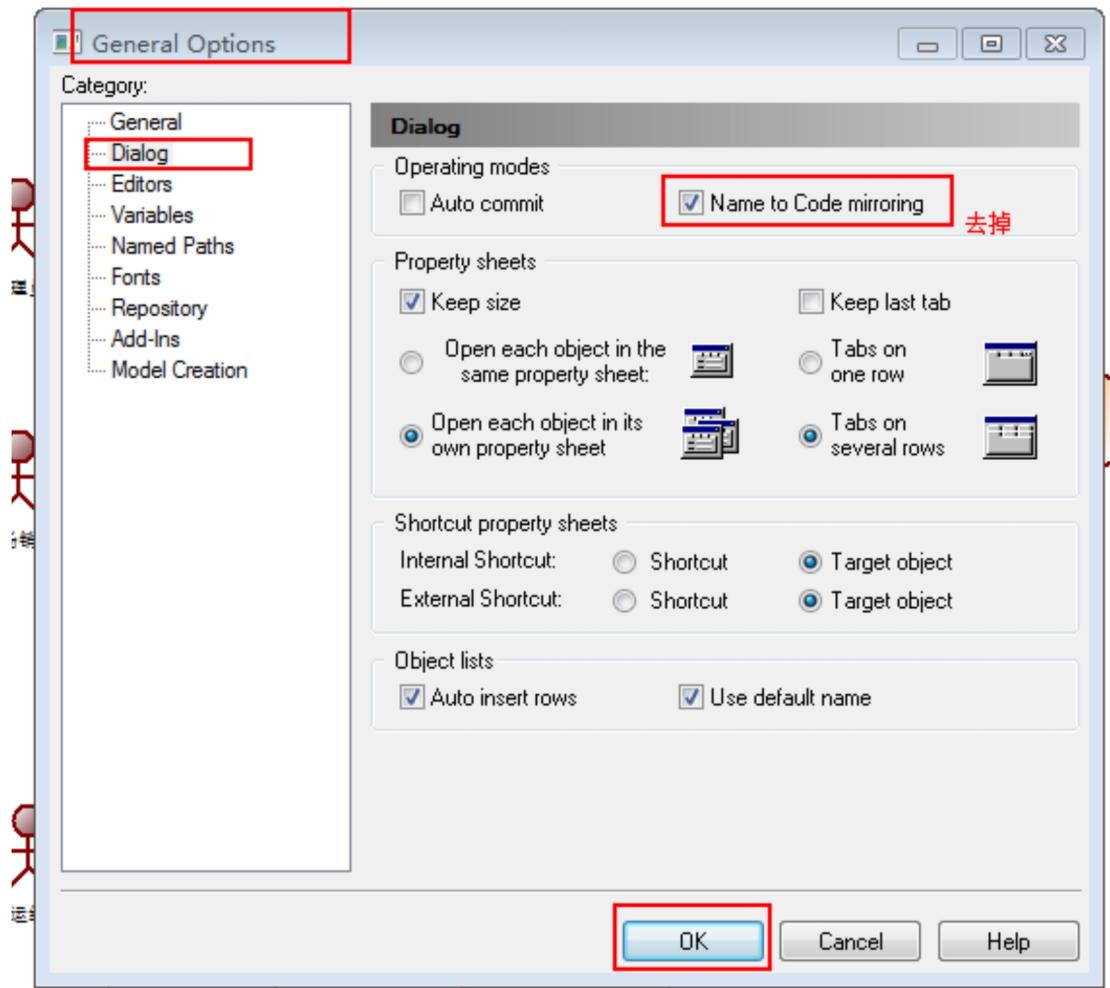
- 1.获取需求是专业的
- 2.usecase 就是用例图，它是专门用于描述需求的
- 3.uml 代表统一建模语言  
    它包含很多图： 用例图，类图，状态图，序列图.....
- 4.软件：  
    Rose, pd(powerDesigner 可以画 uml 图也可以进行数据库设计), Visio  
    OOP 面向对象编程  
    OOM 面向对象建模  
    OOD 面向对象设计  
    OOA 面向对象分析 (analysis)

