智慧开源基础开发引擎(ZRpower)

Version 7.0

(SpringBoot+Mybatis+Thymeleaf+Vue+Element+Shiro 版本)

(环境安装及项目结构技术手册)

文件名称	智慧开源基础开发引擎(环境安装及项目结构技术手册)V7.0	文件版本	V1.2
文件编号	ZR-JSWD-VS01	受控状态	〔√〕受控 〔 〕不受控
编制部门	技术研发部	文件页数	
编制人	付天龙、梁文楷	编制日期	2021-09-08
批准人	徐仲学	批准日期	2021-09-08

技术开发文档

网址：http://www.zrpower.cn

目　录

第一章　版权说明

本项目为免费和开源项目,本文档为付费文档，版权归贵州智慧开源软件有限公司所有，并保留一切权利，本文档及其描述的内容受有关法律的版权保护，对本文档和开源代码以任何形式的非法复制、泄露或散布到网络提供下载，都将导致相应的法律责任。

文档更新

本文档由贵州智慧开源软件有限公司于 2021 年 9 月最后修订。

第二章　前端 VUE 环境搭建

Vue.js（读音 /vju:/，类似于 view）是一个构建数据驱动的 Web 界面的渐进式框架。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

2.1　前端 Vue 各软件版本

Nodejs 版本：12.22.6

Npm 版本：6.11.3

Vue 版本：2.5.21

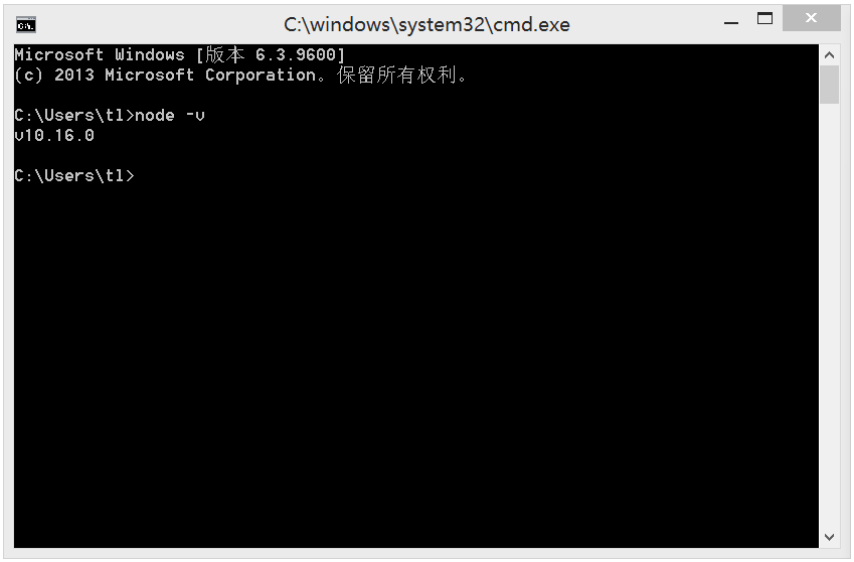
Element-ui 版本：2.15.5

2.2　安装 Node.js

前端开发框架和环境都是需要 Node.js，先安装 node.js 开发环境，vue 的运行是要依赖于 node 的 npm 的管理工具来实现，下载 https://nodejs.org/en/，安装完成之后，打开 cmd 开始输入命令。（我用的是 win10 系统，所以需要管理员权限，右键点击以管理员身份运行 cmd），不然会出现很多报错。

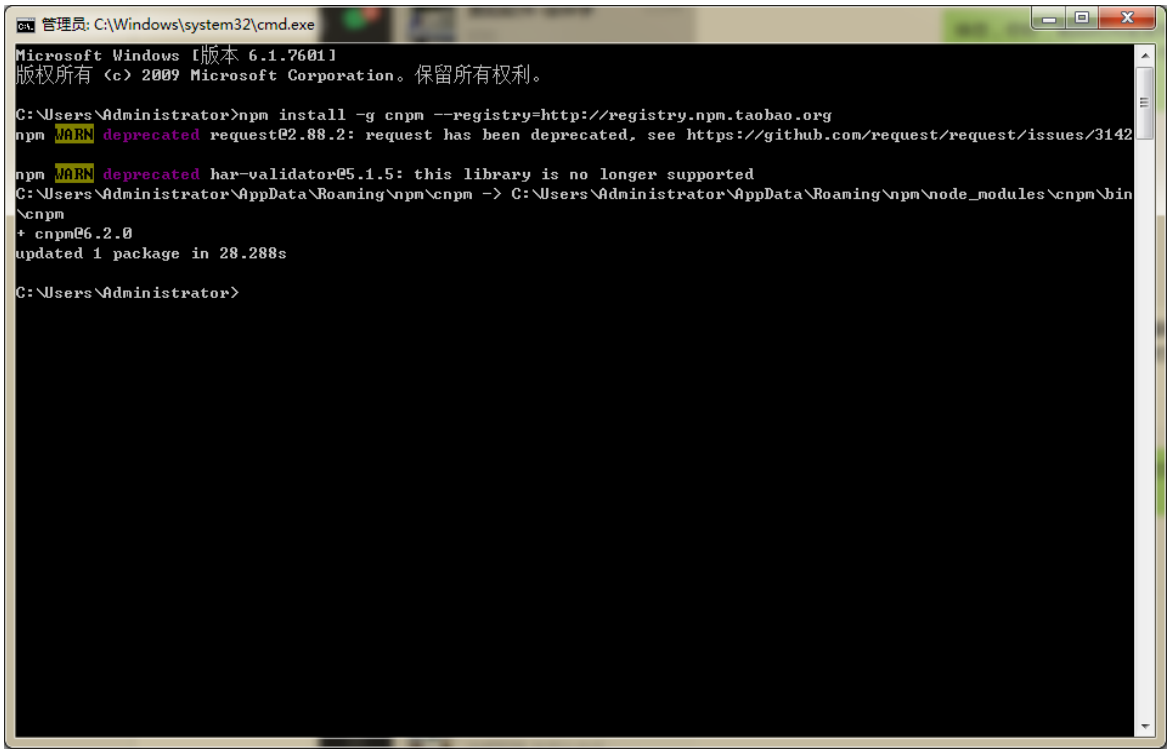
第 1 步，下载完毕后，可以安装 node，建议不要安装在系统盘（如 C：）。

第 2 步，安装完 Nodejs 后，打开电脑终端查看 Node 版本。以 Windows 为例，打开 cmd 命令窗口输入：node -v，查看 node 版本。



第 3 步，安装 cnpm 淘宝镜像。淘宝镜像的 cnpm 来代替 npm 的安装，速度会快很多。打开终端窗口，并输入以下命令：

npm install -g cnpm --registry=http://registry.npm.taobao.org。



安装过程持续 5 分钟左右，耐心等待。

第 4 步，基本环境搭建完成后就可以开始 Vue 开发了。首先需要安装 vue 的脚手架工具：vue-cli，然后初始化一个基本项目。输入：

```
#安装全局 vue-cli
npm install  -global vue-cli
#cmd 进入项目目录
cd D:\ vue-zrshiro
#安装项目依赖
npm install
# 运行测试，访问 http://localhost:8081
npm run dev
# 项目打包
npm run build
```

2.3 安装 Visual Studio Code

去官网上下下载：https://code.visualstudio.com/。下载之后，双击安装，安装完之后左侧栏那边是英文，如何变为中文：按快捷键 ctrl+shift+p，输入配置显示语言或者英文 Configure Display Language，安装完成之后，重启即可。

第三章　后台开发环境搭建

注：支持 jdk1.8 及 jdk1.8 以上的所有版本，满足此条件的 java 开发环境都可以集成我们的基础开发引擎，已经搭建好开发环境的跳过本章。

3.1　基础开发引擎项目的各软件版本

JAVA 运行环境：jdk1.8 及以上

SpringBoot：2.5.3

3.2　用 Eclipse+jdk1.8 搭建 Java 开发环境

我们的程序支持所有的 java 开发环境，已经搭建好开发环境的忽略本节。其它开发工具的搭建不再介绍，请自行查找相关资料。

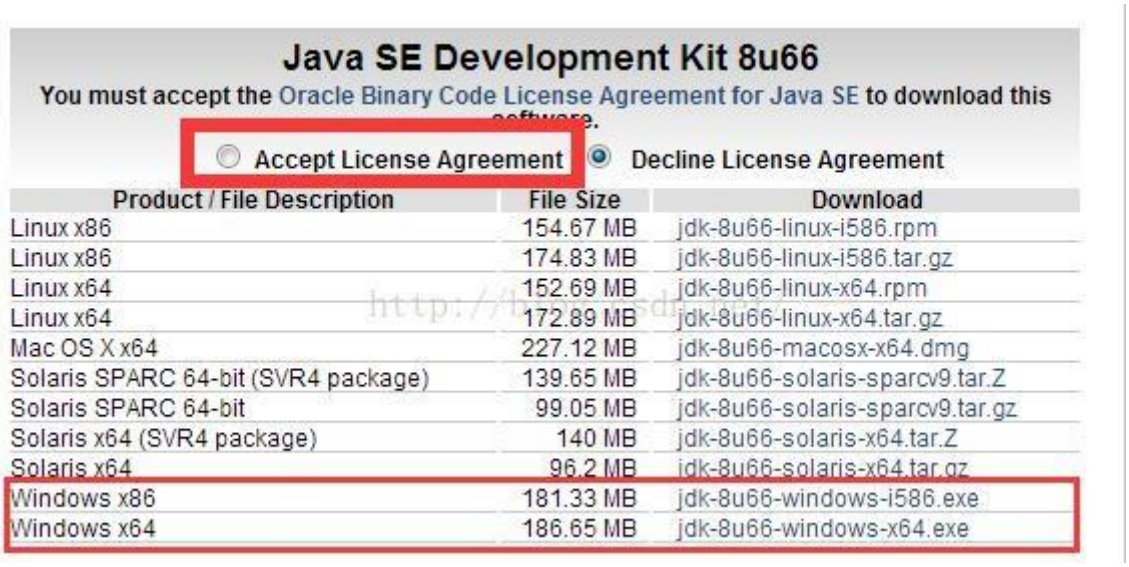
一、前期的准备

1.jdk1.8 的下载

(1)百度搜索 jdk，点击方框里的第一个。

(2)下载 jdk。

(3)选中 Accept License Agreement,然后在 Winows x86（32 位系统的下这个）和 Windows x64（64 位系统的下这个）中选择 一个下载，如果不清楚自己的系统是 32 位还是 64 位，可以百度，或者直接下载 32 位的 Windows x86。

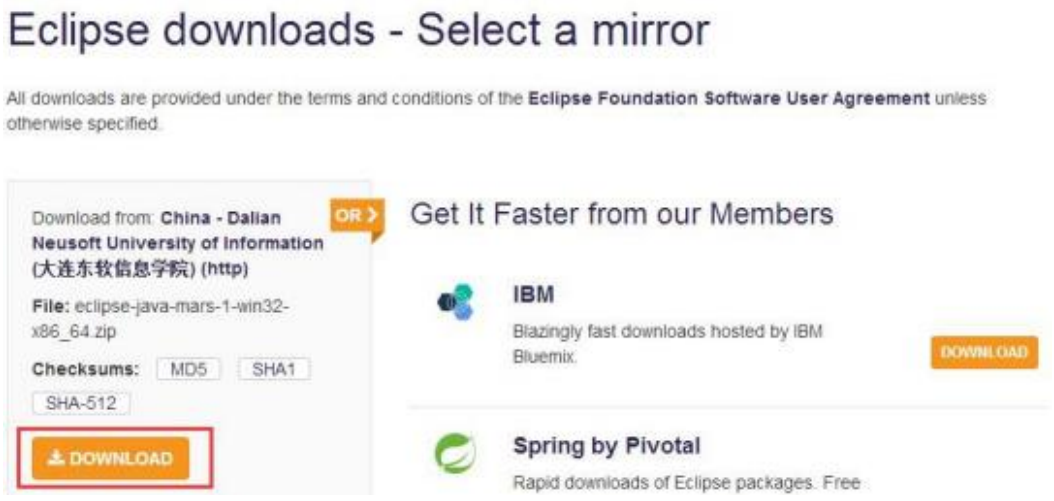


2.Eclipse 的下载

(1)百度搜索 Eclipse，选择第一个。

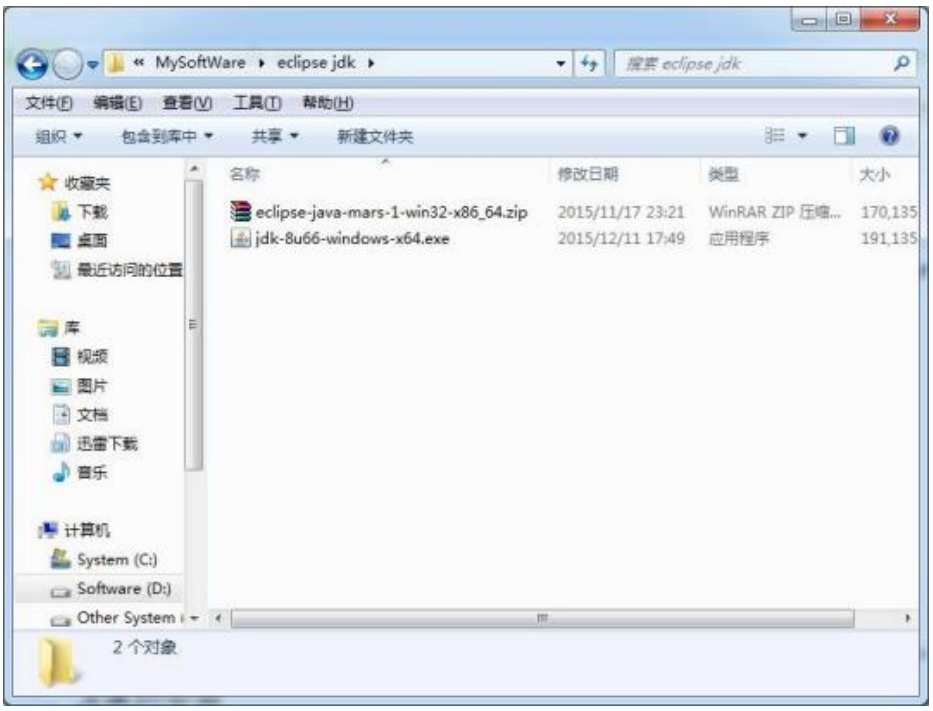
(2)选择方框里边的内容进行下载，这个必须和前面下载的 JDK 版本保持一致， 前边下载的 32 位 JDK 这里就点击 32bit,前边下载的 64 位 JDK，这里就点击 64bit。

(3)继续点击方框中的内容，开始下载。



二、JDK 的安装

1.下载的文件在这里，双击 jdk-8u66-windows-x64.exe 开始安装。



2.平时我们安装软件不要安装在 C 盘，容易导致开机慢，改到 D 盘，点击“更改”。


Java SE Development Kit 8 Update 66 (64-bit) - 定制安装
Java
ORACLE
从下面的列表中选择要安装的可选功能。您可以在安装后使用控制面板中的"添加/删除程序"实用程序更改所选择的功能
开发工具
源代码
公共 JRE
功能说明
Java SE Development Kit 8 Update 66 (64-bit), 包括 JavaFX SDK, 一个专用 JRE 以及 Java Mission Control 工具套件。它要求硬盘驱动器上有 180MB 空间。
安装到:
C:\Program Files\Java\jdk1.8.0_66\
更改(C)...
< 上一步(B)
下一步(N) >
取消

把 C 盘改成 D 盘:


Java SE Development Kit 8 Update 66 (64-bit) - 更改文件夹
Java
ORACLE
浏览至新目标文件夹
查找(L):
jdk1.8.0_66
1
文件夹名:
d:\Program Files\Java\jdk1.8.0_66\
2
确定
取消

下一步:

开始安装...

3. 继续安装 Jre，点击“更改”。

找到刚才我们安装 JDK 的路径，D:\Program Files\Java,并在此目录下新建一个文件夹 jre1.8.0_66，并选中。

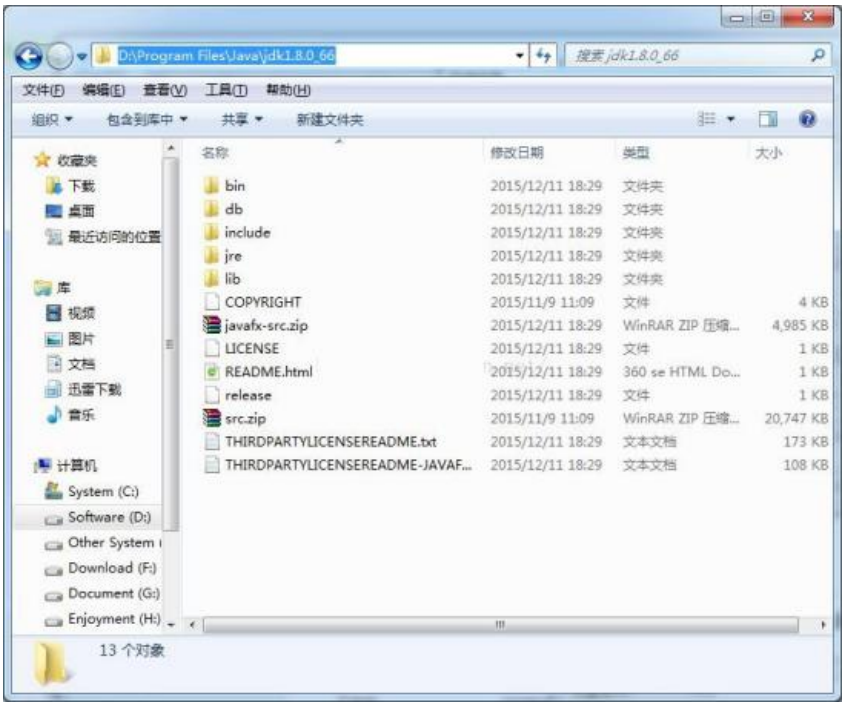
下一步:

jre 安装中...

