

Maven_day1

1.Maven的引言

maven是apache组织的开源的项目构建工具，
所谓的项目的构建是代码的编译, 打包(jar,war), 测试, 部署等一些列流程, 类似于早起的ant

2.为什么是Maven

通过原始的IDE, 已经可以完成项目的打包, 编译, 部署, 为什么还需要maven?








- 可以摆脱IED规范, 一键完成, 方便
- maven支持远程部署
- Maven 可以完成持续集成|持续发布 (CI continuous integration) (其中的一个角色)
- 构建依赖 (解决系统的依赖检查) 防置在用户计算机的本地库 默认是用户的 .m2, 节省用户本地空间的占用

3.第一个maven程序

1. apache的官网下载maven的官方包

2.	 apache-maven-3.1.1-bin.zip	2017/3/22 15:01	WinRAR ZIP 压缩...	6,150 KB
----	---	-----------------	------------------	----------

3. 解压maven的zip包

	bin	2017/3/22 15:03	文件夹	
	boot	2017/3/22 15:03	文件夹	
	conf	2015/4/22 7:55	文件夹	
	lib	2017/3/22 15:03	文件夹	
	LICENSE	2013/9/17 11:24	文件	15 KB
	NOTICE	2013/9/17 11:24	文件	1 KB
	README.txt	2013/9/17 11:19	文本文档	3 KB

4.

5. 配置maven的环境变量 要求java环境1.7+



6.

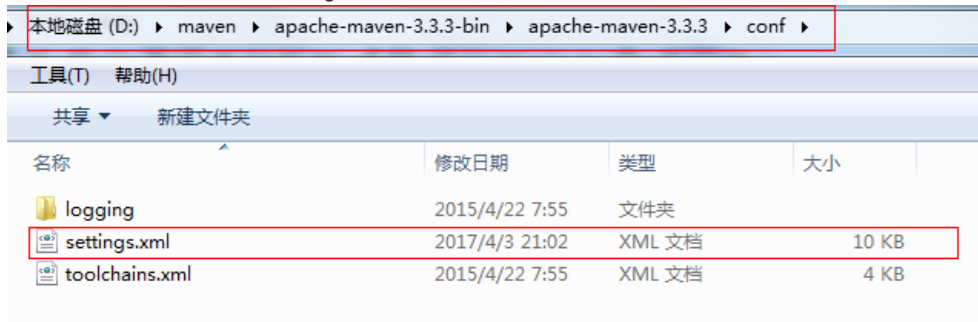


7.

8. CMD执行mvn -version查看环境配置

```
C:\Users\Administrator>mvn -version
Apache Maven 3.1.1 (0728685237757ffbf44136acec0402957f723d9a; 2013-09-17 23:22:22+0800)
Maven home: D:\maven\apache-maven-3.3.3-bin\apache-maven-3.3.3\bin\..
Java version: 1.8.0_111, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_111\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"
```

- 9.
10. 修改maven的核心配置文件 settings



- 11.
12. 修改本地库

```
settings.xml
46 <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
47         xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
48         xsi:schemaLocation="http://maven.apache.org/SE
49 <!-- localRepository
50 | The path to the local repository maven will use to
51 |
52 | Default: ${user.home}/.m2/repository
53 <localRepository>/path/to/local/repo</localRepository>
54 -->
55 <localRepository>D:/maven</localRepository>
```