工作空间—新增----oom----use case diagram

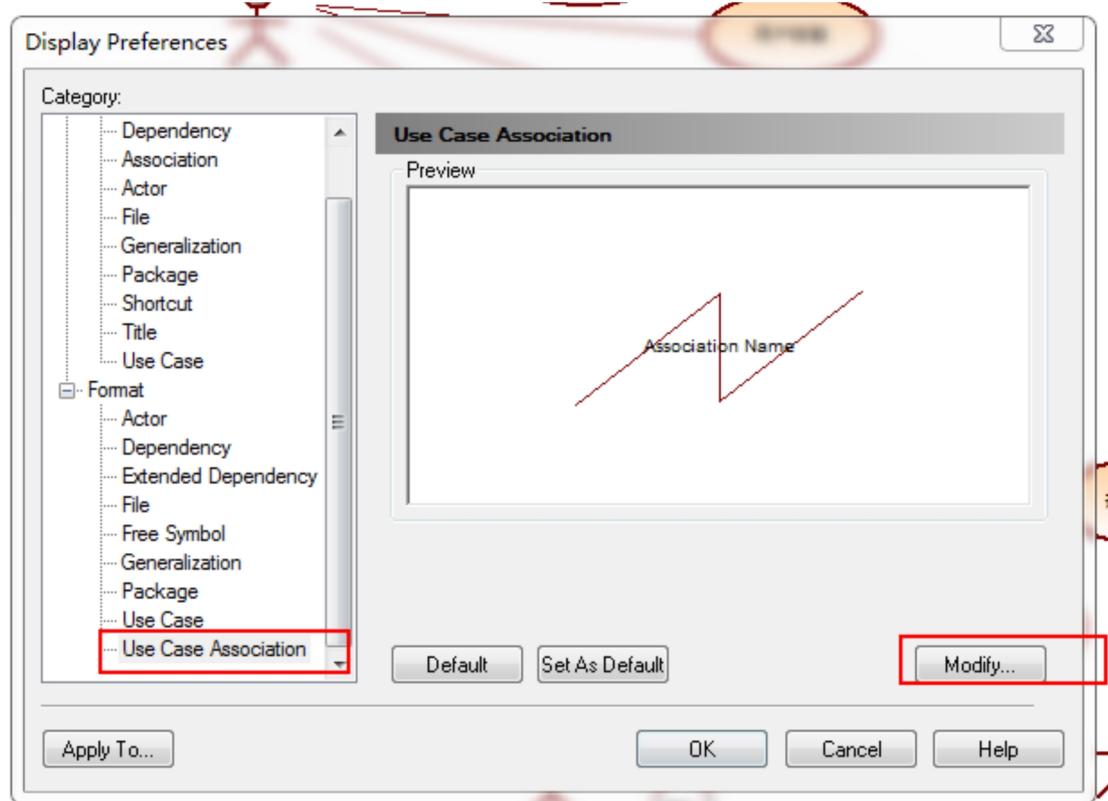


进入一个面板：

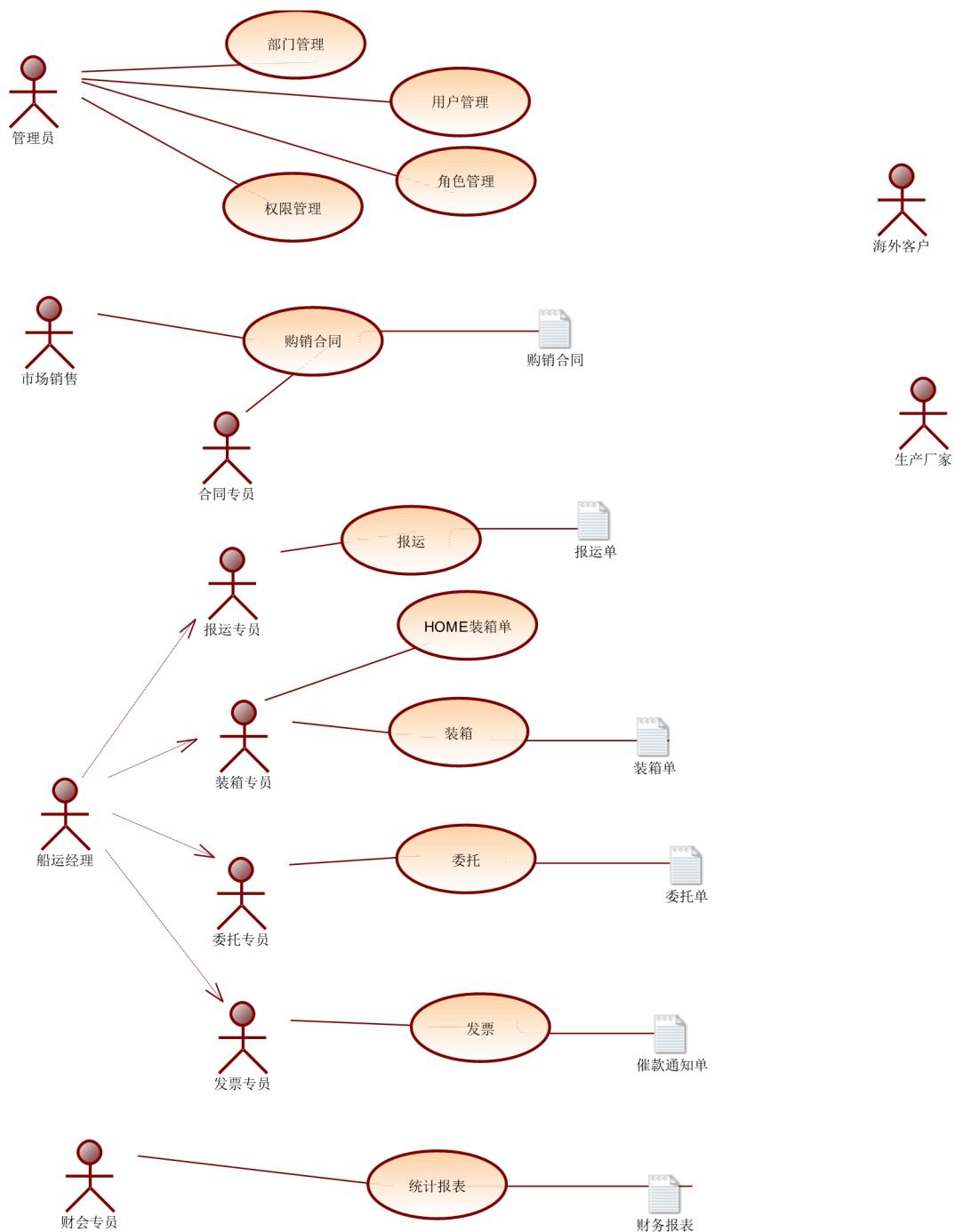




解决线条的问题：

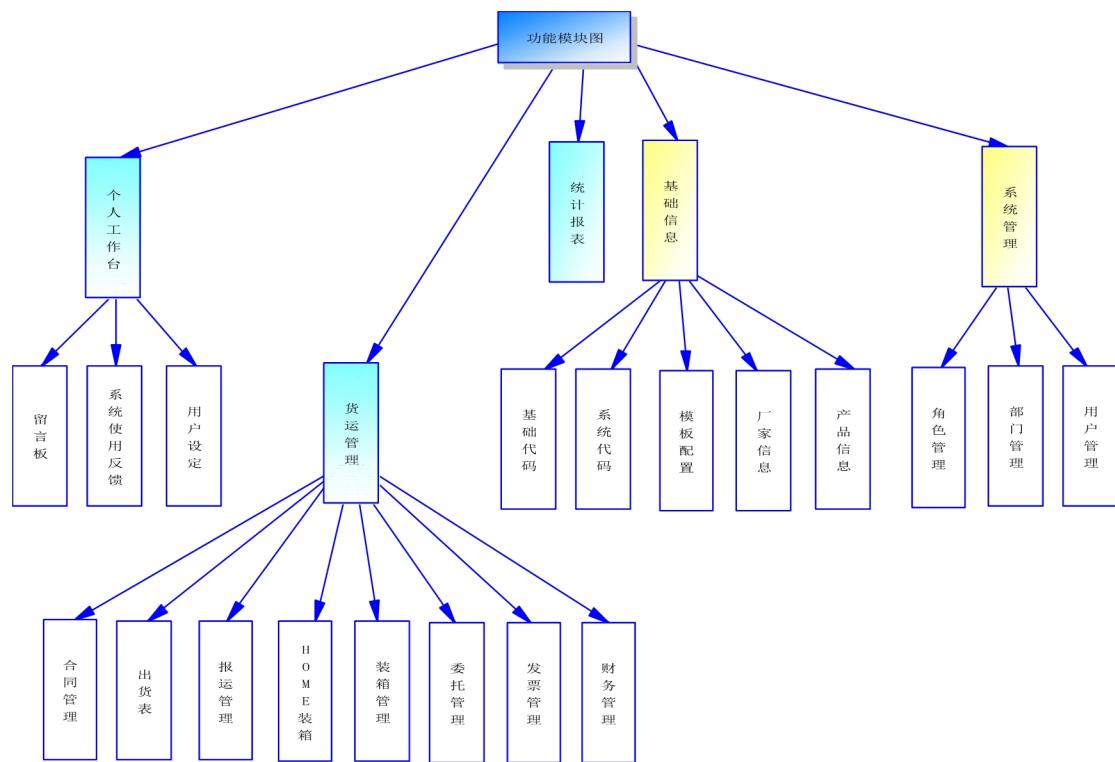


需求描述：