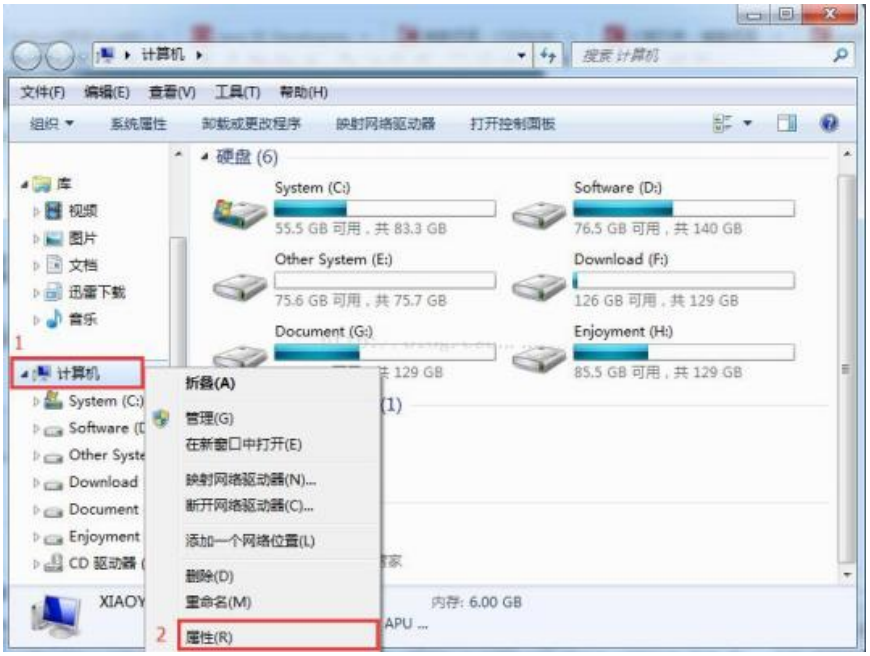
4. 安装完成，点击关闭。

三、JDK 环境变量的配置

1.找到自己 JDK 的安装目录， 复制其安装路径， 我的是 D:\Program Files\Java\jdk1.8.0_66。



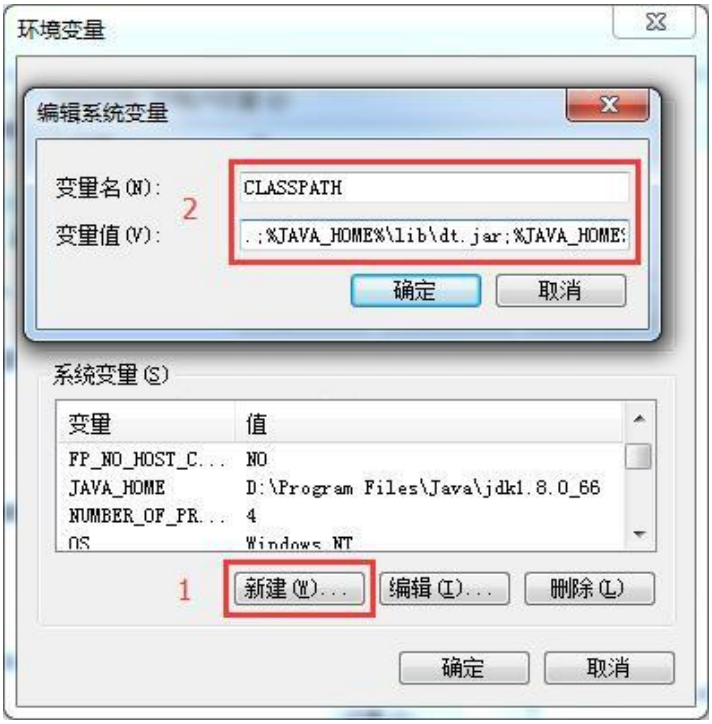
2.在计算机上右键--属性。



3. 按下图的顺序依次点击高级系统设置--环境变量--新建，在变量名中填写 JAVA_HOME, 在变量值中填写第 2 步中复制的路径，并确定。

4. 再次点击新建，变量名填 CLASSPATH, 变量值填下边这个，最好是直接复制我的，每一个标点符号都要复制下去，并确定。

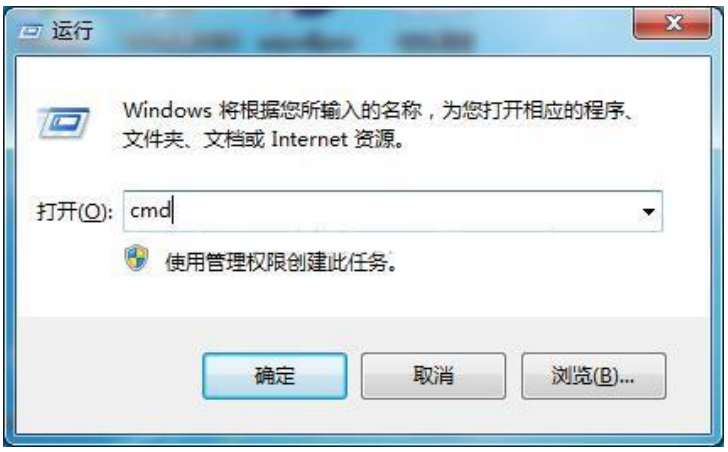
.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;



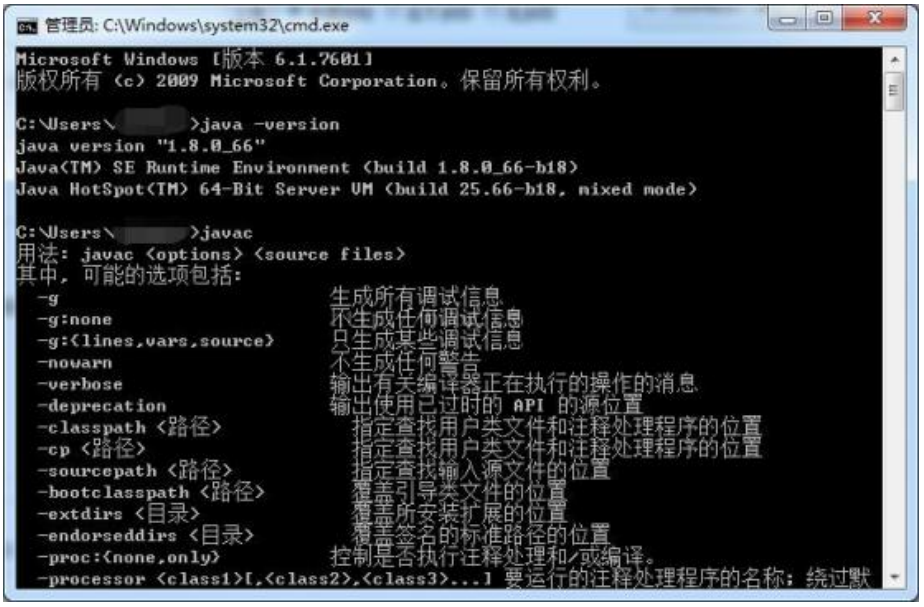
5.在下方的系统变量中找到 Path,如方框 1 所示，选中填并点击编辑，在变量值的最前边添加下面这一行（注意是添加，千万别把以前本来有的内容删掉了），也是最好直接复制我的，每一个标点符号都要复制下去，添加完成后一直点确定直到所有窗口关闭。

%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;

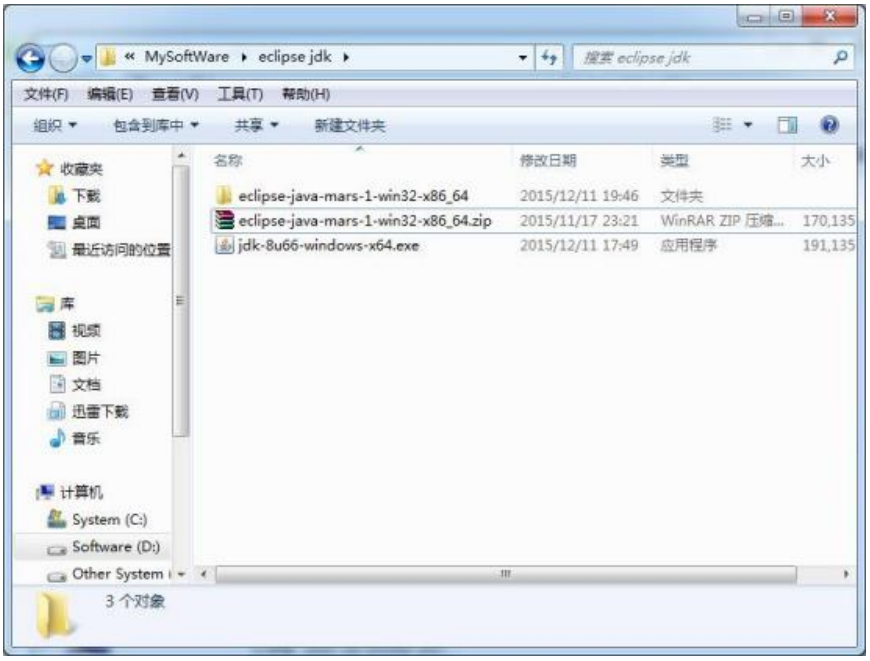
6. 同时按住 Windows 图标键+R 键，弹出如下窗口，输入 cmd，并回车。



7. 依次输入 java -version 和 javac 并回车，如果出现一下信息则表明 JDK 安装成功！阶段性的胜利！

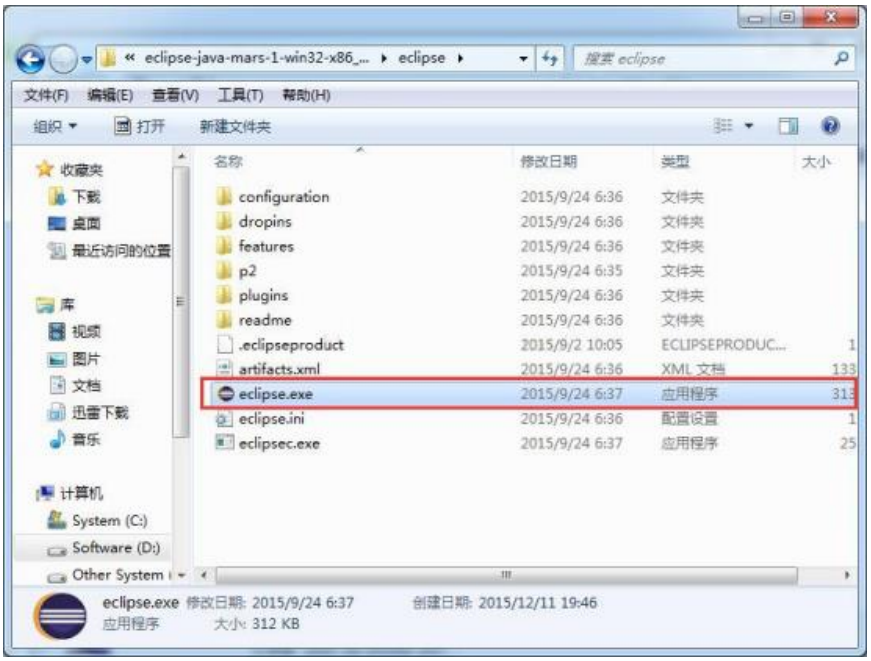


四、Eclipse 的安装配置

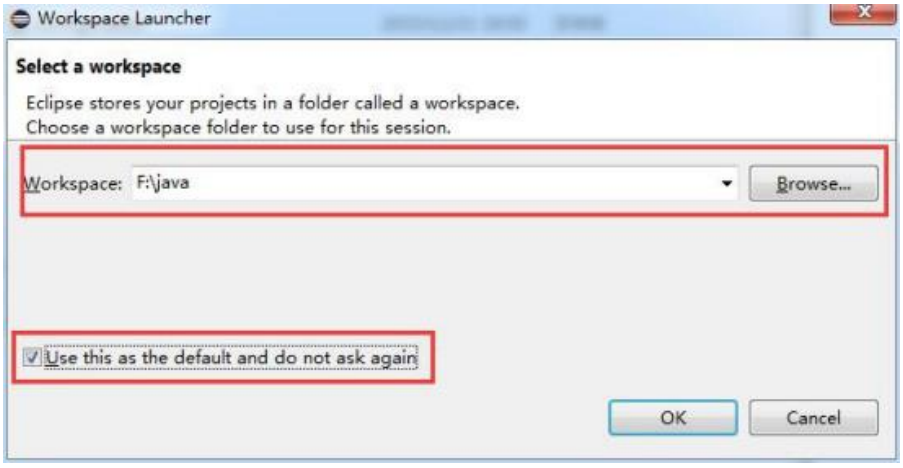


1.解压开下载好的 Eclipse。

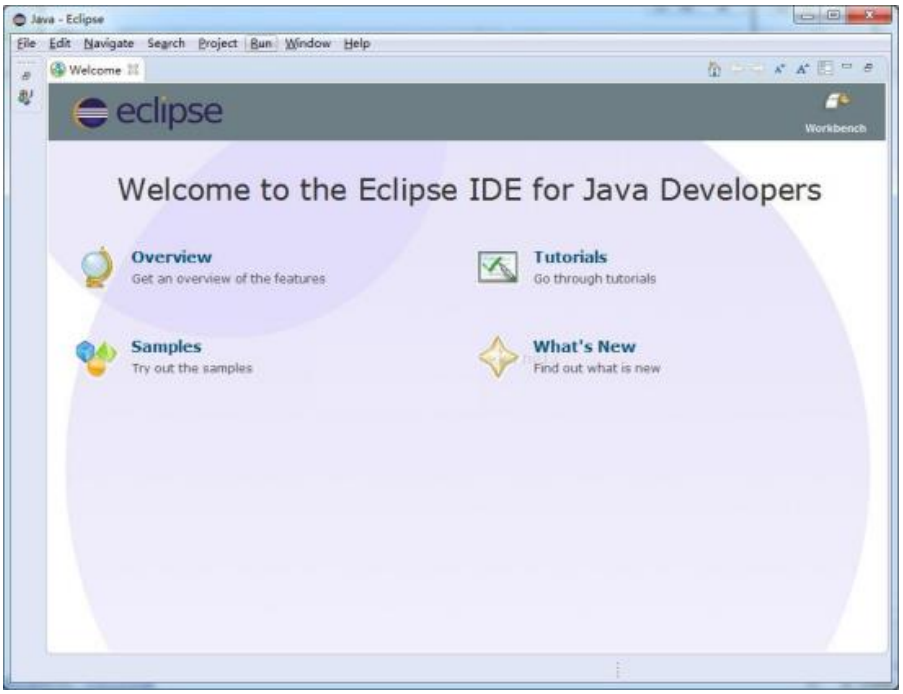
2.进入到这个目录，并双击 eclipse.exe 启动 Eclipse。



3.这里是要你选择工作区间，也就是你代码存放的地方，建议在 E 盘、F 盘等位置新建一个文件夹来单独存放代码，并选择左下方的勾，并 OK。



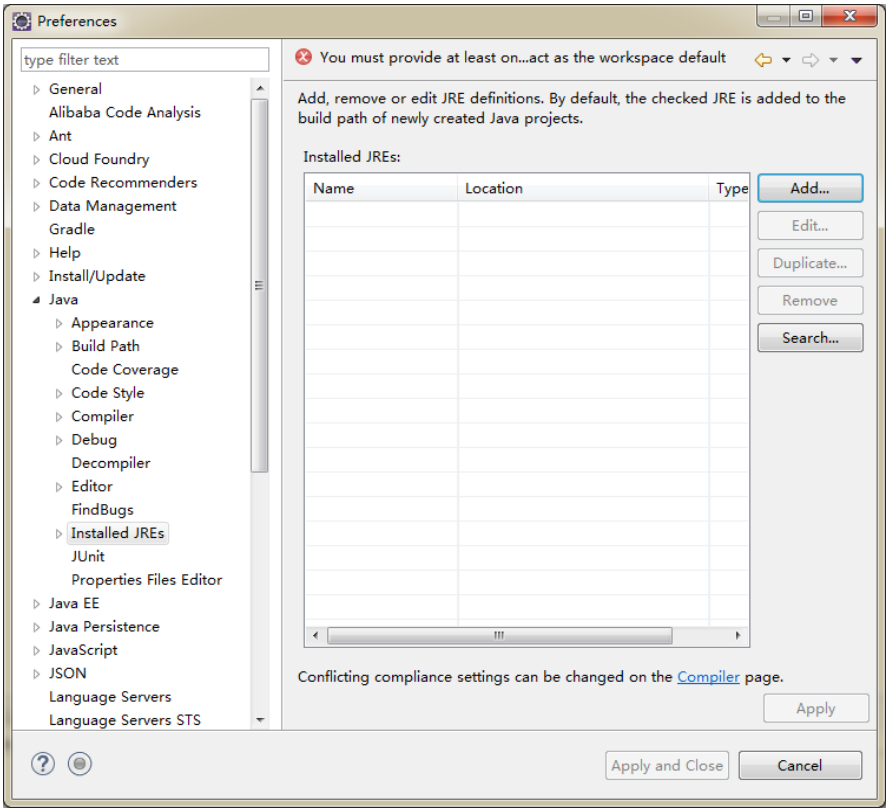
4. 安装成功！



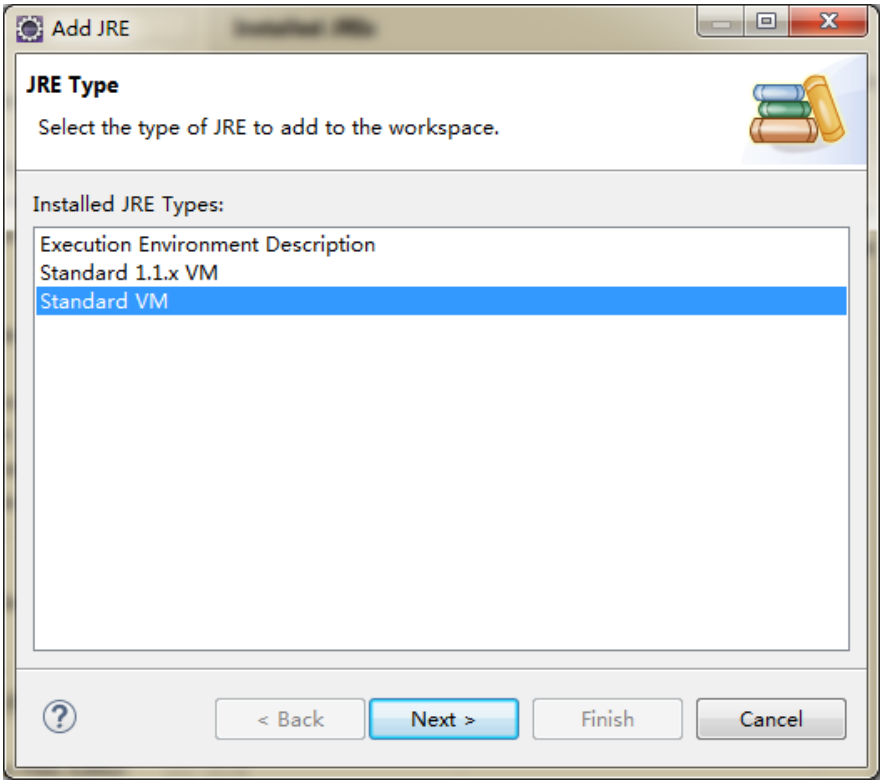
3.3 Eclipse 开发工具配置 jdk

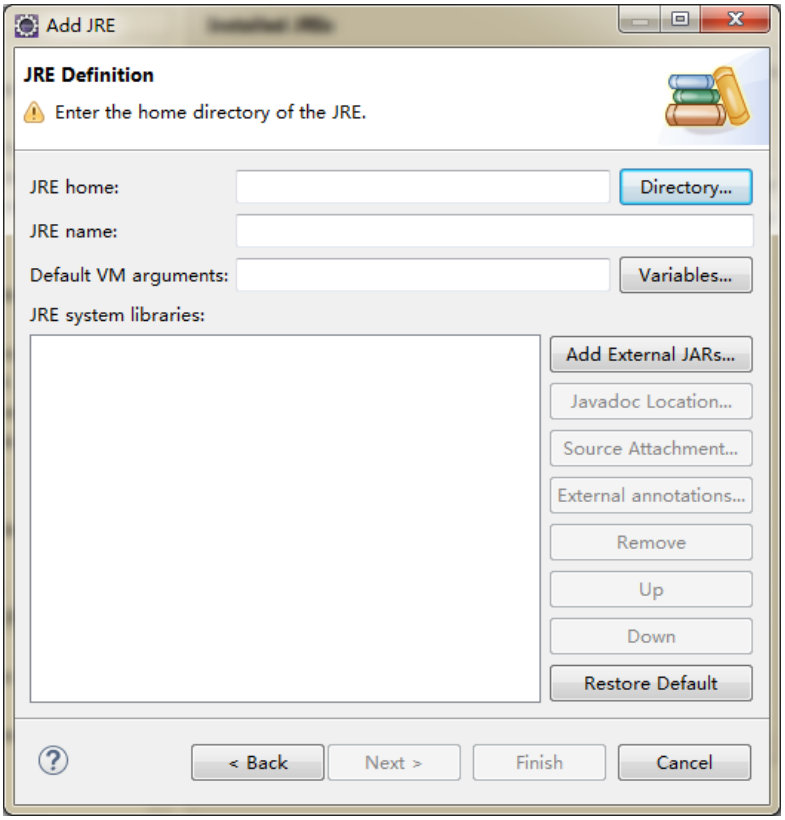
Eclipse 开发工具是要编译并运行 java 项目的，所以需要在 Eclipse 开发工具上配置 jdk 环境。