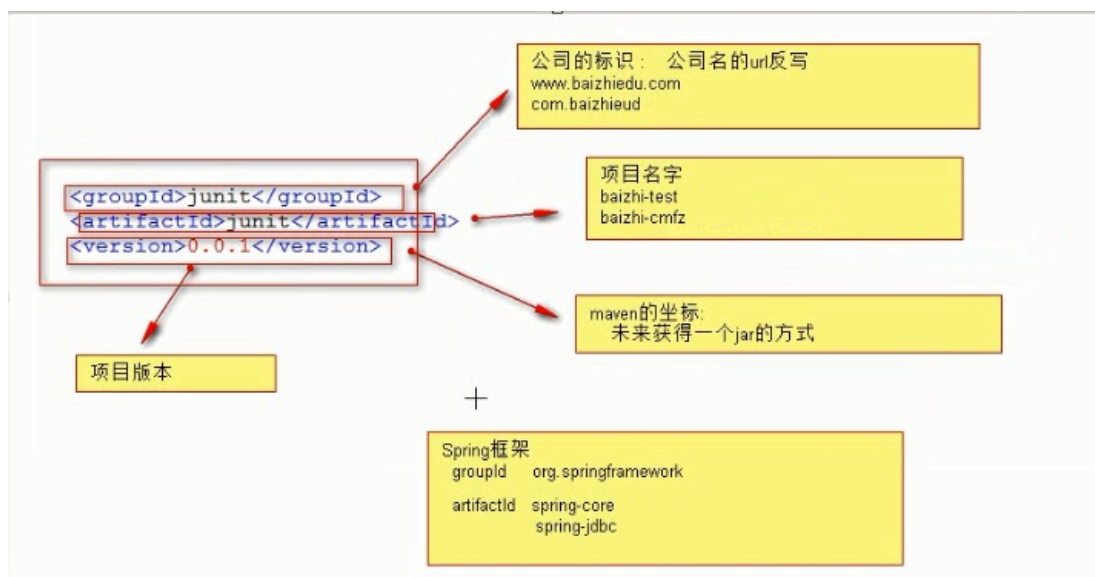
- 13.

4.Maven的约定

1. src/main/java 书写源代码
2. src/main/resources 书写配置文件
3. src/test/java 书写测试代码
4. src/test/resources 书写测试配置文件
5. 项目的根 pom.xml (project object model)