## 六. 系统功能模块结构图

系统功能结构图

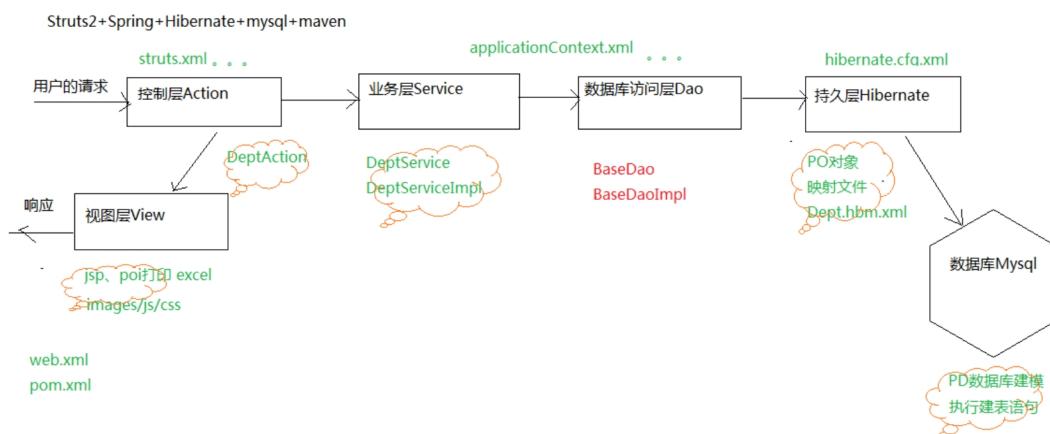


它的作用？

1. 功能模块一目了然
2. 便于分工
3. 便于进行项目报价

## 七. 系统框架

如果自己搭建框架：

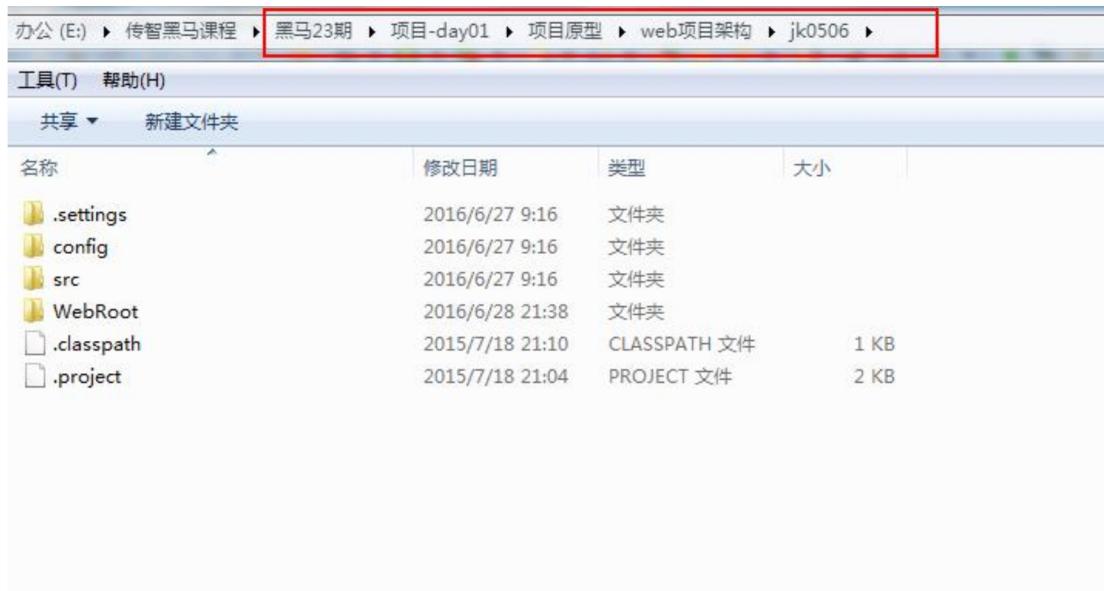


如果使用的别人的框架：

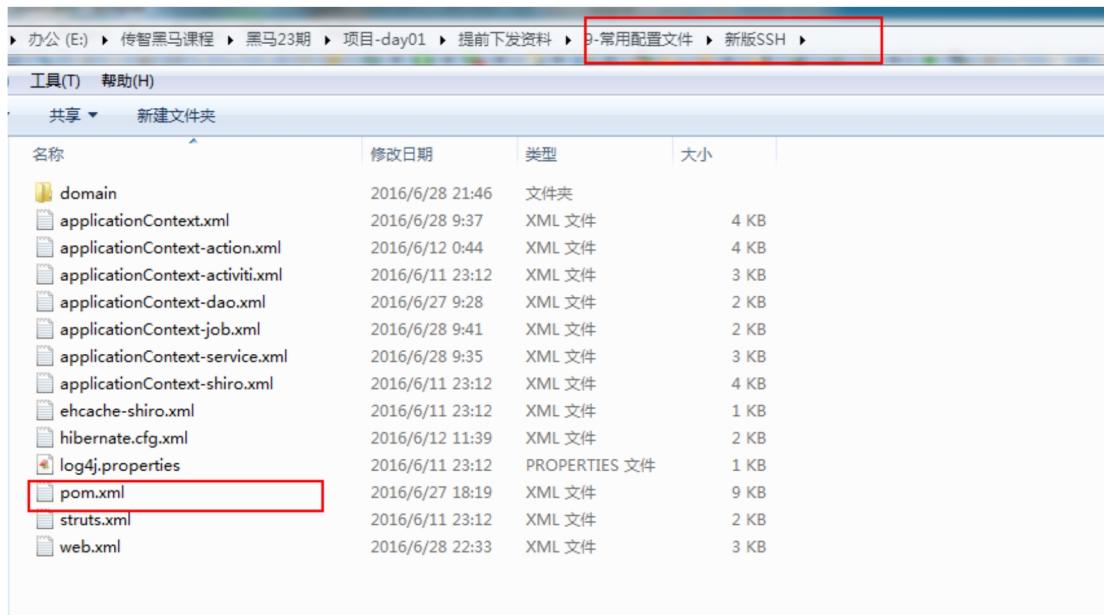