打开 Eclipse，单击“Window”菜单，选择下方的“Preferences”选项。



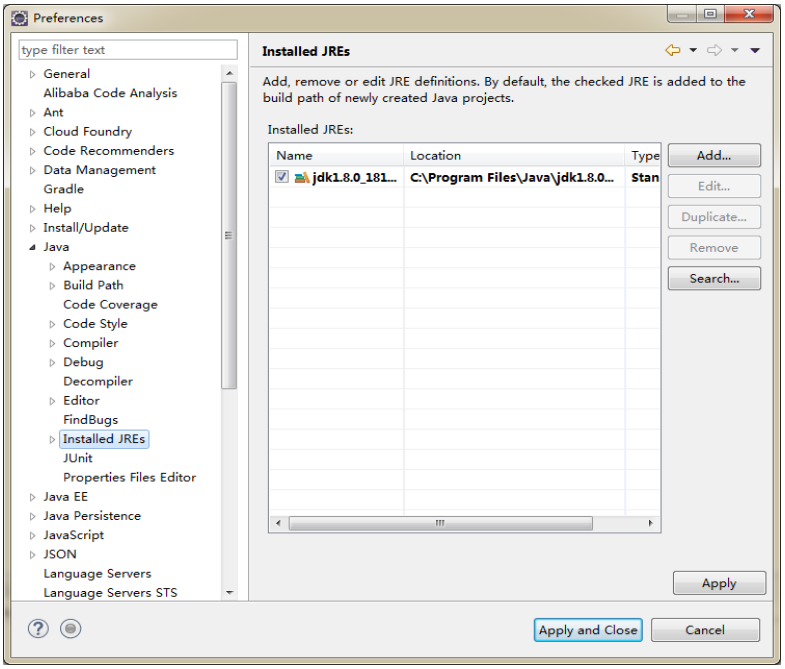
点击 Java -> Installed JREs，在右侧点击 Add 按钮，选择 Standard VM，点击 Next。





JRE home 为当前 jdk 安装的路径，点击 Directory 选择当前 jdk 安装的路径（D:\Program Files\Java\jdk1.8.0_66），JRE name 为当前 jdk 的一个名称，默认 jdk1.8.0_66，然后点击 Finish。

说明：jdk 是带有编译和运行 java 的*.class 文件功能的，而 jre 只有运行 java 的*.class 文件功能，所以建议选择 jdk。



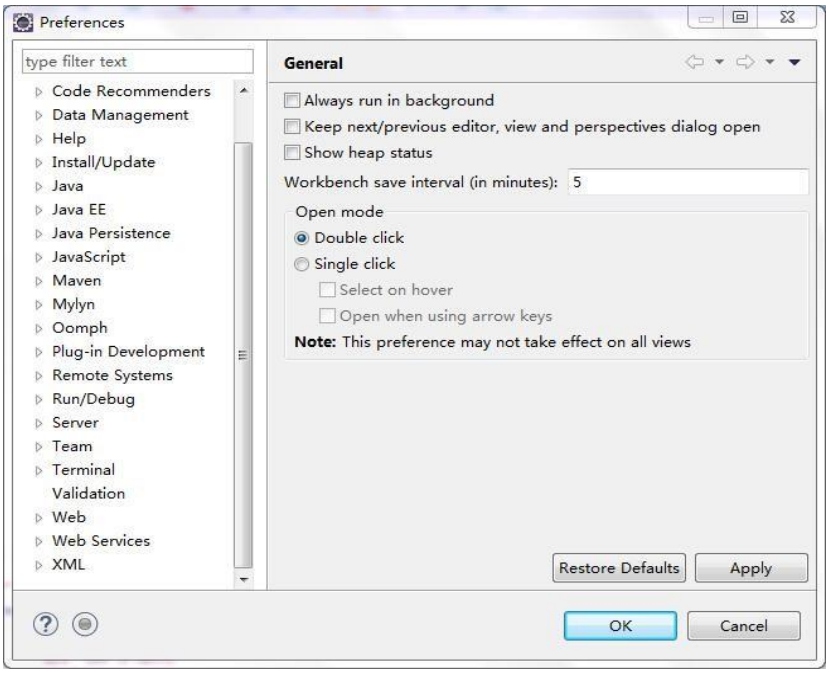
最后点击 Apply and Close 即可完成 Eclipse 开发工具中 jdk 的配置。

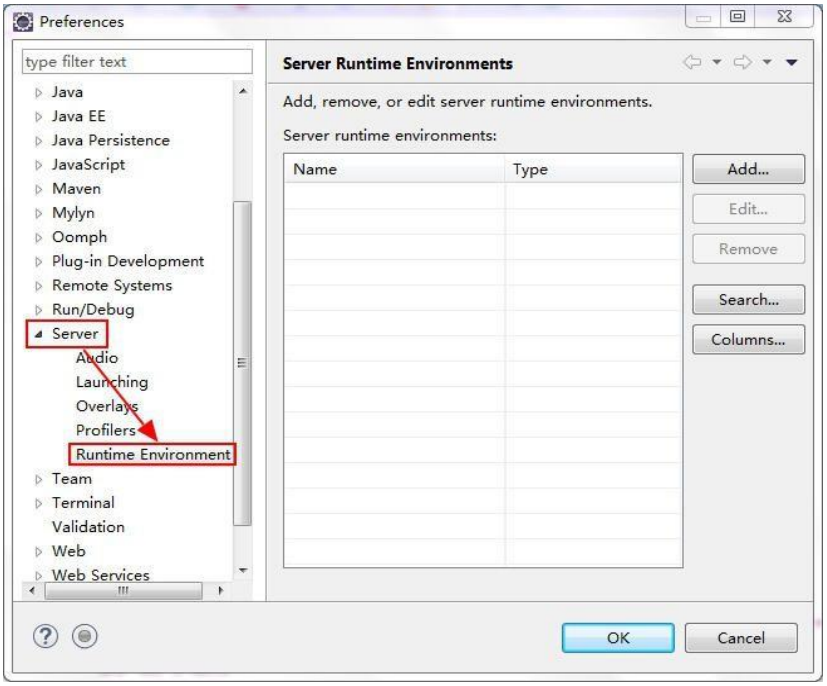
3.4 Eclipse 开发工具配置 tomcat

我们的程序支持所有的 java 开发环境，已经搭建好开发环境的忽略本节。其它开发工具的配置不再介绍，请自行查找相关资料。

Eclipse 开发工具配置 tomcat，并且把项目部署到 Tomcat 服务器上。

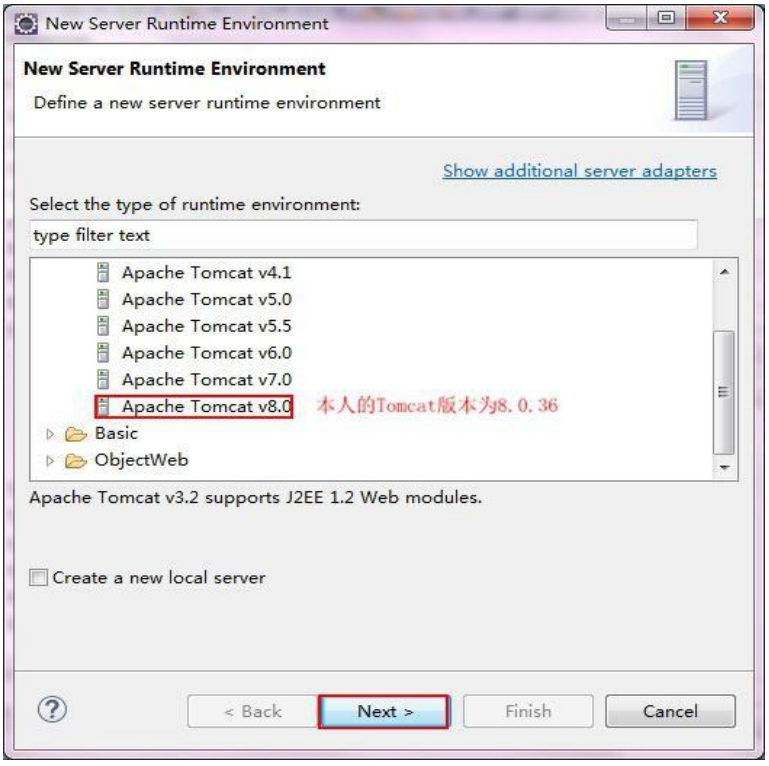
打开 Eclipse，单击“Window”菜单，选择下方的“Preferences”选项。



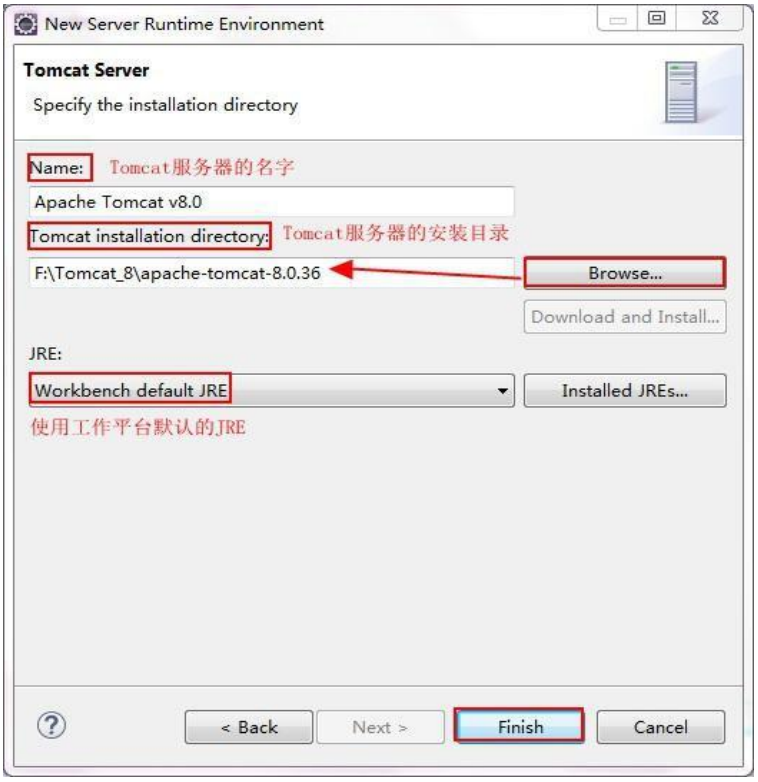


单击“Server”选项，选择下方的“Runtime Environments”。

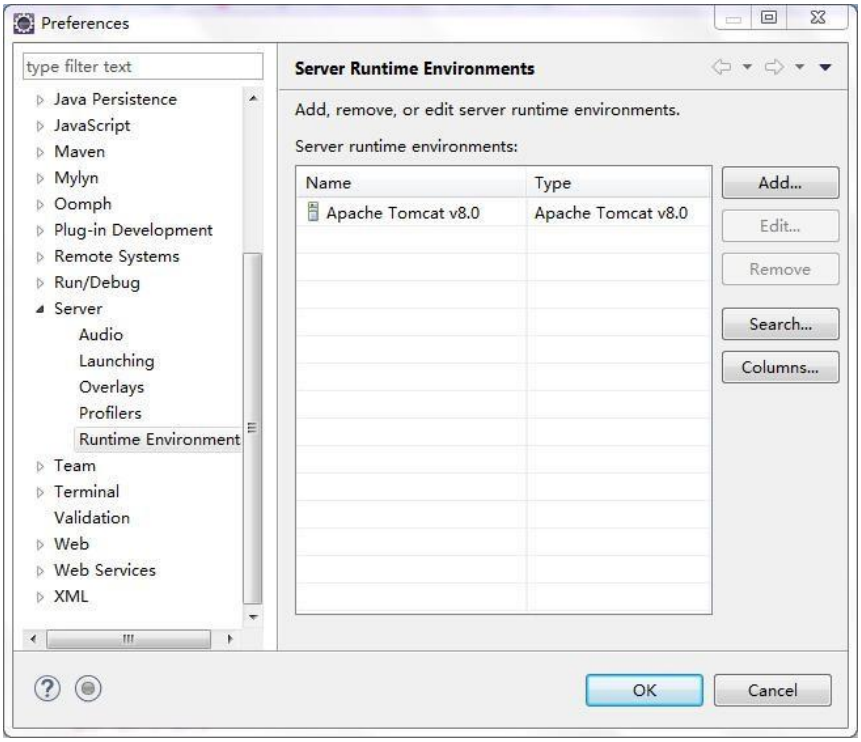
点击“Add”添加 Tomcat。



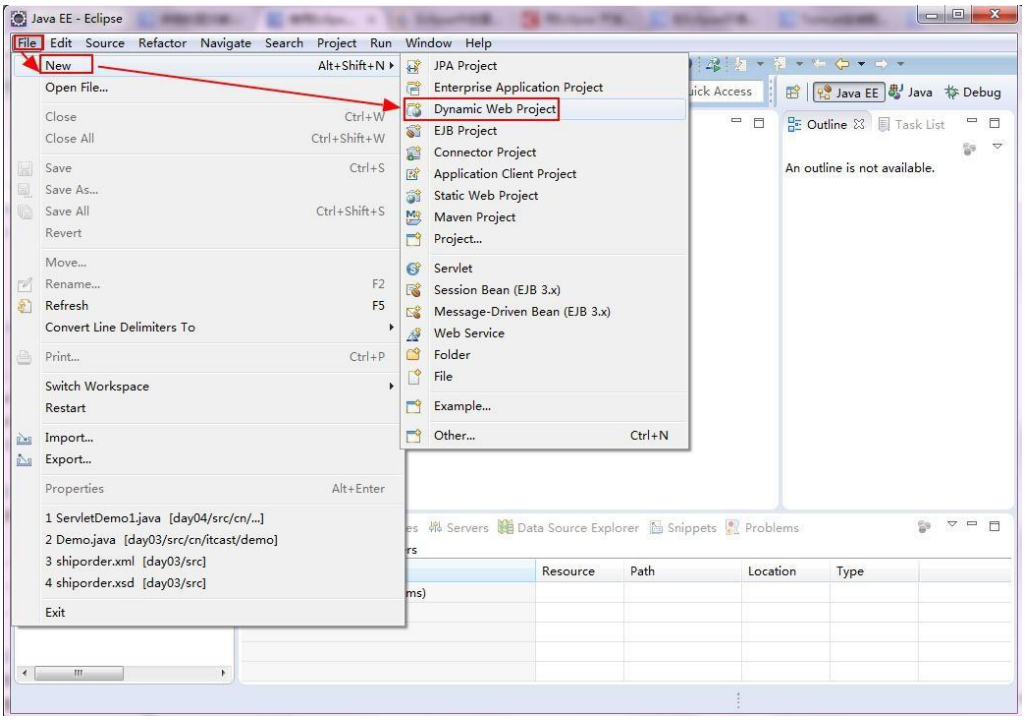
点击“Next”，选中自己安装的 Tomcat 路径。


New Server Runtime Environment
Tomcat Server
Specify the installation directory
Name:
Tomcat服务器的名字
Apache Tomcat v8.0
Tomcat installation directory:
Tomcat服务器的安装目录
F:\Tomcat_8\apache-tomcat-8.0.36
Browse...
Download and Install...
JRE:
Workbench default JRE
Installed JREs...
使用工作平台默认的JRE
< Back
Next >
Finish
Cancel

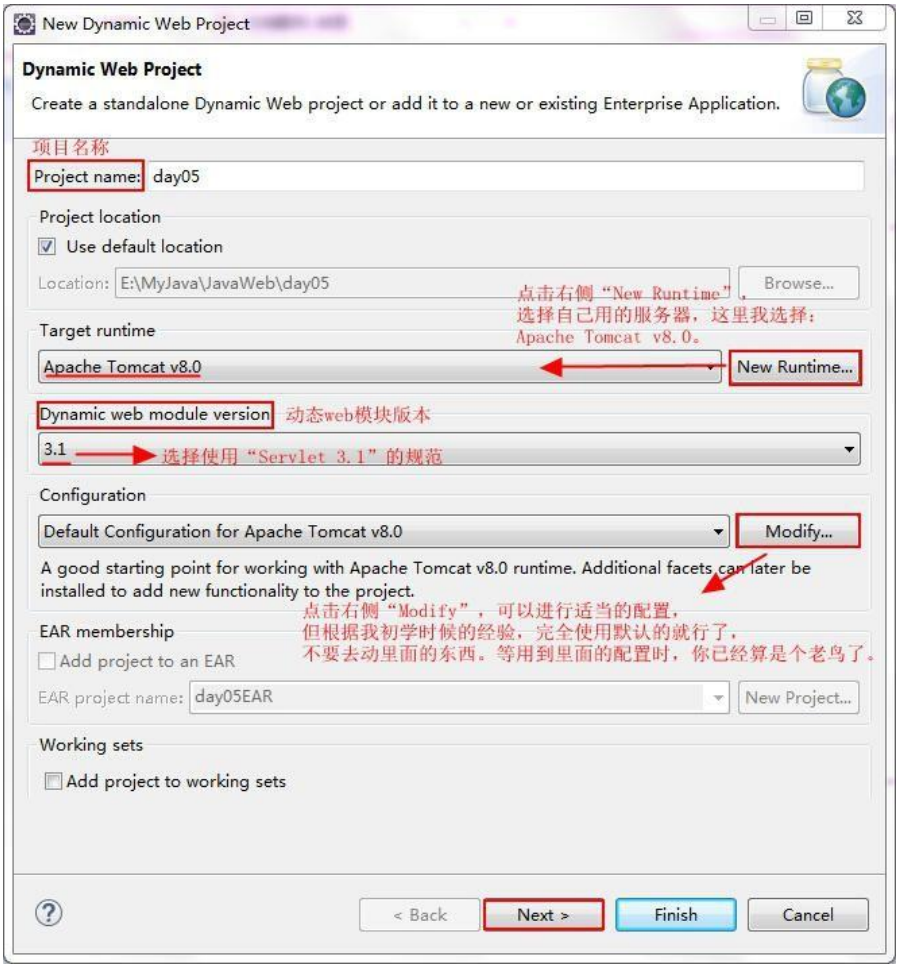
点击“Finish”完成。



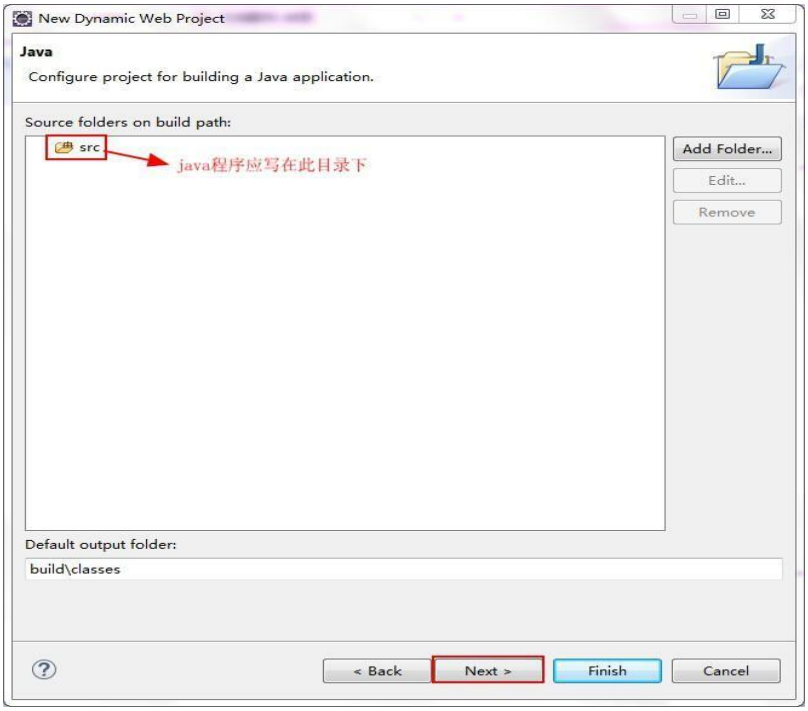
建立一个 Web 应用，File → New → Dynamic Web Project