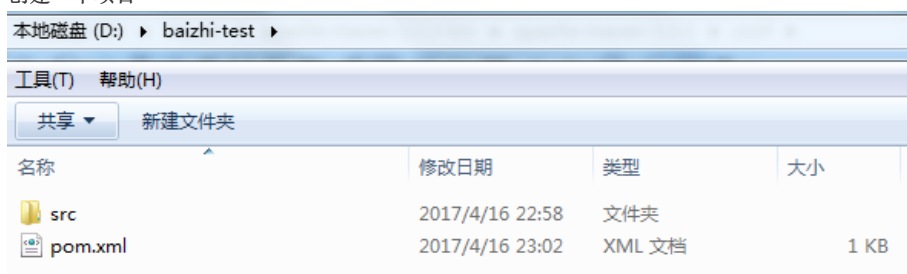
5.Maven项目的坐标

作用: 获取jar包的方式

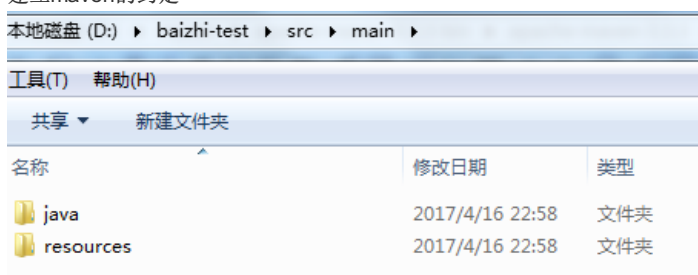


6.手动开发第一个maven项目

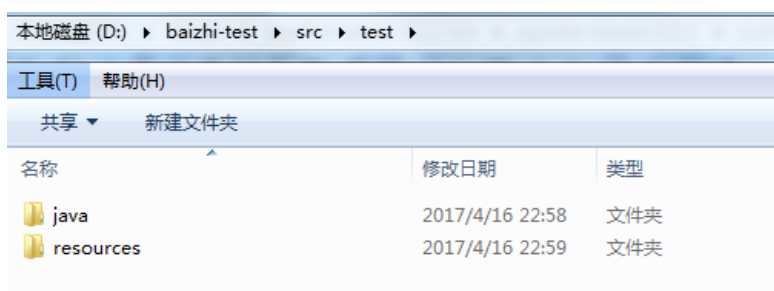
1. 创建一个项目



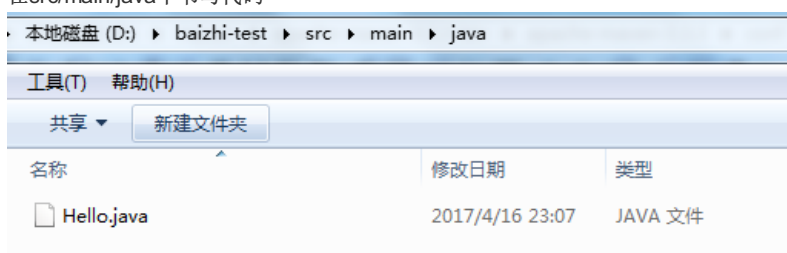
- 2.
3. 建立maven的约定



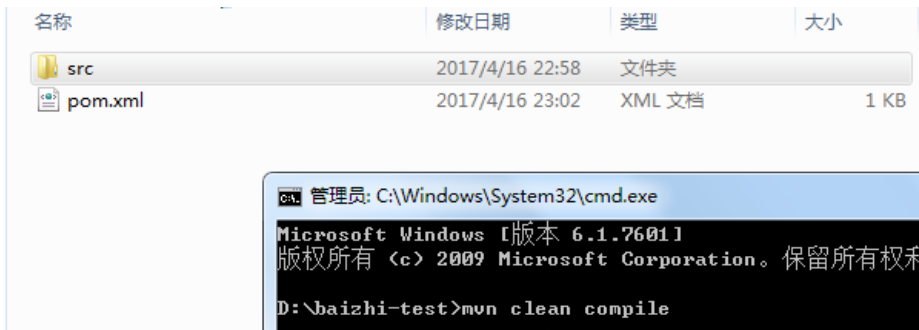
- 4.