- 1.首先看 jar 包
- 2.画图
- 3.确定哪些技术需要学习。

搭建 maven 结构的过程：

1. 找到资源



2. 常用配置文件的资源：



3. 创建一个 maven project: jk2601\_parent

4. 在 jk2601\_parent 中找到 pom.xml 文件，添加相应的依赖

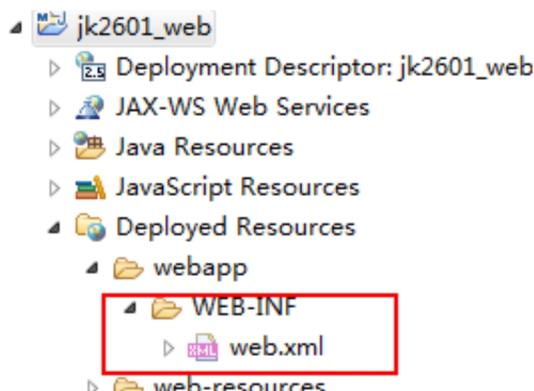


引入这个 pom.xml,如果没有本地的更新，也要先进行本地仓库的更新

5. 拆分的 maven 工程如下：

- ▷  jk2601\_dao
- ▷  jk2601\_domain
- ▷  jk2601\_exception
- ▷  jk2601\_parent
- ▷  jk2601\_service
- ▷  jk2601\_util
- ▷  jk2601\_web