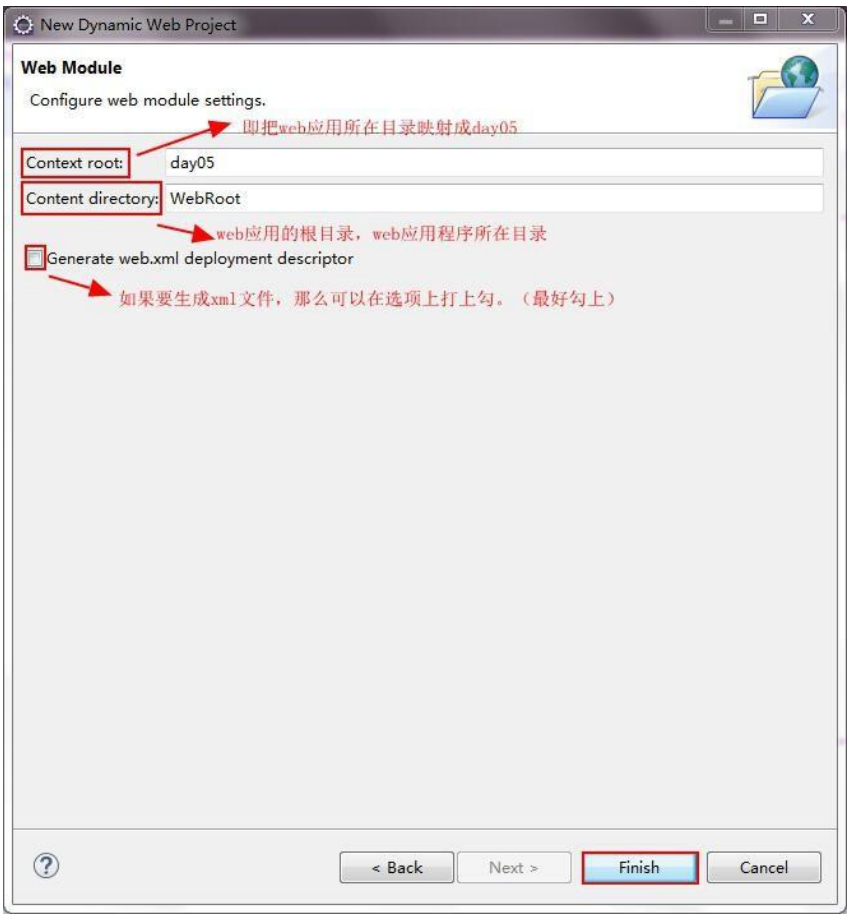
创建一个 Dynamic Web Project


New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
项目名称
Project name: day05
Project location
Use default location
Location: E:\MyJava\JavaWeb\day05
Browse...
点击右侧“New Runtime”，
选择自己用的服务器，这里我选择：
Apache Tomcat v8.0。
Target runtime
Apache Tomcat v8.0
New Runtime...
Dynamic web module version
动态web模块版本
3.1
选择使用“Servlet 3.1”的规范
Configuration
Default Configuration for Apache Tomcat v8.0
Modify...
A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.
点击右侧“Modify”，可以进行适当的配置，
但根据我初学时候的经验，完全使用默认的就行了，
不要去动里面的东西。等用到里面的配置时，你已经算是个老鸟了。
EAR membership
Add project to an EAR
EAR project name: day05EAR
New Project...
Working sets
Add project to working sets
< Back
Next >
Finish
Cancel

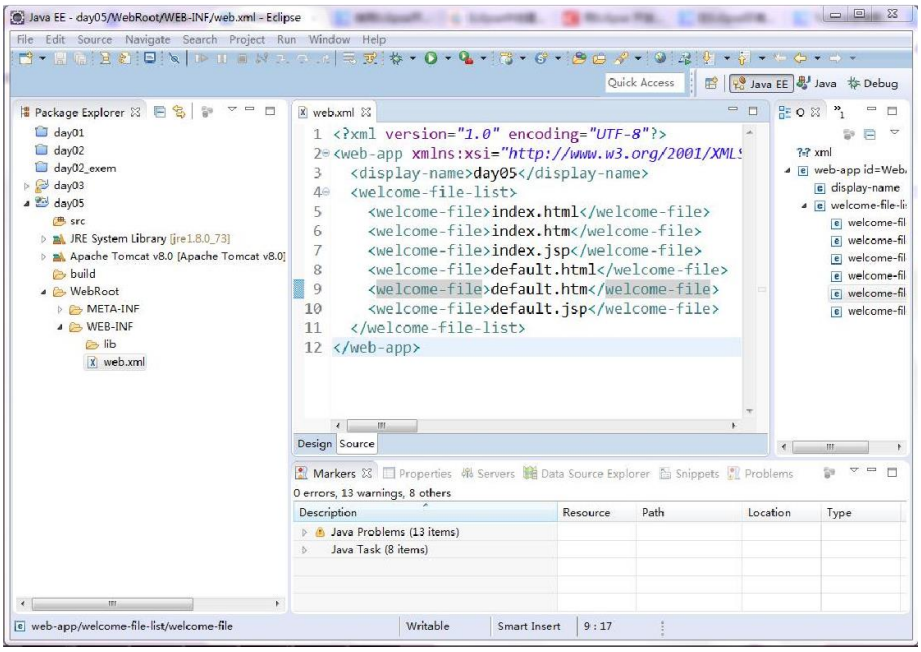
点击“Next”下一步


New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src
java程序应写在此目录下
Add Folder...
Edit...
Remove
Default output folder:
build\classes
< Back
Next >
Finish
Cancel

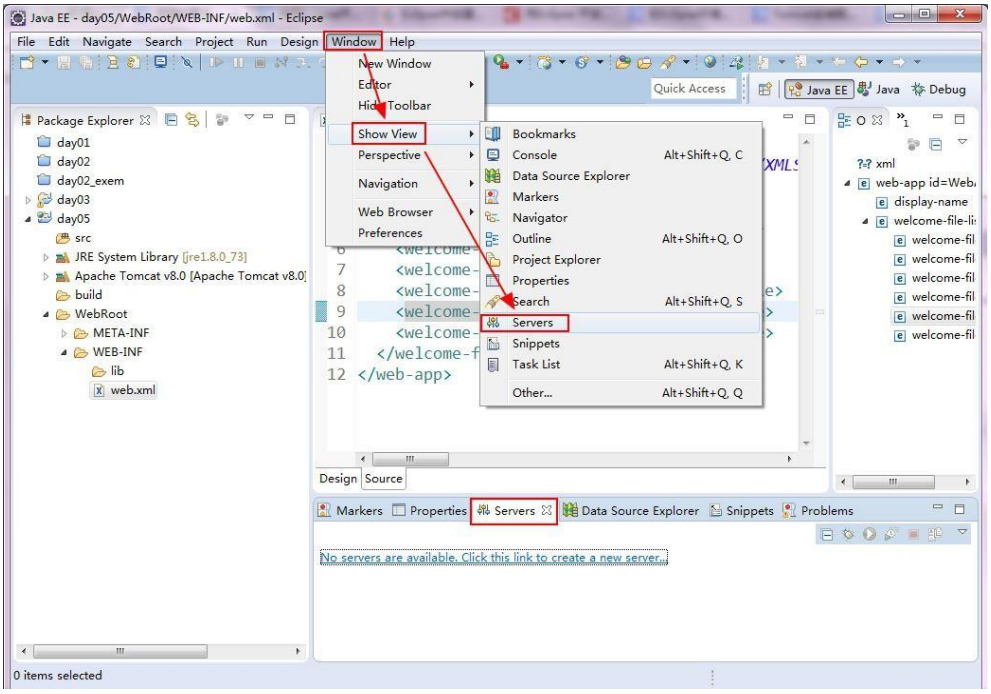
点击“Next”下一步



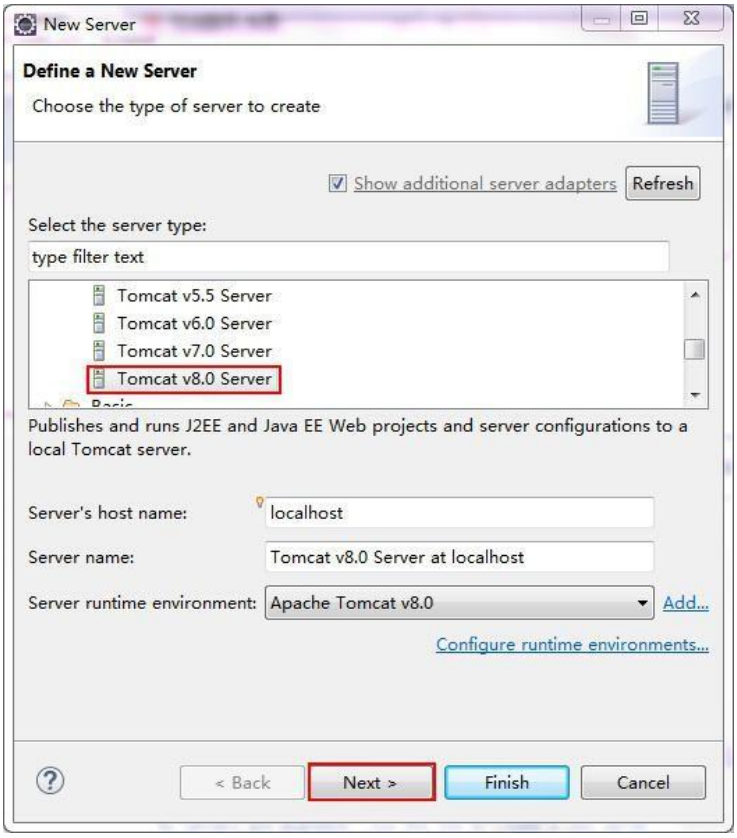
点击“Finish”完成



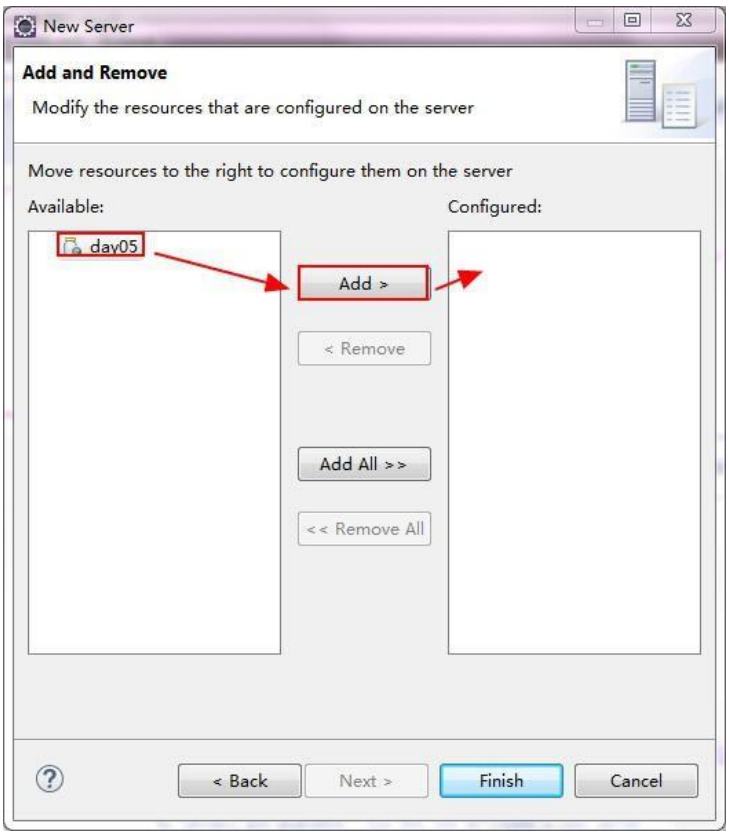
让 Tomcat 服务器显示在控制台上，将 Web 应用部署到 Tomcat 中。1.Window → Show View → Servers



点击链接 No servers are available. Click ths link to create a new server.，在弹出的对话框中选择 Tomcat 版本。

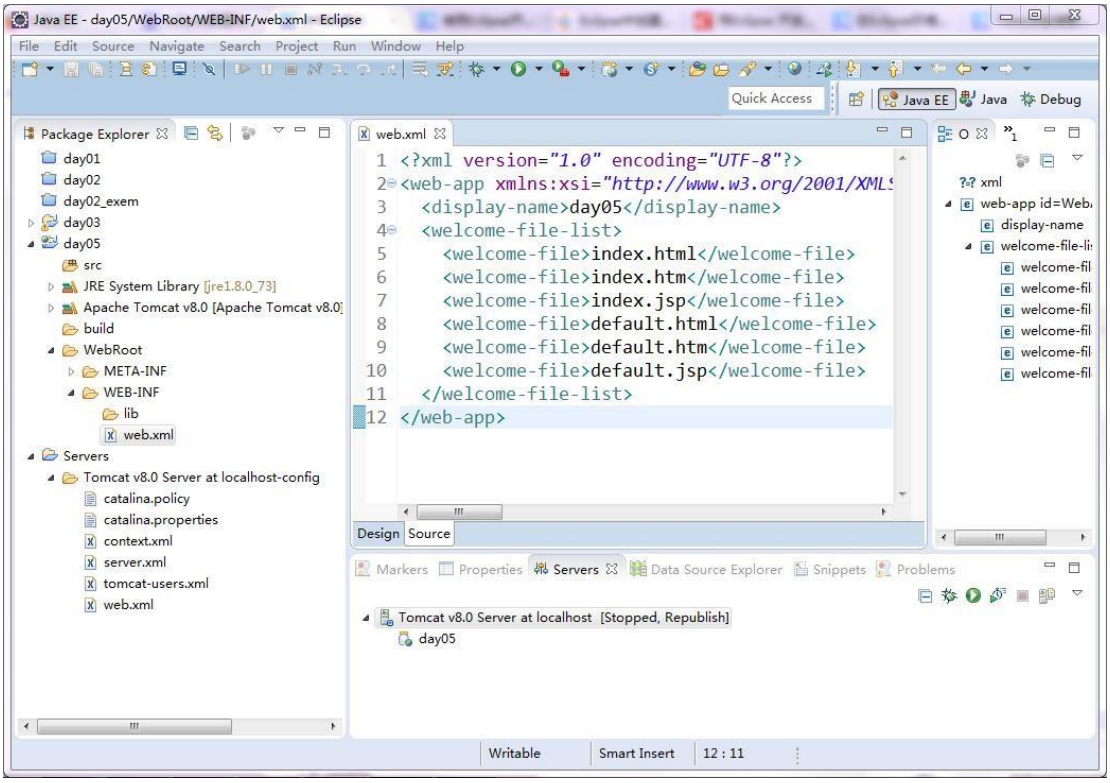


点击“Next”，添加我们的项目。



选中项目并点击 Add，或是双击都可以添加到右边。

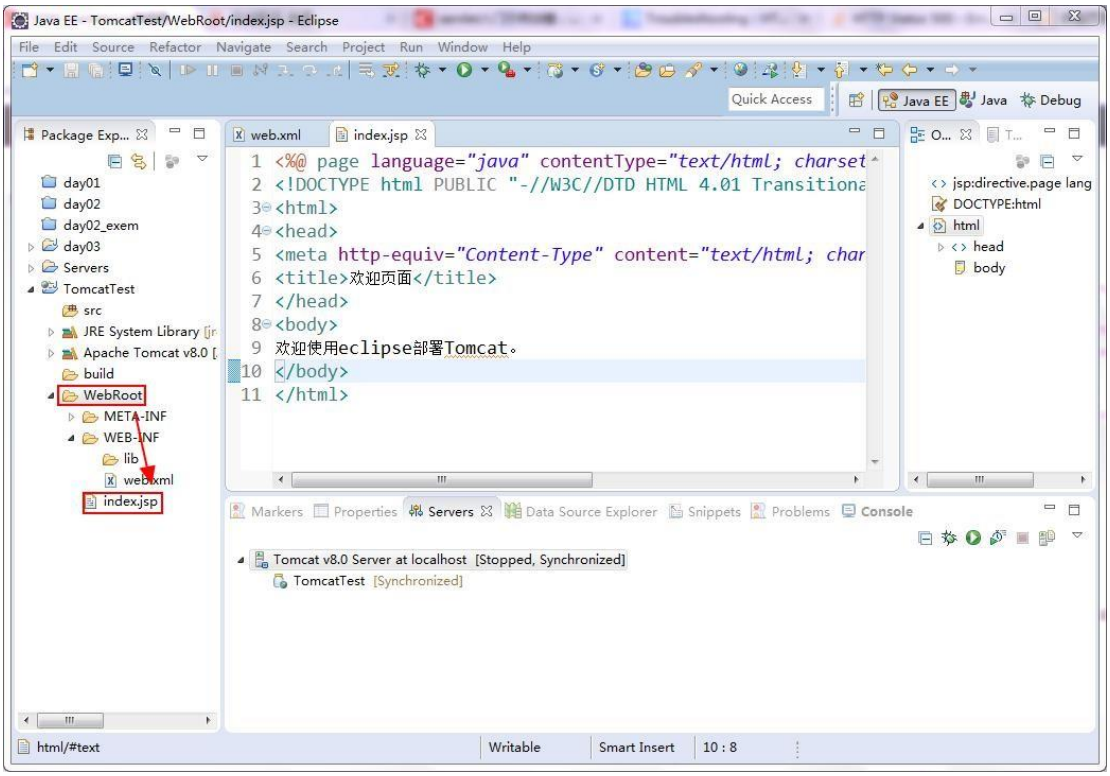
点击“Finish”完成。



返回下方的“Servers”面板，右键单击该面板中的“Tomcat v8.0 Server at localhost”节点，在弹出的快捷菜单中单击“Start”，即可启动指定的 Web 服务器。如果此时直接启动访问 http://localhost:8080/TomcatTest ，会发现会报 404 的错误。这是因为我们没有添加主页，下面添加主页(index.jsp) 的内容：

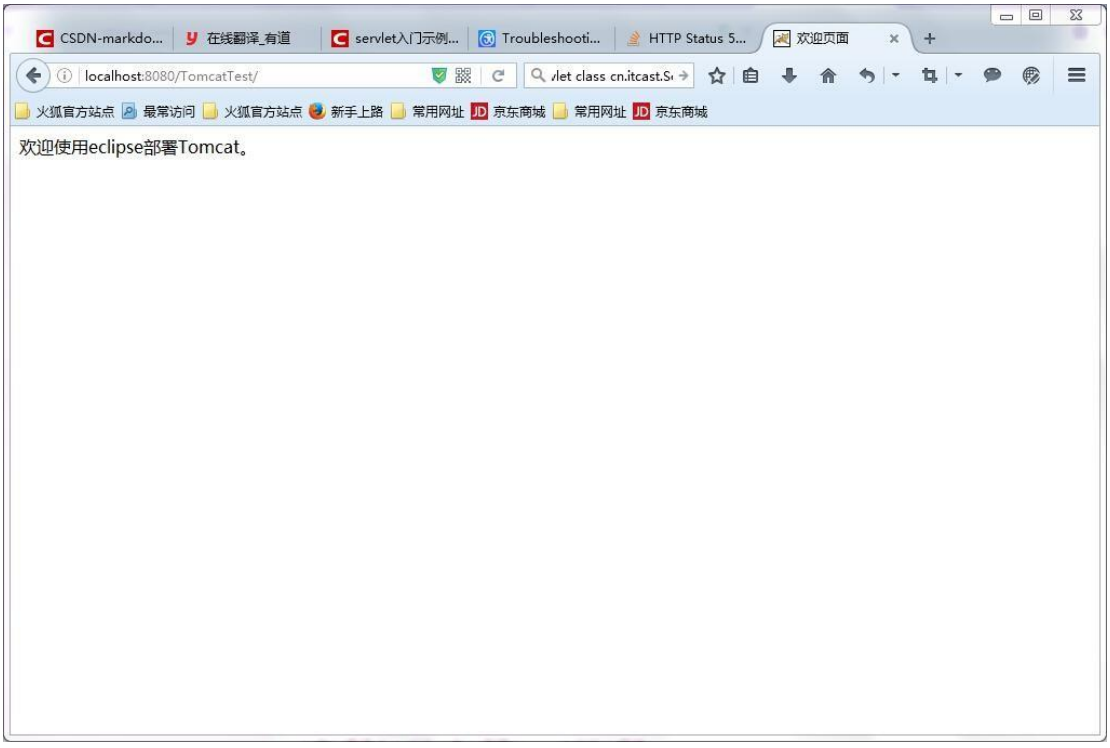
```
<%@page language="java" contentType="text/html; charset=utf-8"
pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>欢迎页面</title>
</head>
<body>
欢迎使用 Eclipse 部署 Tomcat。
</body>
```

```
</html>
```