- 5.
6. 在src/main/java中书写代码

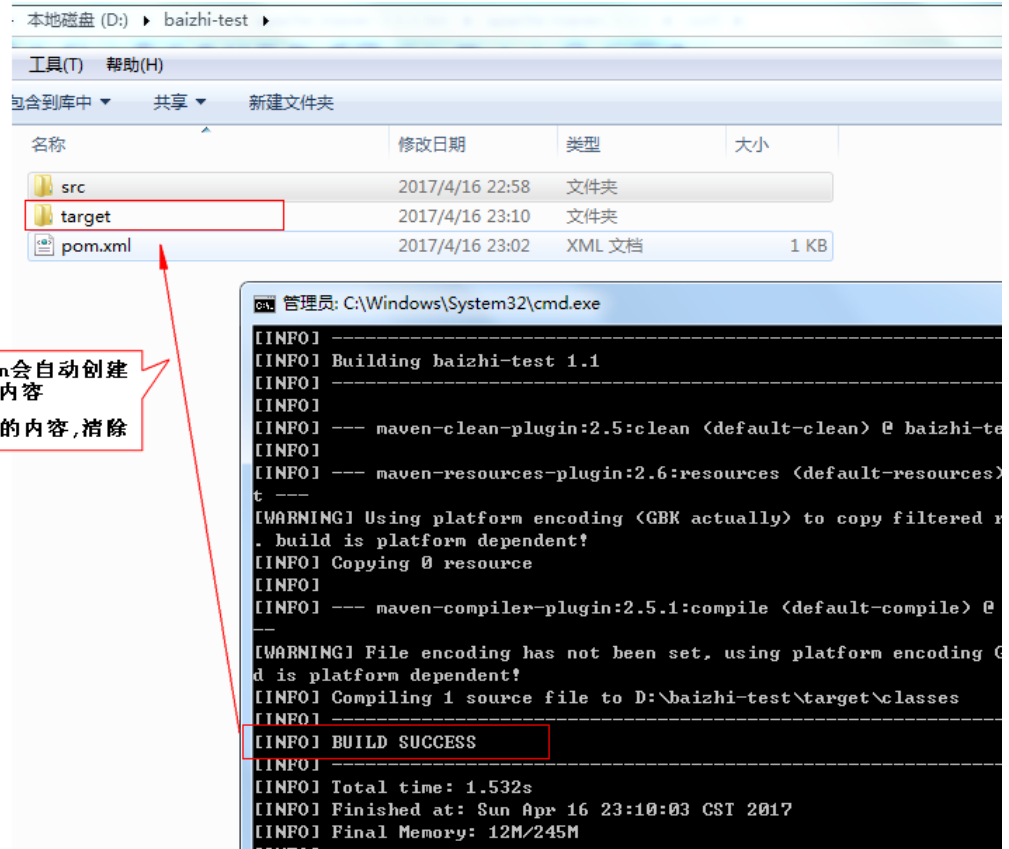


- 7.
8. 运行



Maven命令 plugin[命令]

9.

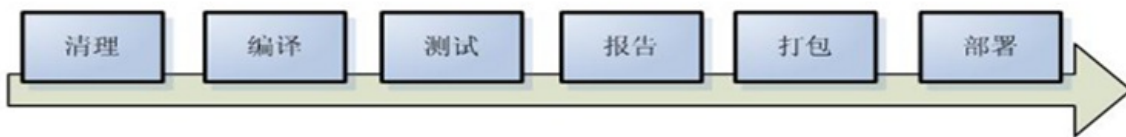


当一段代码运行成功后,maven会自动创建target文件夹,用来存放编译好的内容
clean的目的就是把上一次编译好的内容,消除

10.