6. 解决上面的 jk2601\_web 上面的小错误



7. 添加依赖

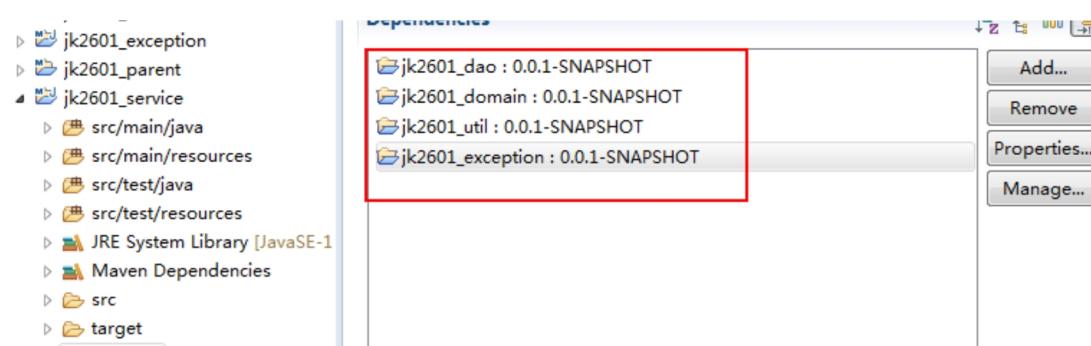
Jk2601\_domain 中添加 jk2601\_util

DAO 中依赖



The screenshot shows the 'Dependencies' view for the 'jk2601\_dao' module. On the left, the project structure is shown with 'jk2601\_dao' expanded, revealing 'src/main/java', 'src/main/resources', 'src/test/java', and 'src/test/resources'. On the right, the 'Dependencies' view lists two dependencies: 'jk2601\_domain : 0.0.1-SNAPSHOT' and 'jk2601\_util : 0.0.1-SNAPSHOT'. These dependencies are also highlighted with a red rectangular box.

Service 依赖:



The screenshot shows the 'Dependencies' view for the 'jk2601\_service' module. On the left, the project structure is shown with 'jk2601\_service' expanded, revealing 'src/main/java', 'src/main/resources', 'src/test/java', 'src/test/resources', 'JRE System Library [JavaSE-1]', 'Maven Dependencies', 'src', and 'target'. On the right, the 'Dependencies' view lists four dependencies: 'jk2601\_dao : 0.0.1-SNAPSHOT', 'jk2601\_domain : 0.0.1-SNAPSHOT', 'jk2601\_util : 0.0.1-SNAPSHOT', and 'jk2601\_exception : 0.0.1-SNAPSHOT'. These dependencies are highlighted with a red rectangular box.



8. 加入相关的 util 工具类

9. 加入 dao 相关的类



```
11 http://www.springframework.org/schema/aop/spring-aop.xsd
12 http://www.springframework.org/schema/tx
13 http://www.springframework.org/schema/tx/spring-tx.xsd
14 http://www.springframework.org/schema/context
15 http://www.springframework.org/schema/context/spring-context.xsd>
16
17<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
18     <property name="dataSource" ref="dataSource"></property>
19 </bean>
20<bean id="sqlDao" class="cn.itcast.jk.dao.common.SqlDao">
21     <property name="jdbcTemplate" ref="jdbcTemplate"></property>
22 </bean>
23
24<bean id="baseDao" class="cn.itcast.jk.dao.impl.BaseDaoImpl">
25     <property name="sessionFactory" ref="sessionFactory"></property>
26 </bean>
```

10. jk\_2601\_domain 中操作的表

Name	Type	Nullable	Default/Expr.	Storage	Comments
DEPT_ID	VARCHAR2(40)	<input type="checkbox"/>	...		
DEPT_NAME	VARCHAR2(40)	<input checked="" type="checkbox"/>	...		
PARENT_ID	VARCHAR2(40)	<input checked="" type="checkbox"/>	...		
STATE	NUMBER(11)	<input checked="" type="checkbox"/>	...		1代表启用，0代表停用，默认为1
					...