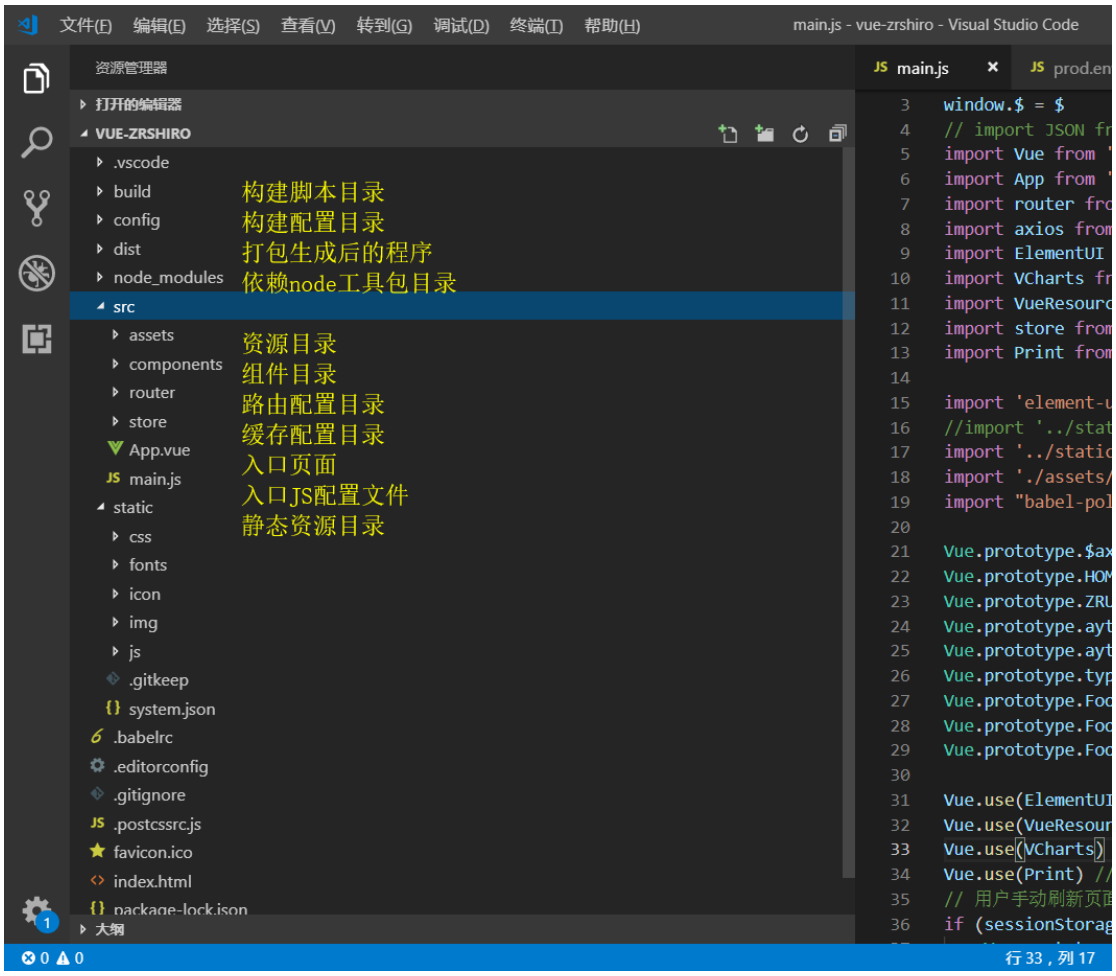
注意：web 资源一定要在 WebRoot 目录下添加。如图：

此时，再一次来访问该链接：http://localhost:8080/TomcatTest ，效果如下：



第四章　VUE 配置及说明

4.1　框架整体



4.2　config 目录

4.2.1　dev.env.js

用于管理开发测试环境的后台接口配置。

4.2.2 index.js

```
const path = require('path')

module.exports = {
  dev: {

    // Paths
    assetsSubDirectory: 'static',
    assetsPublicPath: '/',
    proxyTable: {
      //数据接口地址
      '/api':{
          target:'http://localhost:8083/',//http://localhost:8083/
          changeOrigin:true,
          pathRewrite:{
              '^/api': ''//'/api':''
          }
      }
    },
    // Various Dev Server settings
    host: 'localhost', // can be overwritten by process.env.HOST
    port: 8081, // can be overwritten by process.env.PORT, if port is in use, a free one will be determined
    autoOpenBrowser: false,
    errorOverlay: true,
    notifyOnErrors: true,
    poll: false, // https://webpack.js.org/configuration/dev-server/#devserver-watchoptions-
```

开发及测试环境下的api接口地址

开发及测试环境下的访问地址

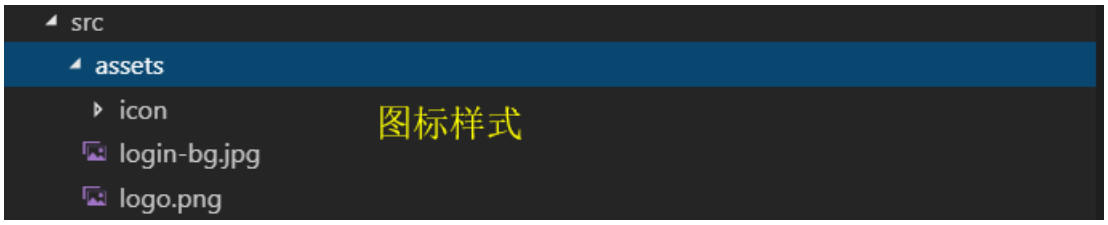
开发及测试环境下的访问端口

4.2.3 prod.env.js

用于管理正式环境打包的后台接口配置。

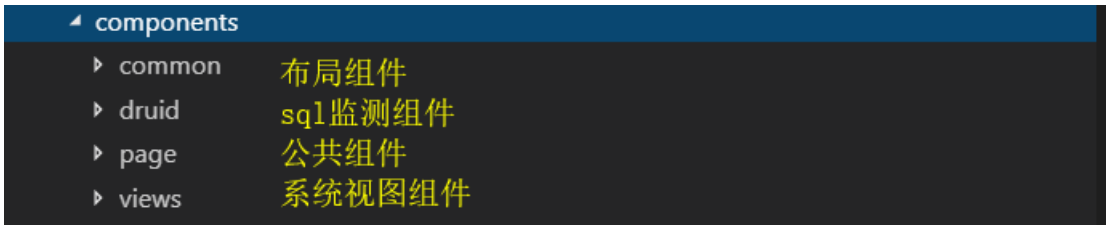
4.3 src 目录

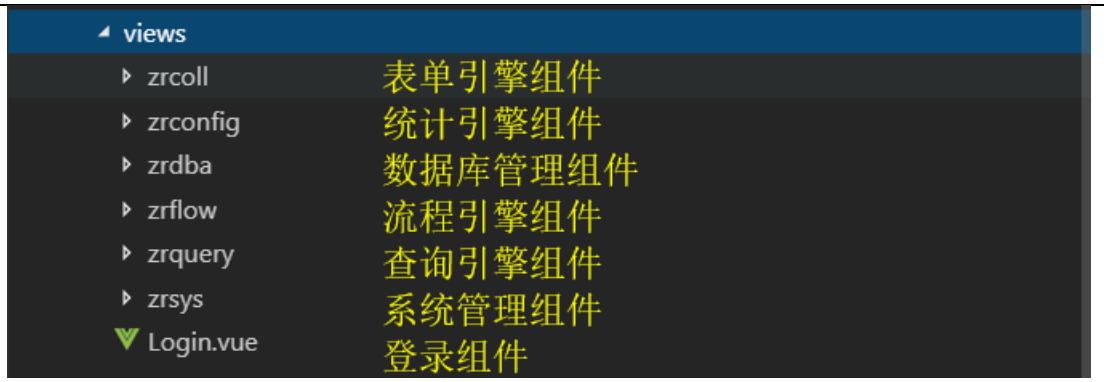
4.3.1 assets 资源目录



4.3.2 components 组件目录

系统框架主要组件。





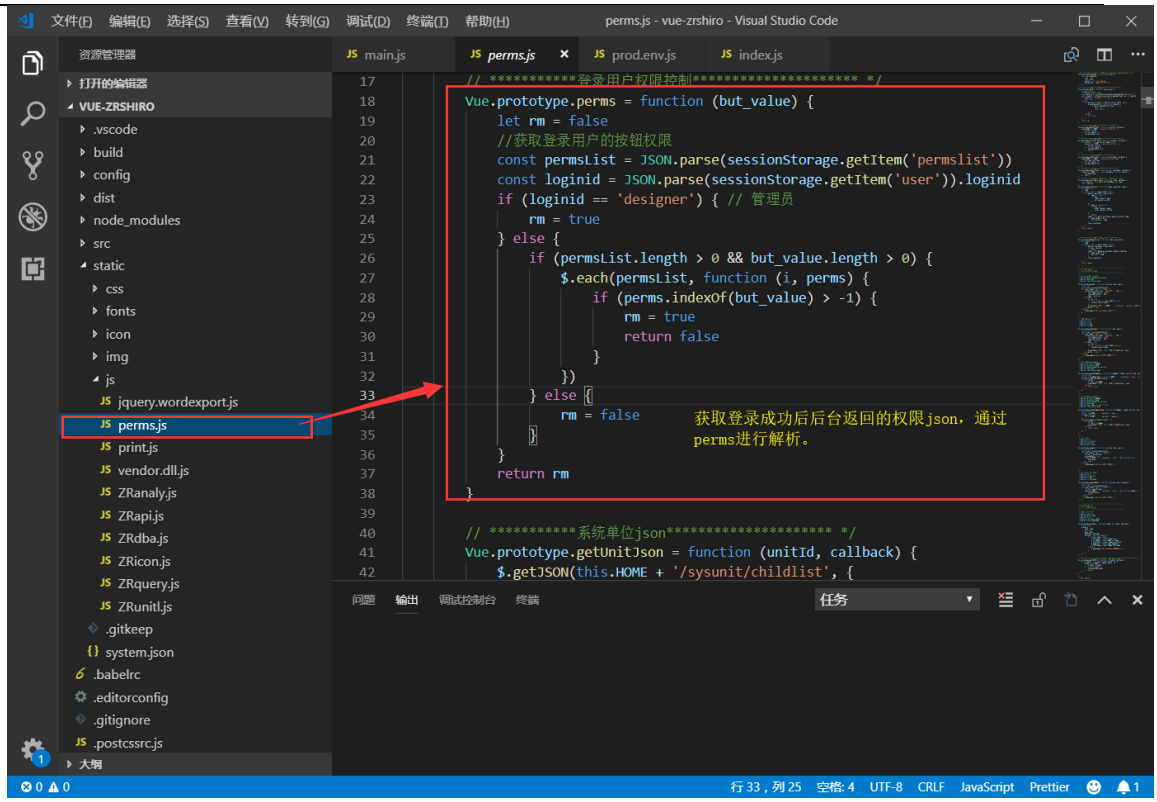
4.3.3 router 静态路由目录

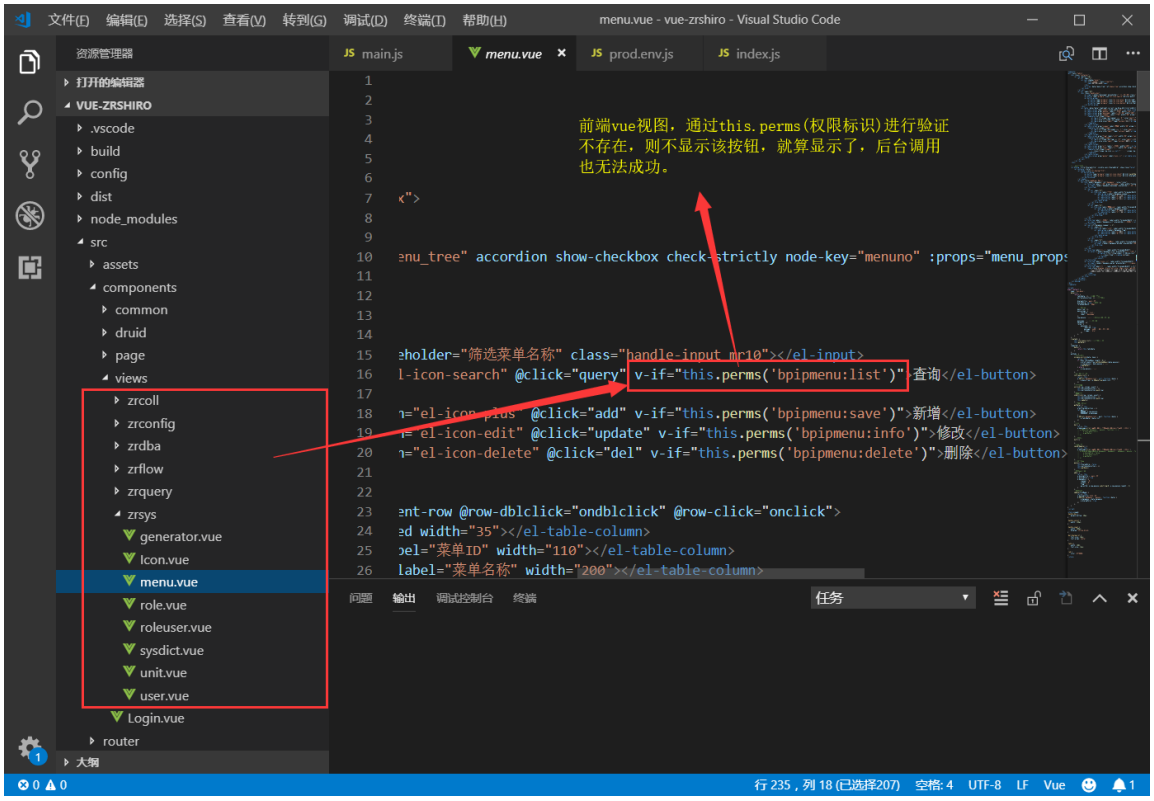
用于配置静态路由。

4.3.4 store 动态路由目录

4.4 前端 shiro 权限控制

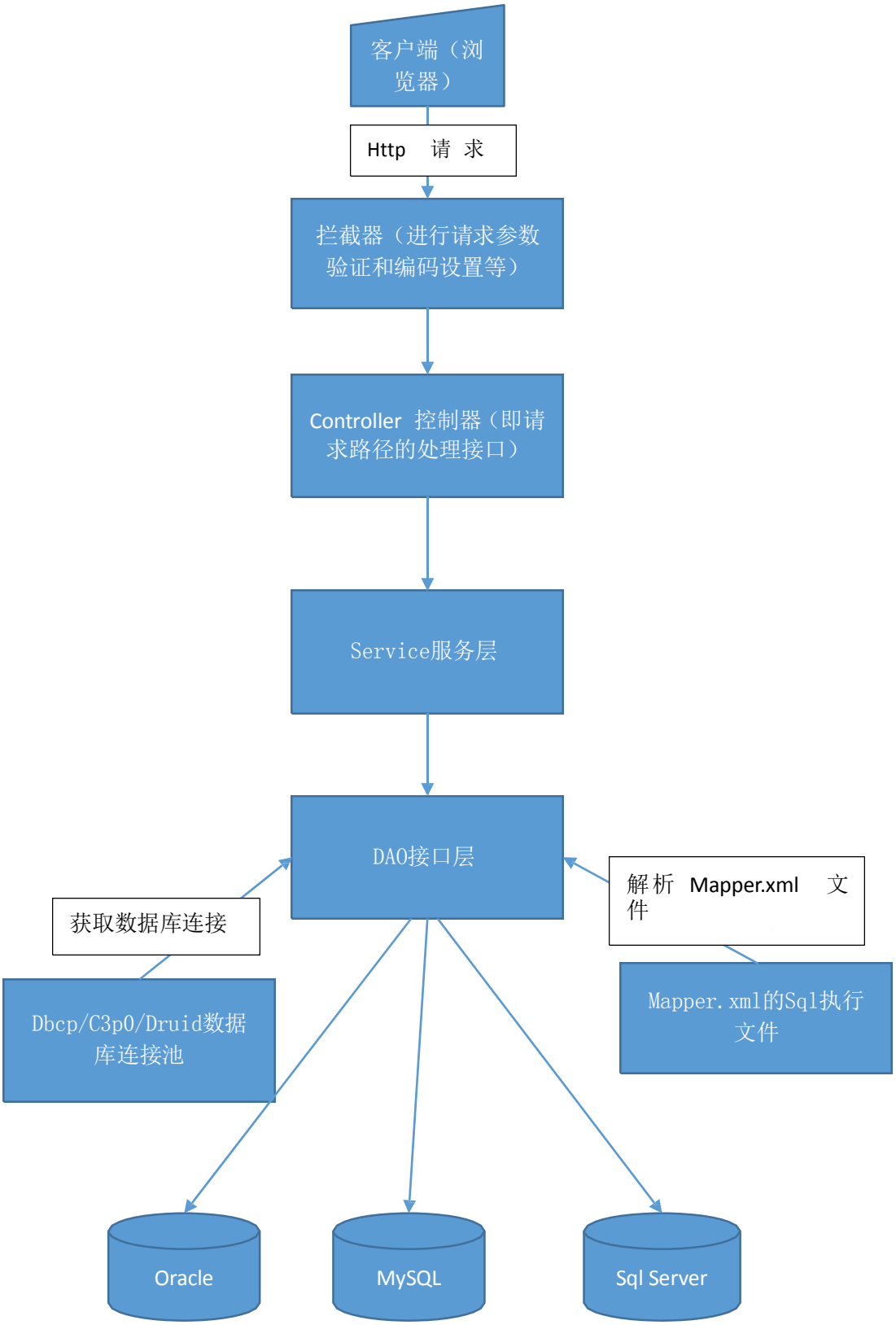
本项目为 shiro 权限控制，能对接后台接口，与后台权限保持一致，具体的 shiro 控制代码如下，主要实现前端按钮权限控制。


perms.js - vue-zrshiro - Visual Studio Code
// **********登录用户权限控制********** */
Vue.prototype.perms = function (but_value) {
let rm = false
//获取登录用户的按钮权限
const permsList = JSON.parse(sessionStorage.getItem('permslist'))
const loginid = JSON.parse(sessionStorage.getItem('user')).loginid
if (loginid == 'designer') { // 管理员
rm = true
} else {
if (permsList.length > 0 && but_value.length > 0) {
$.each(permsList, function (i, perms) {
if (perms.indexOf(but_value) > -1) {
rm = true
return false
}
})
} else {
rm = false
}
}
return rm
}
// **********系统单位json********** */
Vue.prototype.getUnitJson = function (unitId, callback) {
$.getJSON(this.HOME + '/sysunit/childlist', {
获取登录成功后后台返回的权限json，通过perms进行解析。

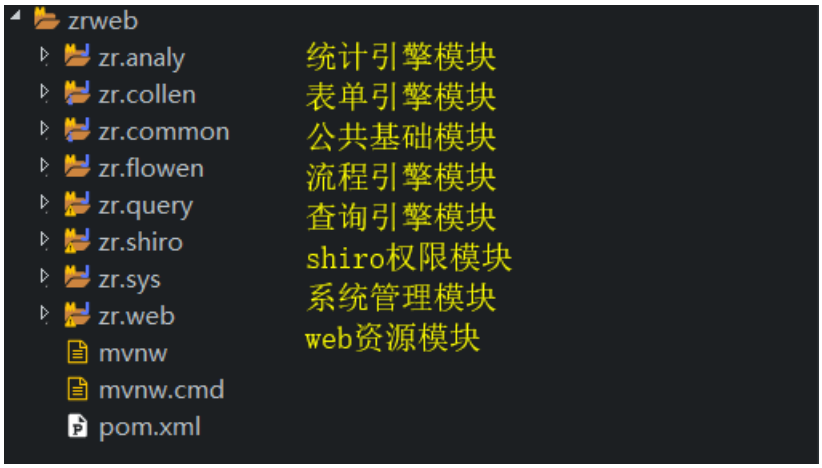

menu.vue - vue-zrshiro - Visual Studio Code
前端vue视图，通过this.perms(权限标识)进行验证
不存在，则不显示该按钮，就算显示了，后台调用
也无法成功。
v-if="this.perms('bpipmenu:list')"
v-if="this.perms('bpipmenu:save')"
v-if="this.perms('bpipmenu:info')"
v-if="this.perms('bpipmenu:delete')"