7.maven的生命周期

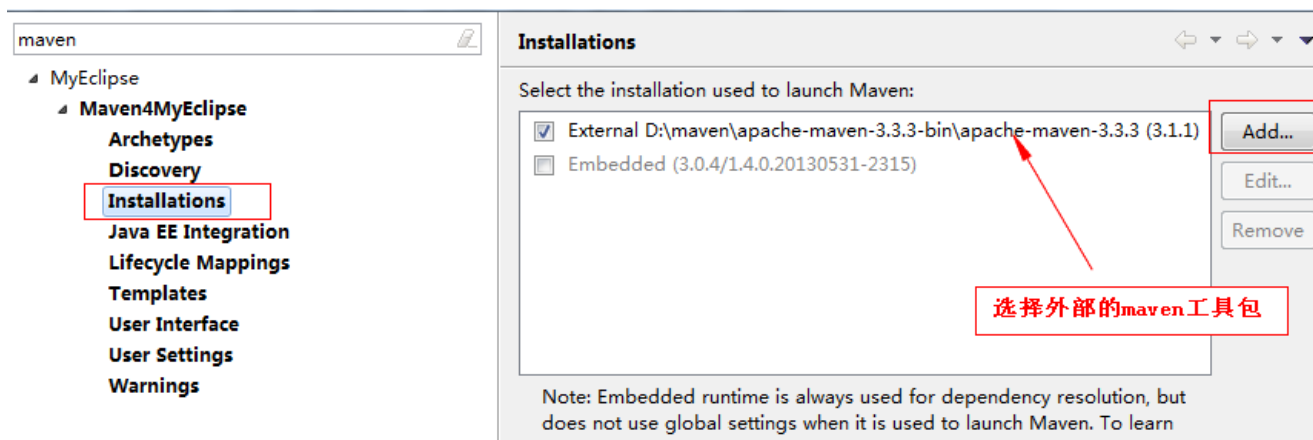
maven 的生命周期包含: 清理 编译 测试 报告 打包 部署



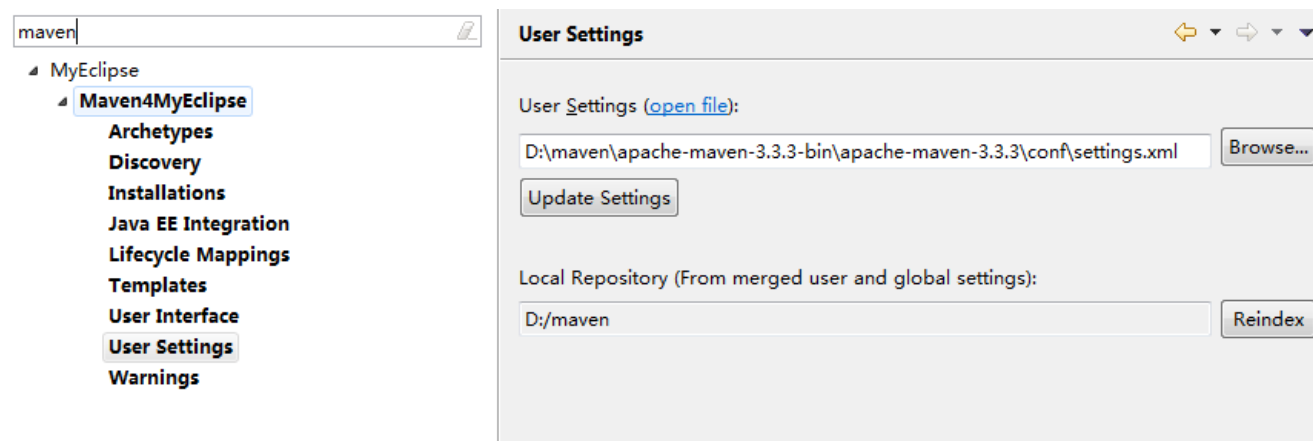
1.