编写 domain 中的 Dept 类及它的映射文件

11. jk\_2601\_service 中

12. jk2601\_web

13. 测试并运行

## 八.面试

1.请写出 hibernate 中主键生成策略常用 6 种方式

native,uuid,increment,identity,sequence,assigned

2.Byte/Short/int/char/varchar 运行速度及区别

运行速度: Byte/Short/int/char/varchar 由快到慢

Char 是一个固定长度, 这样可能导致空间浪费, 但反应快速比 varchar 快。

Varchar 可以做到空间最省, 但会多长度比较的过程, 所以很浪费时间。

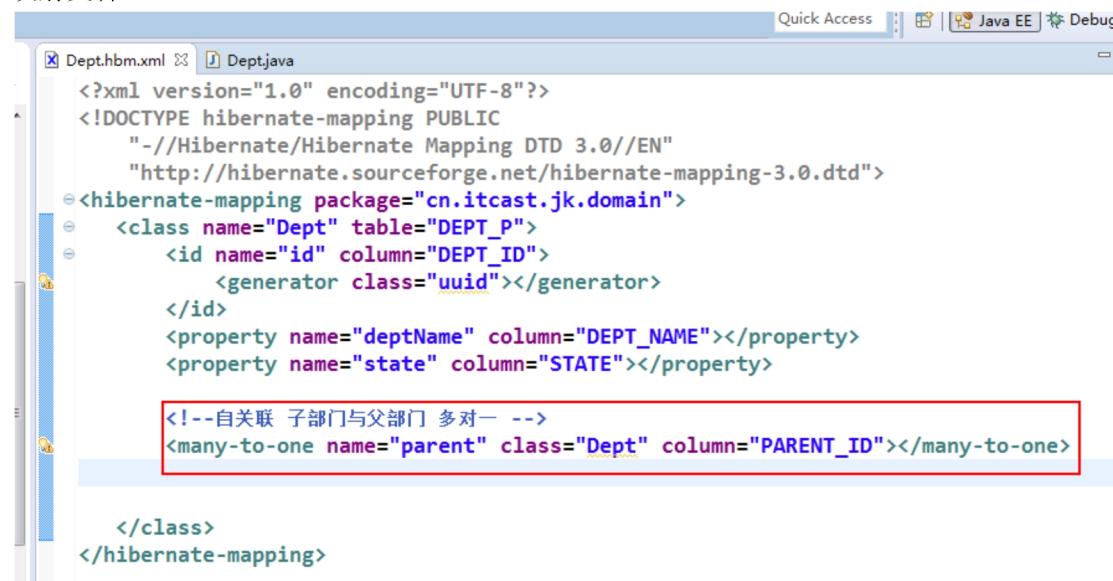
### 3.面试

请参考数据库表结构写出 PO 类及映射文件

PO类:

```
public class Dept {  
    private String id;  
    private String deptName; //部门名称  
    private Dept parent; //父部门 自关联（自己关联自身） 子部门与父部门 多对一  
    private Integer state; //状态 1启用 0停用 默认为1  
    ...  
}
```

映射文件



```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC  
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
<hibernate-mapping package="cn.itcast.jk.domain">  
  <class name="Dept" table="DEPT_P">  
    <id name="id" column="DEPT_ID">  
      <generator class="uuid"></generator>  
    </id>  
    <property name="deptName" column="DEPT_NAME"></property>  
    <property name="state" column="STATE"></property>  
  
    <!--自关联 子部门与父部门 多对一 -->  
    <many-to-one name="parent" class="Dept" column="PARENT_ID"></many-to-one>  
  
  </class>  
</hibernate-mapping>
```

面试：PO类的定义规范

- \* PO的规范？(Persist Object)
- \* 1.是一个公有类
- \* 2.提供无参公有构造方法
- \* 3.属性是私有的
- \* 4.为私有属性提供公有的getter/setter
- \* 5.不能使用final修饰
- \* 6.可以实现java.io.Serializable接口
- \* 7.如果是基本类型，请使用它的包装类

hibernate 配置文件

Hibernate.cfg.xml 文件编写

```
<!--加载映射文件-->  
<mapping resource="cn/itcast/jk/domain/Dept.hbm.xml"></mapping>
```