第五章　框架配置及说明

5.1　项目的软件架构图

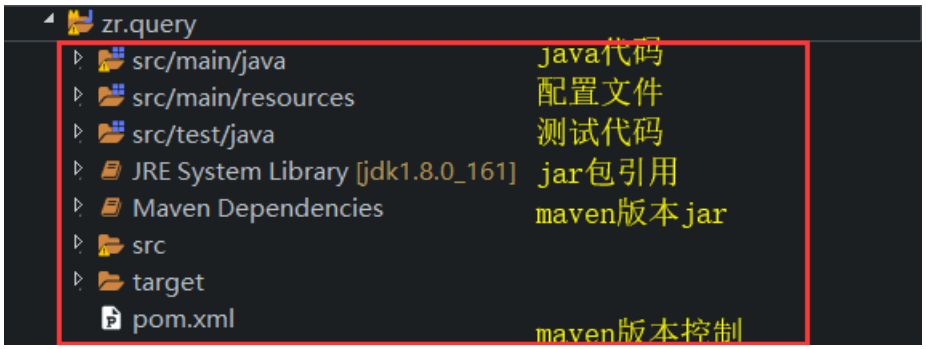


5.2 项目结构说明



项目框架包含统计、流程、表单、查询等四大引擎及公共基础、数据库管理、系统管理等 maven 模块目录，以及 web 静态资源。

基于查询、流程、表单、统计四大引擎项目框架包含以下：



src/main/java ： 包 含 了 controller 、 mapper 、 model\entiy 、 service\serviceImp 等控制、映射、模型\实体、接口等标准目录。

src/main/resources：包含了 mapper、template、lib 等目录，分别存放 mybatis 数据库映射文件 xml、静态文件、jar 包等。

src/test/java：maven 模块下的测试代码。

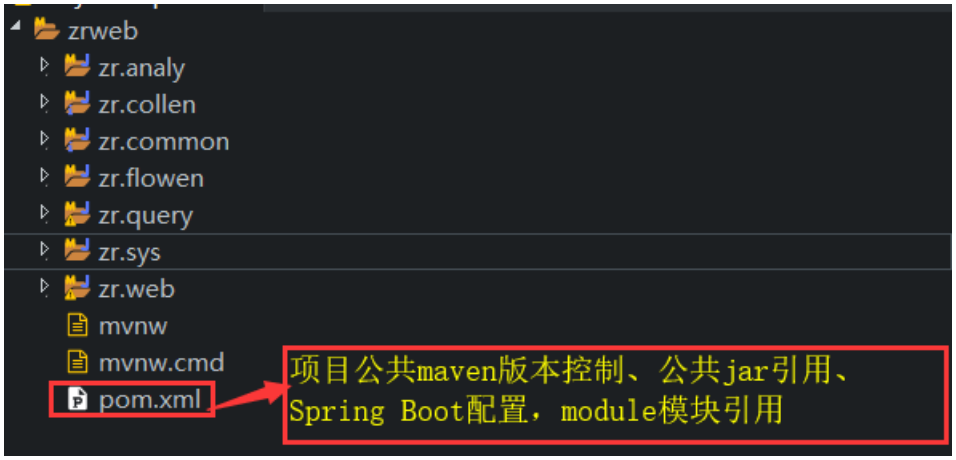
pom.xml：项目下的子模块（module）的 maven 版本控制，相对独立。

5.3 项目的核心技术说明

1、项目的 Maven 配置，pom.xml 文件中，本项目采用 maven 模块开发，用多

个 pom.xml 文件，主要分为 maven project、maven module，主要解释如下：

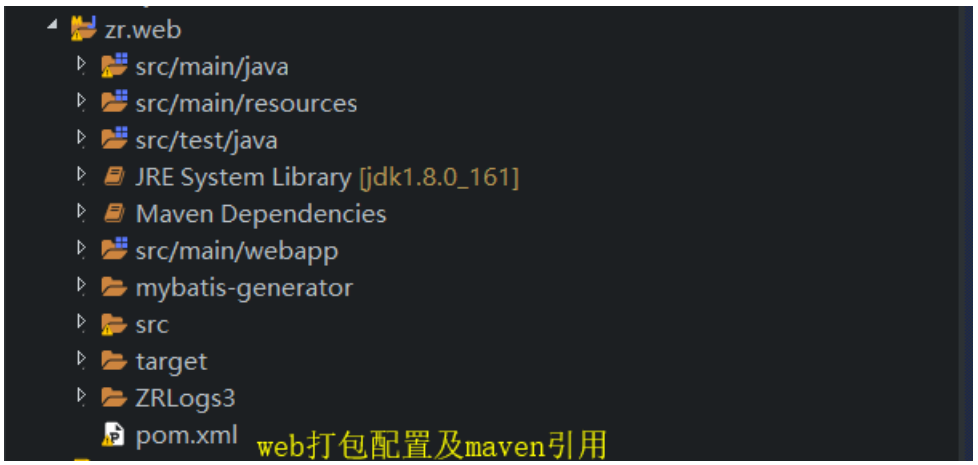
（1）主项目 maven 的 pom.xml：




```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
    <modelVersion>4.0.0</modelVersion>
    <groupId>zr.zrpower</groupId>
    <artifactId>ZRSpringBootThymfEsyui</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>pom</packaging>
    <name>ZRSpringBootThymfEsyui</name>

    <!--maven module WEB打包 -->
    <modules>
        <module>zr.web</module>
        <module>zr.common</module>
        <module>zr.query</module>
        <module>zr.sys</module>
        <module>zr.collen</module>
        <module>zr.analy</module>
        <module>zr.flowen</module>
        <module>zr.dbEngine</module>
</modules>
```

加载maven项目所有子模块，引用到主maven项目

（2）入口 maven 的 pom.xml：



```xml
<dependencies>
    <!-- 加载模块 -->
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.dbEngine</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.sys</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.query</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.common</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.collen</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.analy</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.flowen</artifactId>
        <version>${project.version}</version>
```

加载web项目需要引用的子模块

（3）其它 maven 的 pom.xml：

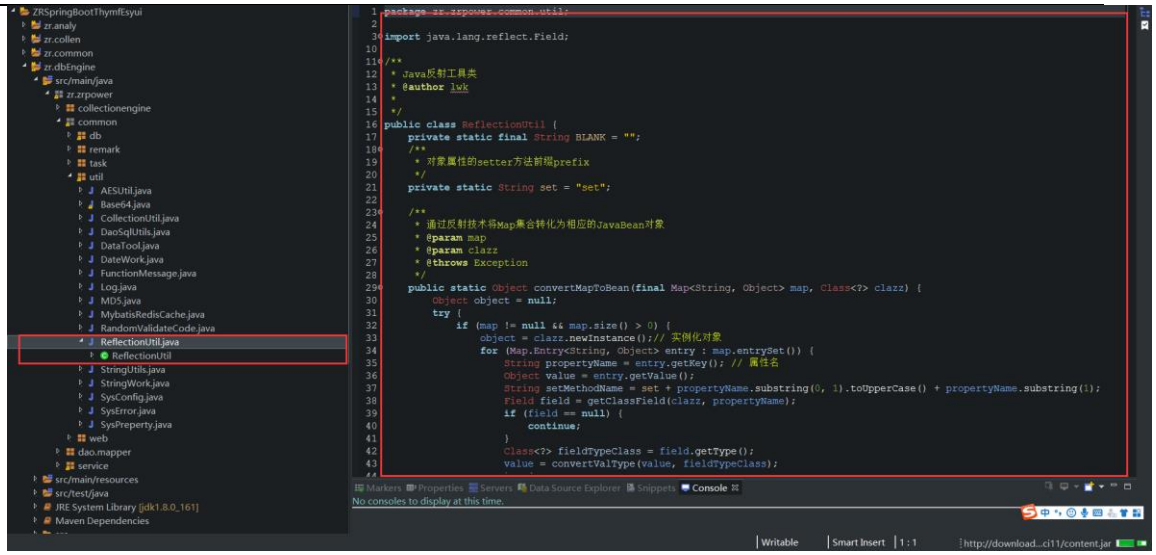
```xml
<parent>
    <groupId>zr.zrpower</groupId>
    <artifactId>ZRSpringBootThymfEsyui</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</parent>

<artifactId>zr.query</artifactId>
<packaging>jar</packaging>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
    <!-- 引用工具的模块 -->
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.common</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>zr.zrpower</groupId>
        <artifactId>zr.sys</artifactId>
        <version>${project.version}</version>
    </dependency>

</dependencies>
```

定义父亲maven项目的名称、版本

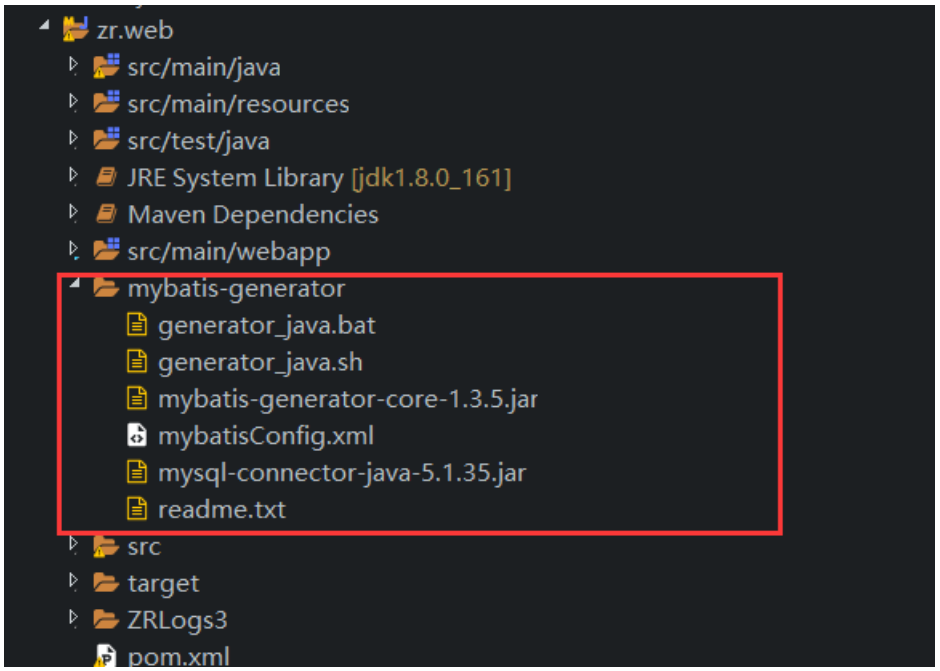
定义本maven模块的标识符、打包类型为jar

本maven模块是否引用框架内其它的maven项目及版本号。

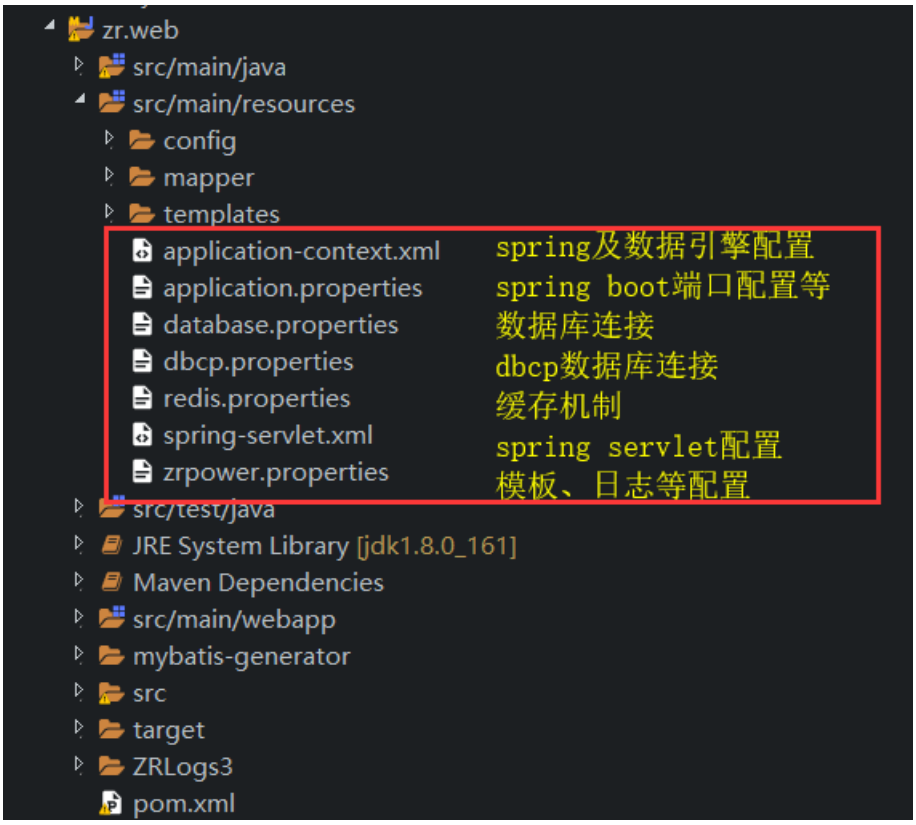
2、 项目中的 Java 反射工具类，用于解析 Map 和 JavaBean 之间的转换，如图：



3、 Mybatis 动态生成代码的配置，如图



4、 项目中的数据库连接池配置，如图：



druid 数据库连接池：

```
    <!-- 数据源 : druid数据库连接池 -->
    <!-- 参数说明:
    filters: 配置监控统计拦截的filters
    initialSize: 初始化时建立物理连接的个数。初始化发生在显示调用init方法，或者第一次getConnection时
    maxActive: 最大连接池数量。Default: 8
    minIdle: 最小连接池数量
    maxWait: 获取连接时最大等待时间，单位毫秒
    timeBetweenEvictionRunsMillis: 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒
    minEvictableIdleTimeMillis: 配置一个连接在池中最小生存的时间，单位是毫秒
    poolPreparedStatements: 是否缓存preparedStatement，也就是PSCache。
                            PSCache对支持游标的数据库性能提升巨大，比如说oracle。在Mysql下建议关闭
    maxOpenPreparedStatements: 要启用PSCache，必须配置大于0，当大于0时，poolPreparedStatements自动触发修改为true
    -->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">
        <property name="driverClassName" value="${db.driver}" />
        <property name="url"             value="${db.url}" />
        <property name="username"        value="${db.username}" />
        <property name="password"        value="${db.password}" />
        <property name="filters"         value="stat" />
        <property name="initialSize"     value="${db.initialSize}" />
        <property name="maxActive"       value="${db.maxActive}" />
        <property name="minIdle"         value="${db.minIdle}" />
        <property name="maxWait"         value="${db.maxWait}" />
        <!-- 这里配置提交方式，默认就是TRUE，可以不用配置 -->
        <property name="defaultAutoCommit" value="true" />
        <property name="timeBetweenEvictionRunsMillis" value="${db.timeBetweenEvictionRunsMillis}" />
        <property name="minEvictableIdleTimeMillis"    value="${db.minEvictableIdleTimeMillis}" />
        <property name="testWhileIdle" value="true" />
        <property name="testOnBorrow"  value="false" />
        <property name="testOnReturn"  value="false" />
        <property name="poolPreparedStatements" value="true" />
        <property name="maxOpenPreparedStatements" value="50" />
    </bean>
```

c3p0 数据库连接池：

```
    <!-- 数据源 : c3p0数据库连接池 -->
    <!-- 参数说明，
    initialPoolSize: 初始化时获取的连接数，取值应在minPoolSize与maxPoolSize之间。Default: 3
    minPoolSize: 连接池中保留的最小连接数。Default: 3
    maxPoolSize: 连接池中保留的最大连接数。Default: 15
    maxIdleTime: 最大空闲时间,60秒内未使用则连接被丢弃。若为0则永不丢弃。Default: 0
    acquireIncrement: 当连接池中的连接耗尽的时候c3p0一次同时获取的连接数。Default: 3
    maxStatements: JDBC的标准参数，用以控制数据源内加载的PreparedStatements数量。但由于预缓存的statements
                   属于单个connection而不是整个连接池。 所以设置这个参数需要考虑到多方面的因素。
                   如果maxStatements与maxStatementsPerConnection均为0，则缓存被关闭。Default: 0
    idleConnectionTestPeriod: 每60秒检查所有连接池中的空闲连接。Default: 0
    acquireRetryAttempts: 定义在从数据库获取新连接失败后重复尝试的次数。Default: 30
    -->
    <!-- <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">
        <property name="driverClass" value="${db.driver}" />
        <property name="jdbcUrl"     value="${db.url}" />
        <property name="user"        value="${db.username}" />
        <property name="password"    value="${db.password}" />
        <property name="initialPoolSize" value="${db.initialSize}" />
        <property name="minPoolSize" value="${db.minIdle}" />
        <property name="maxPoolSize" value="${db.maxIdle}" />
        <property name="acquireIncrement" value="${db.acquireIncrement}" />
        <property name="maxStatements" value="${db.maxStatements}" />
        <property name="idleConnectionTestPeriod" value="${db.idleConnectionTestPeriod}" />
        <property name="acquireRetryAttempts" value="${db.acquireRetryAttempts}" />
    </bean> -->
```

dbcp 数据库连接池：

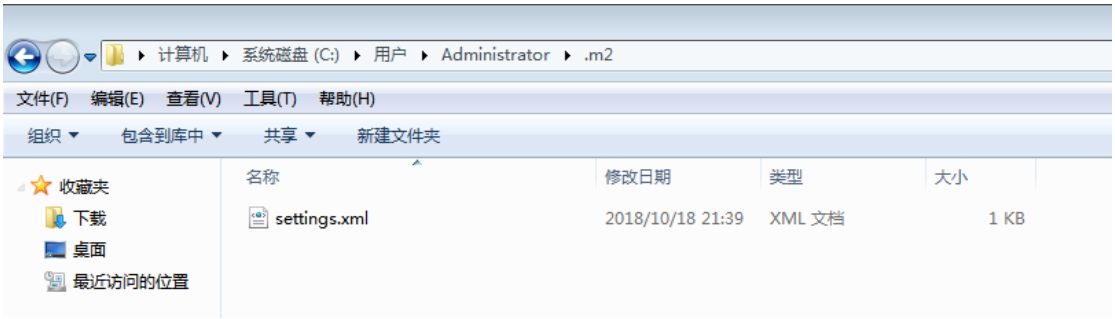
```
    <!-- 数据源 : dbcp数据库连接池 -->
    <!-- 参数说明，
    initialSize: 连接初始值，连接池启动时创建的连接数量的初始值，默认值是0
    minIdle: 最小空闲值，当空闲的连接数少于阀值时，连接池就会预申请去一些连接，以免洪峰来时来不及申请，默认值是0
    maxIdle: 最大空闲值，当经过一个高峰时间后，连接池将已经用不到的连接释放一部分，一直减少到maxIdle为止，0时无限制，默认值是8
    defaultAutoCommit: 默认的SQL语句自动提交状态(开启或关闭)设置由连接池本身设置(false由连接池定)，
                       不设置该值setAutoCommit方法不被调用
    -->
    <!-- <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="${db.driver}" />
        <property name="url"         value="${db.url}" />
        <property name="username"    value="${db.username}" />
        <property name="password"    value="${db.password}" />
        <property name="initialSize" value="${db.initialSize}" />
        <property name="minIdle" value="${db.minIdle}" />
        <property name="maxIdle" value="${db.maxIdle}" />
        <property name="defaultAutoCommit" value="${db.defaultAutoCommit}" />
    </bean> -->
```

5、项目中 Java 使用 dom4j 动态生成 mybatisConfig.xml 文件。

6、项目中 Java 动态生成 generator_java.bat 脚本文件及调用。

5.4 Eclipse 开发工具的 Maven 配置

在 C:\Users\Administrator\目录下创建.m2 文件夹，在.m2 文件夹下创建 settings.xml 的文件，如图所示：

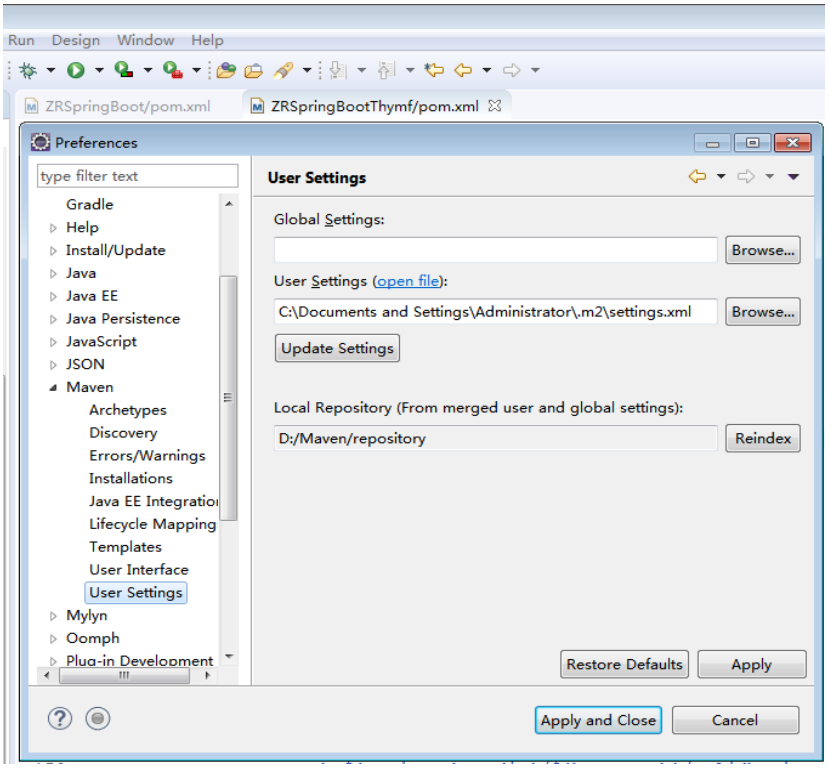


在 settings.xml 的文件中配置本地 Maven 仓库地址及阿里云代理，文件内容如下：

具体代码如下：

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
    <!-- 指定Maven本地仓库的位置 -->
    <localRepository>D:/repos</localRepository>
    <!--
    <localRepository>${user.home}/.m2/repository</localRepository>
     -->
    <interactiveMode>true</interactiveMode>
    <usePluginRegistry>false</usePluginRegistry>
    <offline>false</offline>

    <mirrors>
    <mirror>
        <id>nexus-aliyun</id>
            <mirrorOf>*</mirrorOf>
            <name>Nexus aliyun</name>
            <url>http://maven.aliyun.com/nexus/content/groups/public</url>
    </mirror>
    </mirrors>
</settings>
```

打开 Eclipse 开发工具，点击菜单 Window→Preferences→Maven→User Segttings,配置 User Settings(浏览找到 settings.xml 文件),点击 Update Settings 按钮，如图所示：



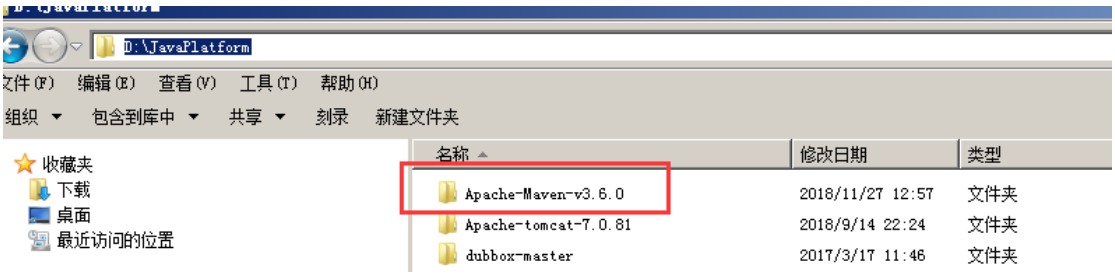
点击 Reindex 按钮，然后点击 Apply and Close 按钮即可 Eclipse 开发工具的 Maven 配置。

5.5 打包部署操作说明

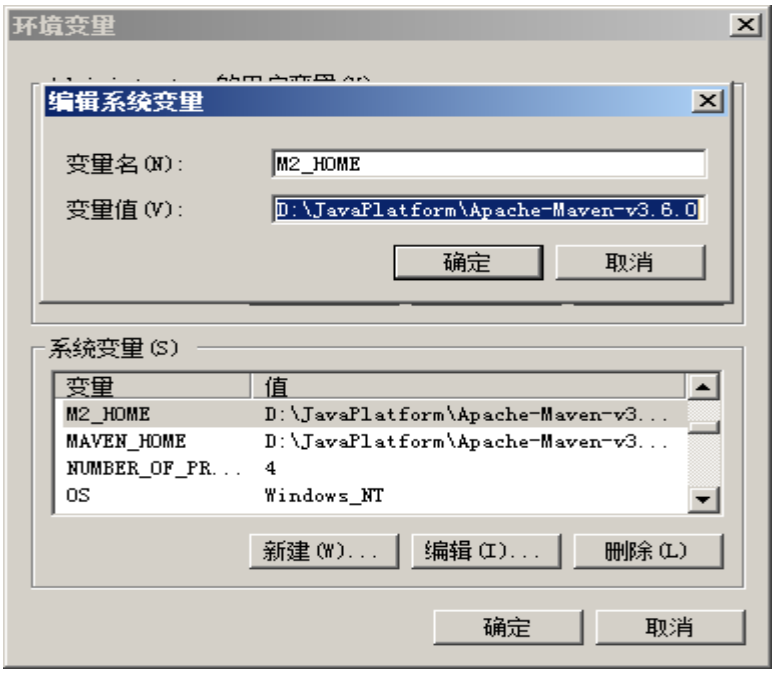
一、在本地上安装 Maven 环境

1. Maven 下载地址 http://maven.apache.org/download.cgi

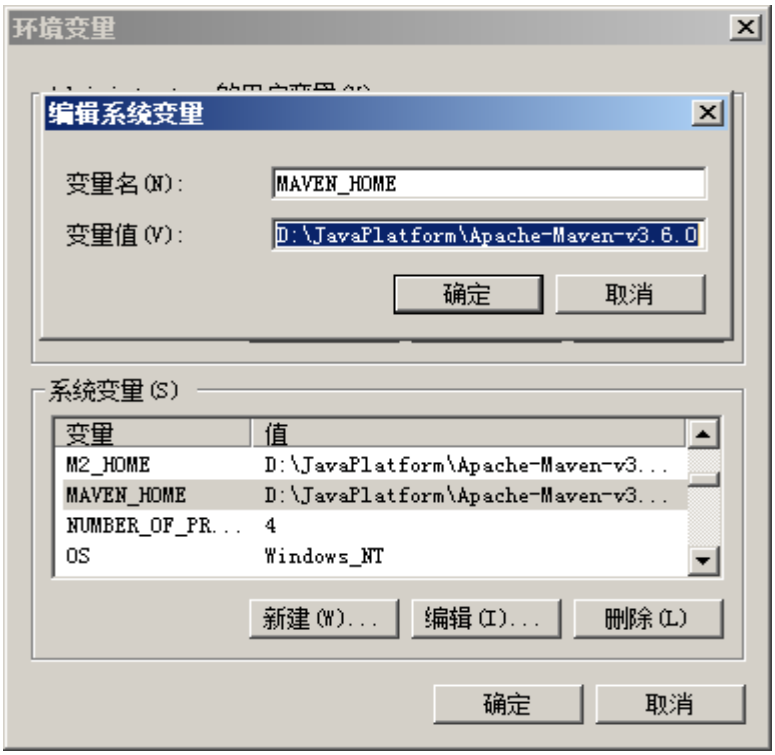
2. 将下载的文件（apache-maven-3.6.0-bin.zip）放在 D:\JavaPlatform 目录下并解压，如图



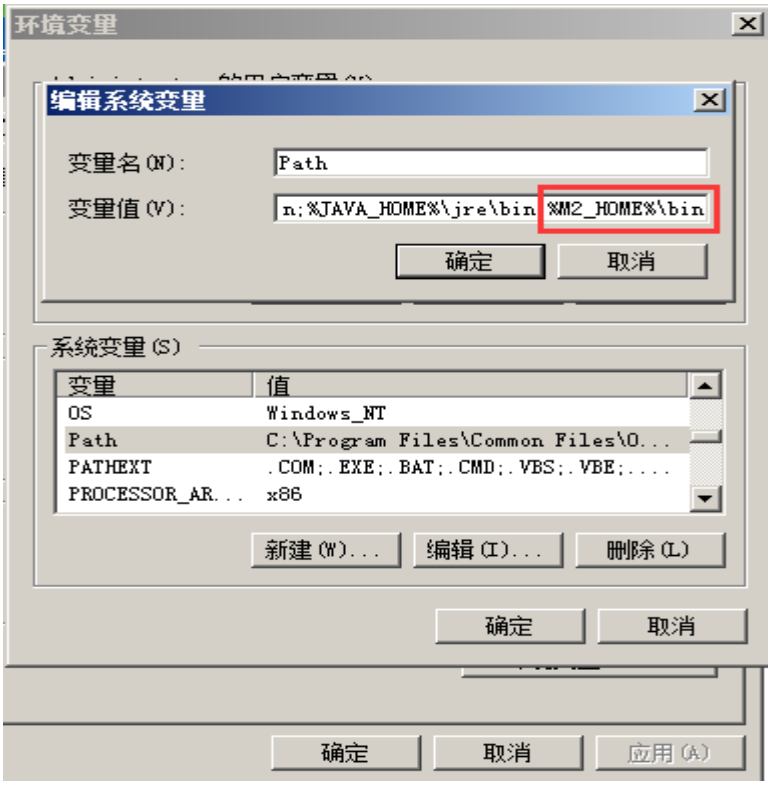
3. 配置 Maven 环境：



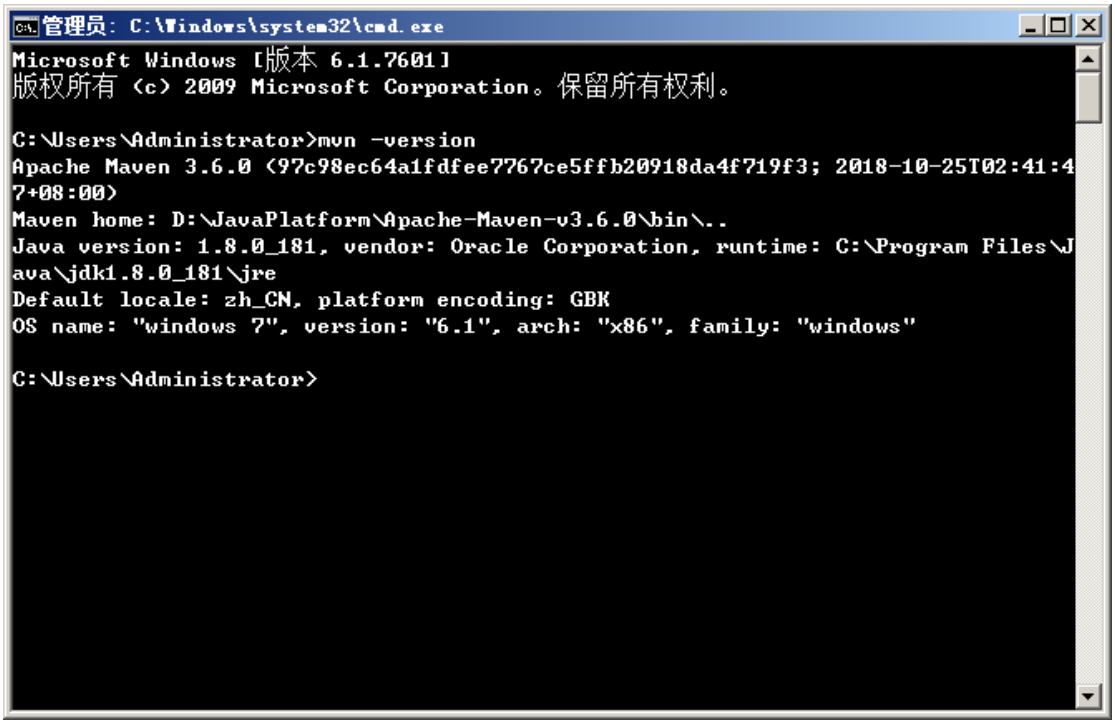
新增 M2_HOME 环境变量



新增 MAVEN_HOME 环境变量



在 Path 环境变量中新增 Maven 配置，点击确定即可。%M2_HOME%\bin

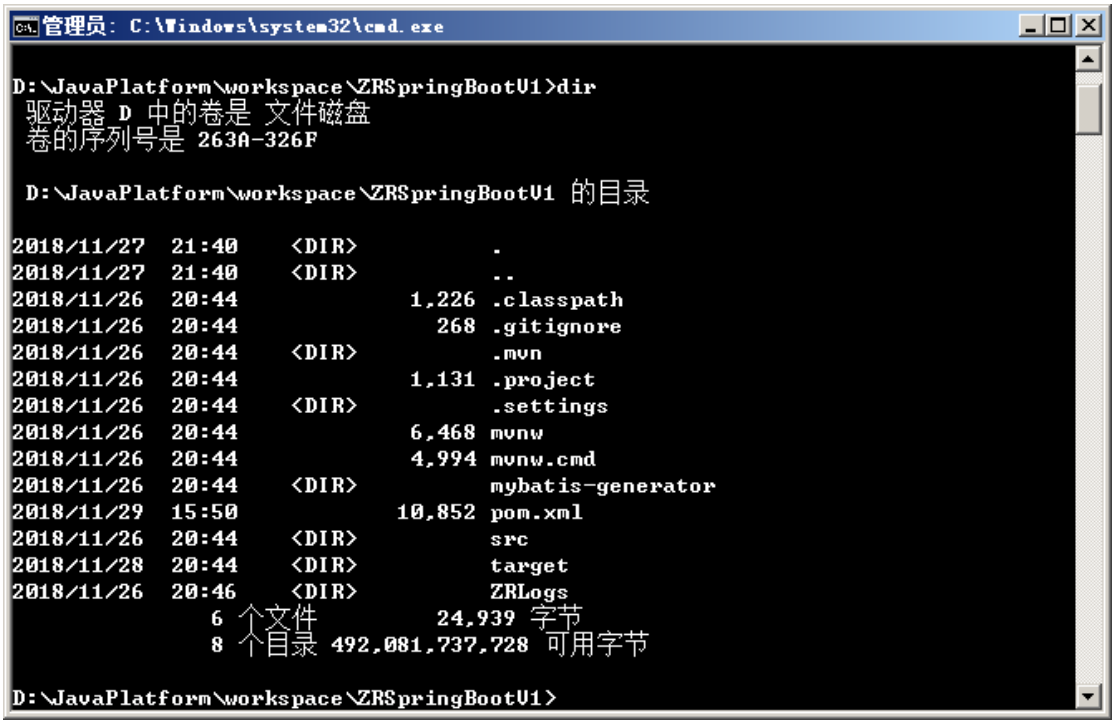


查看 maven 是否安装成功，输入 mvn -version，出现上图说明安装配置完成。

二、将 ZRSpringBoot 项目中本地引用的 jar 包发布到本地 Maven 仓库

说明：理论上可以在 ZRSpringBoot 的 pom.xml 文件中设置将本地引用 jar 包打包到项目 jar 包 zrweb-0.0.1-SNAPSHOT.jar 中，但是实际上会有很多问题出现，不建议使用。

进入 ZRSpringBoot 项目所在的目录，如图：



将所有的本地引用 jar 包发布到本地 Maven 仓库中去，命令如下：

如将 xssProtect-0.1.jar 发布到本地 Maven 仓库中去，输入命令执行

```
mvn install:install-file -Dfile=D:\JavaPlatform\TempJarLib\xssProtect-0.1.jar
-DgroupId=com.zrpower -DartifactId=xssProtect -Dversion=0.1 -Dpackaging=jar
```

```
<dependency>
    <groupId>com.zrpower</groupId>
    <artifactId>xssProtect</artifactId>
    <version>0.1</version>
    <!--
```

命令参数说明：

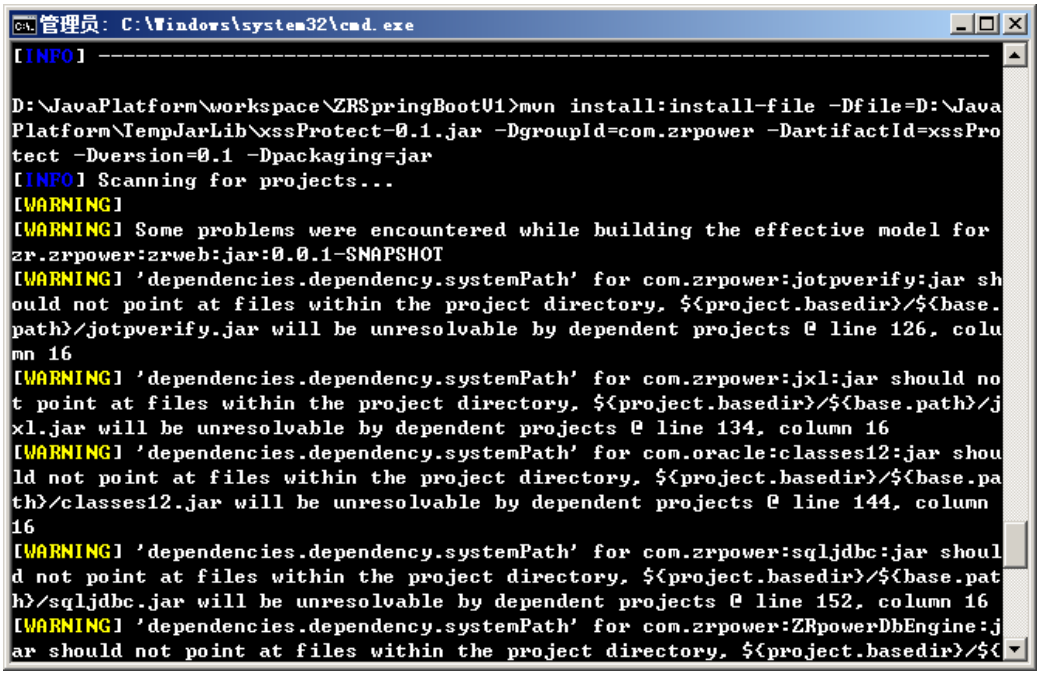
-Dfile 为 jar 文件所在目录

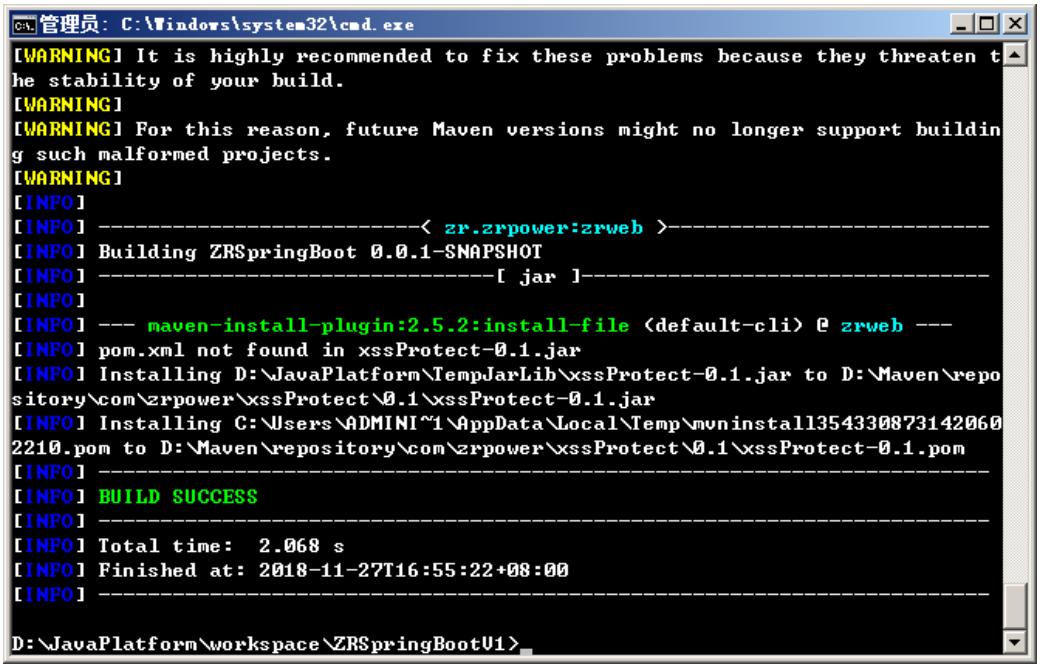
-Dgroup=com.zrpower 为 jar 包的组

-DartifactId=xssProtect 为 jar 包的 ID

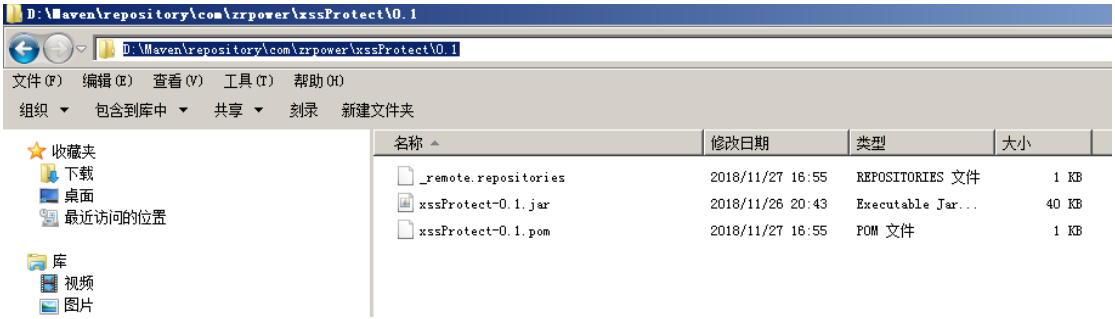
-Dversion=0.1 为 jar 包的版本号

-Dpackaging=jar 为发布类型为 jar 包





至此，发布 jar 包到本地 Maven 仓库成功，可以在本地 Maven 仓库中查看到发布的 jar 包，如下图



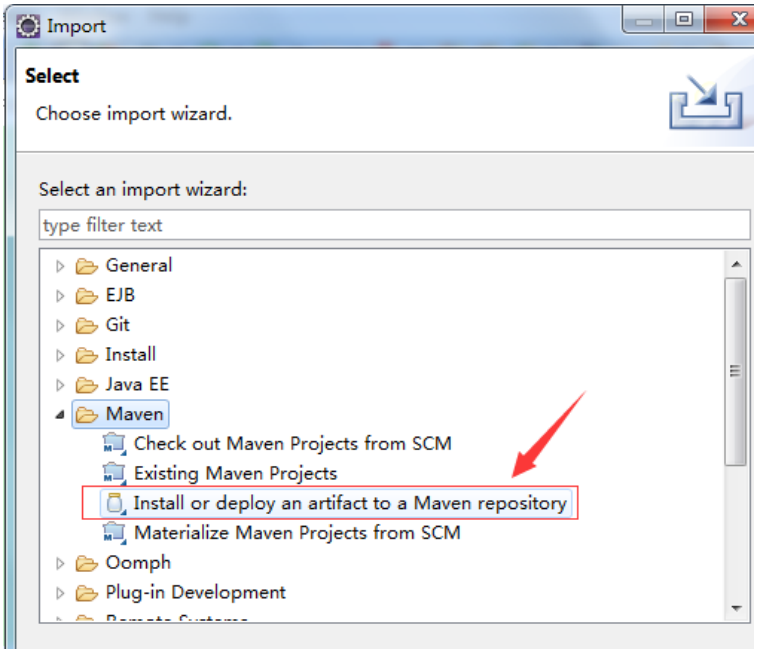
其他 jar 包的发布操作类似。

另一种方式：使用 eclipse 安装 jar 包：

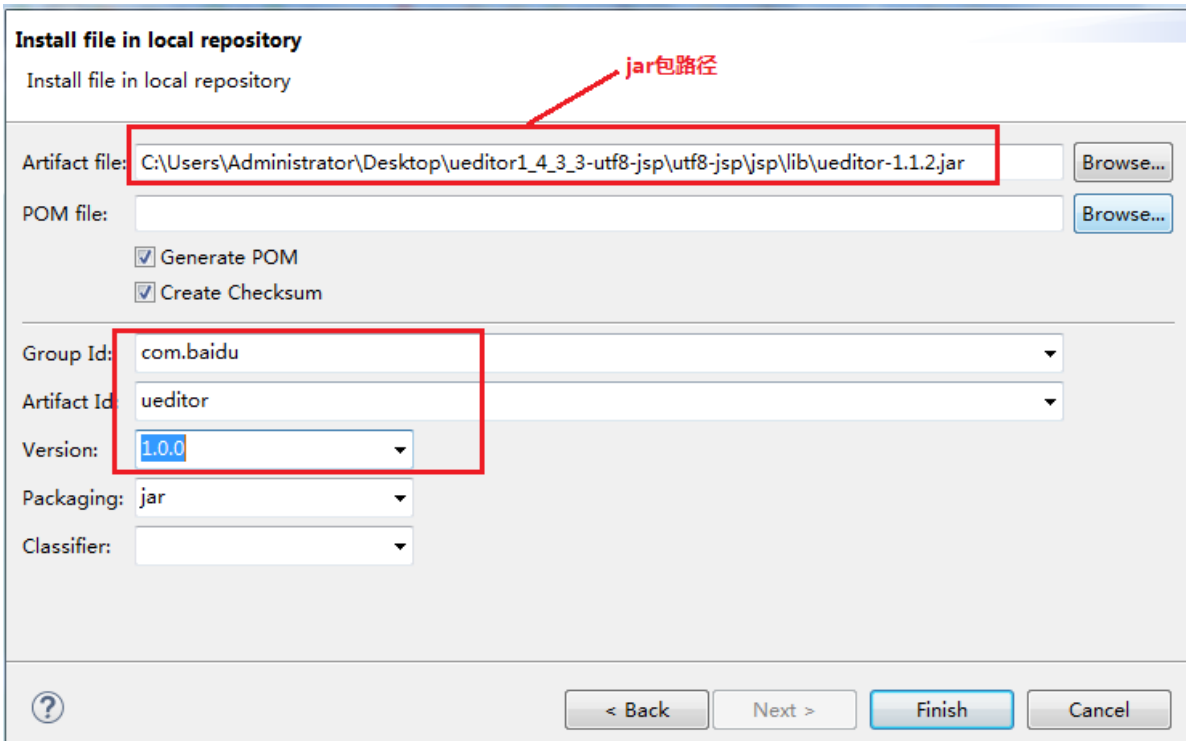
使用 eclipse 安装也有个前提，就是 eclipse 的 Maven 要先配置好。不过我相信使用 Maven 的小伙伴的 eclipse 的 Maven 设置肯定是没问题的，不然还怎么用 Maven。

具体操作：

（1）File -->import -->Maven -->instal or deploy an artifact to a Maven repository



（2）填写相关信息，如 Maven 坐标，具体参考下图。完成后点击 Finish。



Group Id,Artifact Id,Version 的设置参考 pom.xml 文件。

（3）完成上述步骤后，就已经大功告成了，可以去本地仓库看下上面安装的东西在不在。

三、将 ZRSpringBoot 项目 pom.xml 中所有本地引用的 jar 包改为正常引用

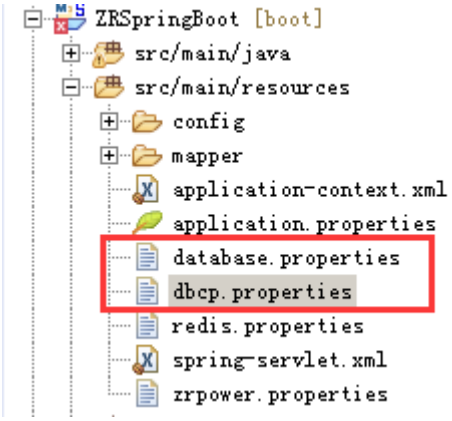
如图所示：

```
<dependency>
    <groupId>com.zrpower</groupId>
    <artifactId>jotpverify</artifactId>
    <version>1.0.0</version>
    <!--
    <scope>system</scope>
    本地jar的路径，相对或者绝对都可以
    <systemPath>${project.basedir}/${base.path}/jotpverify.jar</systemPath>
     -->
</dependency>
<dependency>
    <groupId>com.zrpower</groupId>
    <artifactId>jxl</artifactId>
    <version>1.0.0</version>
    <!--
    <scope>system</scope>
    本地jar的路径，相对或者绝对都可以
    <systemPath>${project.basedir}/${base.path}/jxl.jar</systemPath>
     -->
</dependency>
```

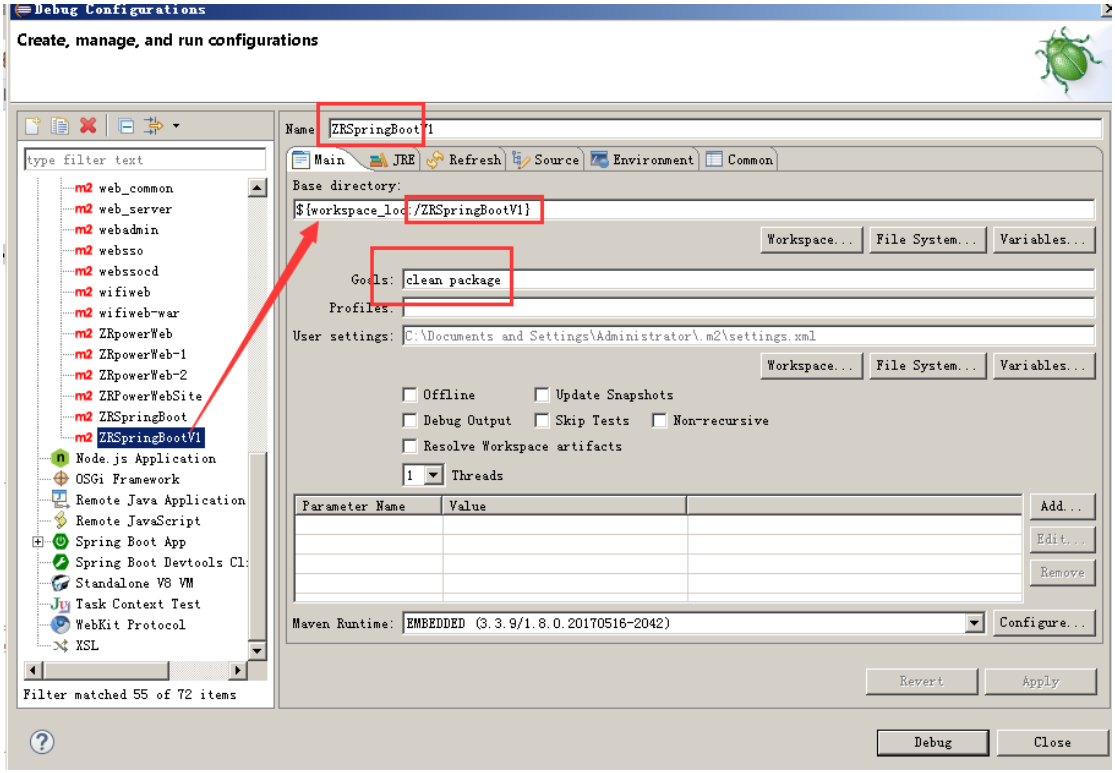
```
<!-- 运行时需要加入的第三方jar -->
<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>classes12</artifactId>
    <version>1.0.0</version>
    <!--
    <scope>system</scope>
    本地jar的路径，相对或者绝对都可以
    <systemPath>${project.basedir}/${base.path}/classes12.jar</systemPath>
     -->
</dependency>
<dependency>
    <groupId>com.zrpower</groupId>
    <artifactId>sqljdbc</artifactId>
    <version>1.0.0</version>
    <!--
    <scope>system</scope>
    本地jar的路径，相对或者绝对都可以
    <systemPath>${project.basedir}/${base.path}/sqljdbc.jar</systemPath>
     -->
</dependency>
```

```
<dependency>
    <groupId>com.zrpower</groupId>
    <artifactId>ZRpowerDbEngine</artifactId>
    <version>1.0.0</version>
    <!--
    <scope>system</scope>
    本地jar的路径，相对或者绝对都可以
    <systemPath>${project.basedir}/${base.path}/ZRpowerDbEngine.jar</systemPath>
     -->
</dependency>

<dependency>
    <groupId>com.zrpower</groupId>
    <artifactId>antlr4-runtime</artifactId>
    <version>4.1</version>
    <!--
    <scope>system</scope>
    本地jar的路径，相对或者绝对都可以
    <systemPath>${project.basedir}/${base.path}/antlr4-runtime-4.1.jar</systemPath>
     -->
</dependency>
<dependency>
    <groupId>com.zrpower</groupId>
    <artifactId>xssProtect</artifactId>
    <version>0.1</version>
    <!--
    <scope>system</scope>
    本地jar的路径，相对或者绝对都可以
    <systemPath>${project.basedir}/${base.path}/xssProtect-0.1.jar</systemPath>
     -->
</dependency>
```

四、将 ZRSpringBoot 项目的数据库配置改为测试环境的配置参数

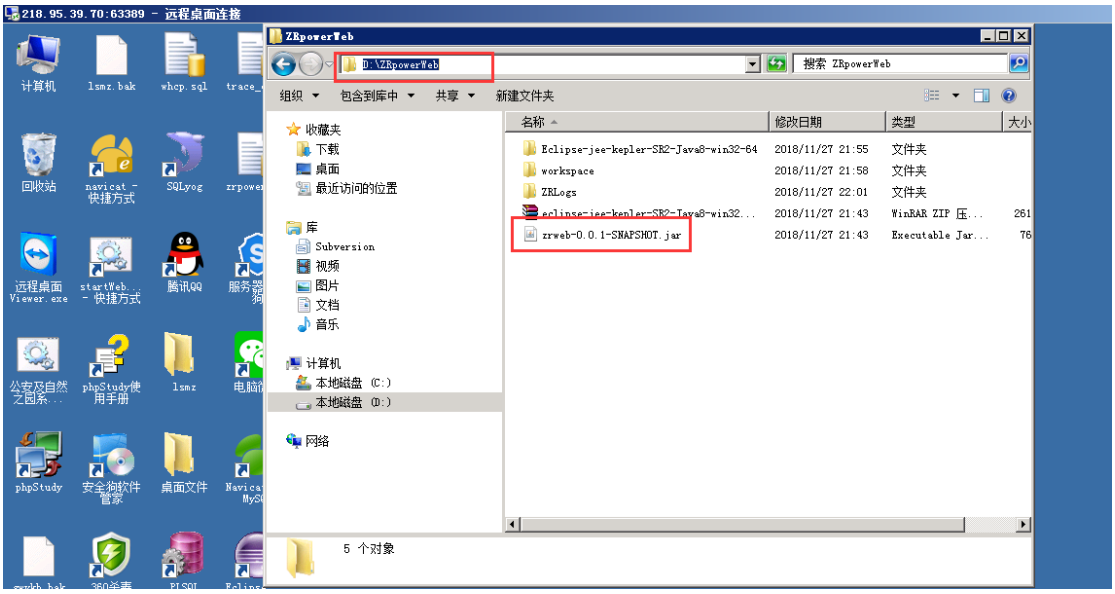


五、使用 clean package 命令将项目打成 jar 包，如图：

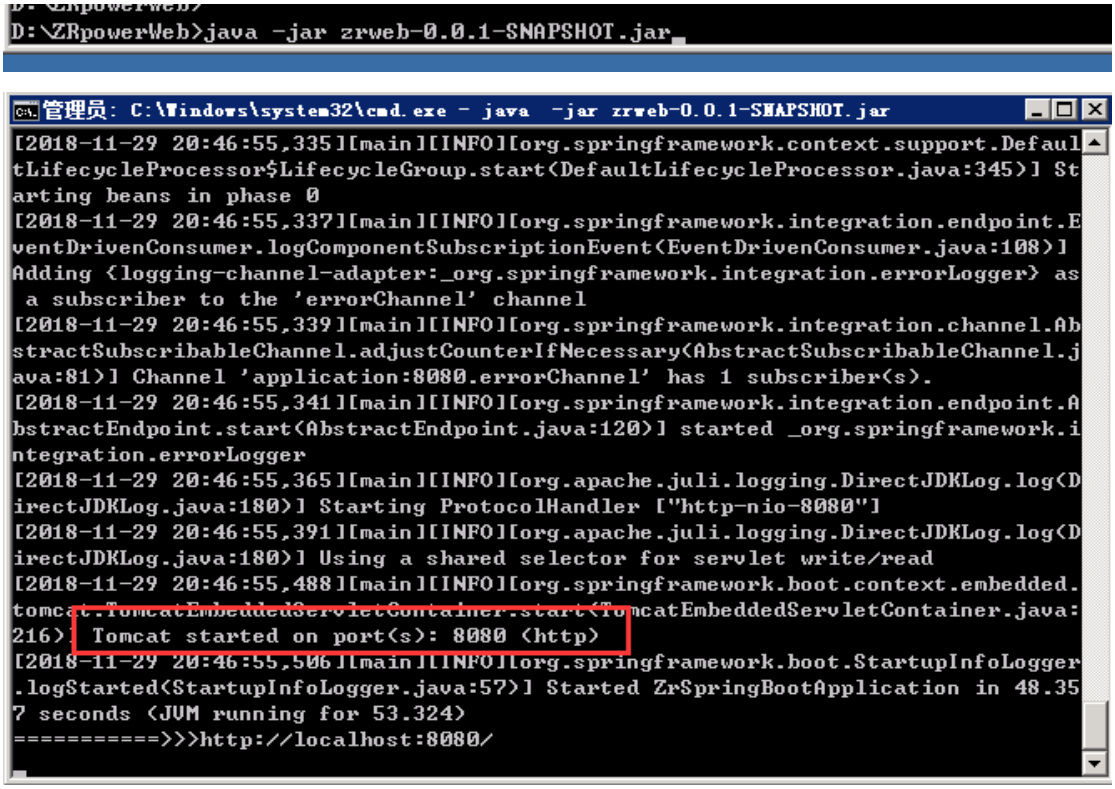


点击 Debug 即可运行打包操作。

六、将项目的 jar 包拷贝到服务器 D:\ZRpowerWeb 的目录下，如图：



七、cmd 进入到 D:\ZRpowerWeb 的目录，执行命令：java -jar zrweb-0.0.1-SNAPSHOT.jar，如图：



至此，项目在服务器上的部署完成。

5.6 只集成引擎时说明

不使用公司的系统管理部分，只使用其它引擎时集成说明(目的是对集成到项目中的各种引擎赋 session 值，不然不能正常使用各种引擎)：

引入 Session Bean:

```
import zr.zrpower.entity.sys.SessionUser;
```

Session 赋值的方法如下：

```
SessionUser su = new SessionUser();
su.setUserID("0000000000000001");//用户 ID
su.setLCODE("000000");//用户编号
su.setName("张三");//用户名称
su.setUnitID("001");//单位(部门)编号
su.setUnitName("办公室");//单位(部门)名称
su.setUserPageSize(28);//表格每页显示的记录数
```

```
su.setCustom1("");//自定义 1
su.setCustom2("");//自定义 2
su.setCustom3("");//自定义 3
su.setCustom4("");//自定义 4
su.setCustom5("");//自定义 5
session.setAttribute("userinfo", su);
```

基础数据同步。

使用各种引擎需要将原开发框架中的一些数据同步到指定的表中。 数据表结构请参照开发框架部分的表结构。

需要同步的表包括：

单位[部门]表(BPIP_UNIT)

用户表(BPIP_USER) 角色表(BPIP_ROLE)

用户角色表(BPIP_USER_ROLE)

注：单位[部门]ID、用户 ID、角色 ID 规范请参照示例数据生成，单位 ID 是 12 位、用户 ID 是 16 位(前 12 位为单位编号，后 4 位为流水号)，角色 ID 为数字类型编号,不按编码规范生成引擎的有些功能不能正常使用。

由于各种 ID 是按新的规范生成，所以需在集成项目的相关表中存放生成的编号，以便更新数据。

上面赋值 session 值时用的单位[部门]编号、用户 ID 要用新生成的编号赋值。

5.7 创建数据表及演示数据

通过[开发文档及数据库建表 sql / 数据库建表语句及示例数据]下的 sql 语句建表或插入演示数据，演示效果与公司网站 www.zrpower.cn 上的在线示例一样。

提供三种数据库（oracle、mssql、mysql）的 sql 语句，用户根据所使用的数据库选用相关的 sql 语句执行。

5.8 Mysql 数据库乱码处理

如果软件执行保存到数据库中的中文出现乱码，按以下方法设置 mysql。

检查本地 mysql 安装文件目录下的 my.ini 配置文件，服务器和客户端的默认编码方式设置成 utf8，设置的项目如下。

```
character-set-server=utf8
default-character-set=utf8
```

第六章　开发技术文档捐赠

免费和开源项目，项目的可持续发展离不开您的支持，请作者喝杯咖啡吧。

扫描下面的二维码捐赠 298 元，获取项目二次开发《智慧开源基础开发引擎（项目开发技术手册）(SpringBoot+Mybatis+Thymeleaf+Vue+ElementUI+Shiro) V7.0 版本》(共 98 页)。

说明：捐赠后请将使用人姓名、工作单位、手机号、qq 号、邮箱、捐赠凭证截图发到 672561350@qq.com,我们会在 24 小时内通过邮件发送最新完整的技术开发文档。

贵州智慧开源软件有限公司

2021 年 9 月