8.maven与eclipse工具的集成

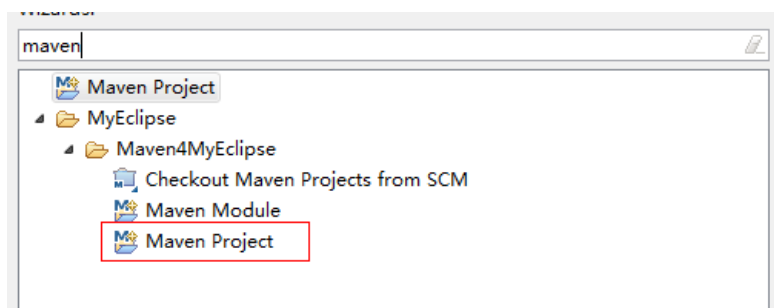
1. 引入外部的maven工具包替换eclipse中的工具包

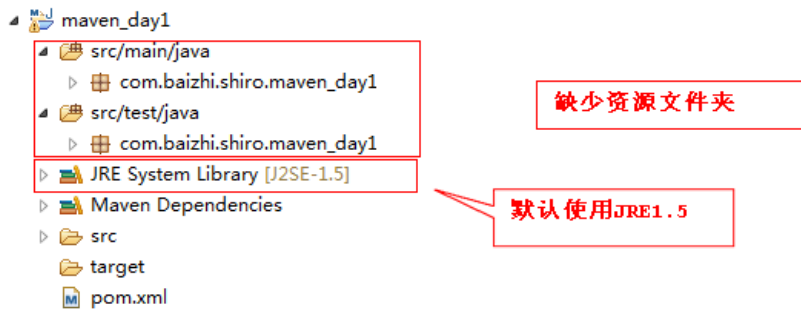


2. 将外部工具包与工具进行集成并配置外部的配置文件



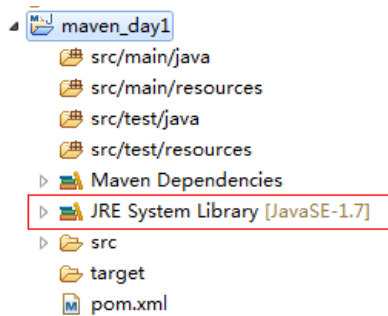
3. 创建一个maven项目



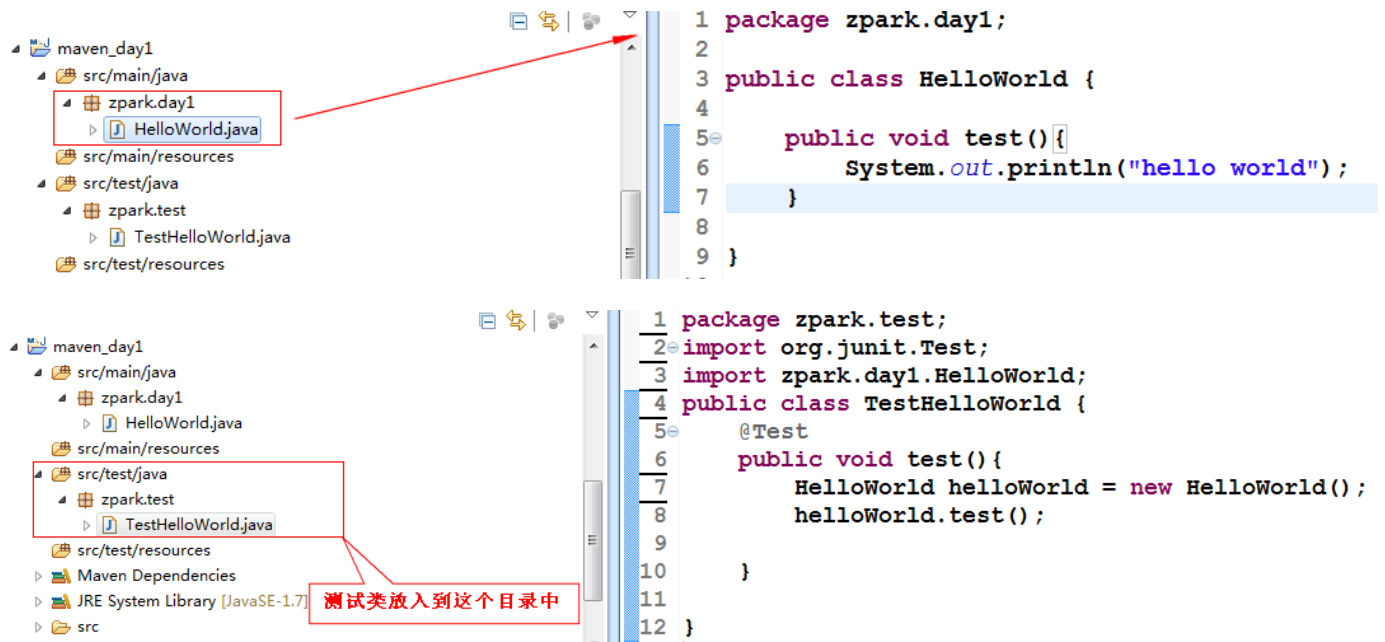


4. 在pom.xml中修改默认项目的编译环境

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.6.0</version>
  <configuration>
    <target>1.7</target>
    <source>1.7</source>
  </configuration>
</plugin>
```



5. 开发项目

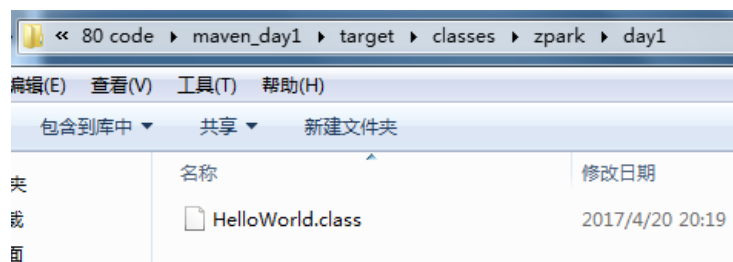


6. 运行测试

```

D:\80 code\maven_day1>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building maven_day1 0.0.1-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ maven_day1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.6.0:compile (default-compile) @ maven_day1 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to D:\80 code\maven_day1\target\classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.017s
[INFO] Finished at: Thu Apr 20 20:19:25 CST 2017

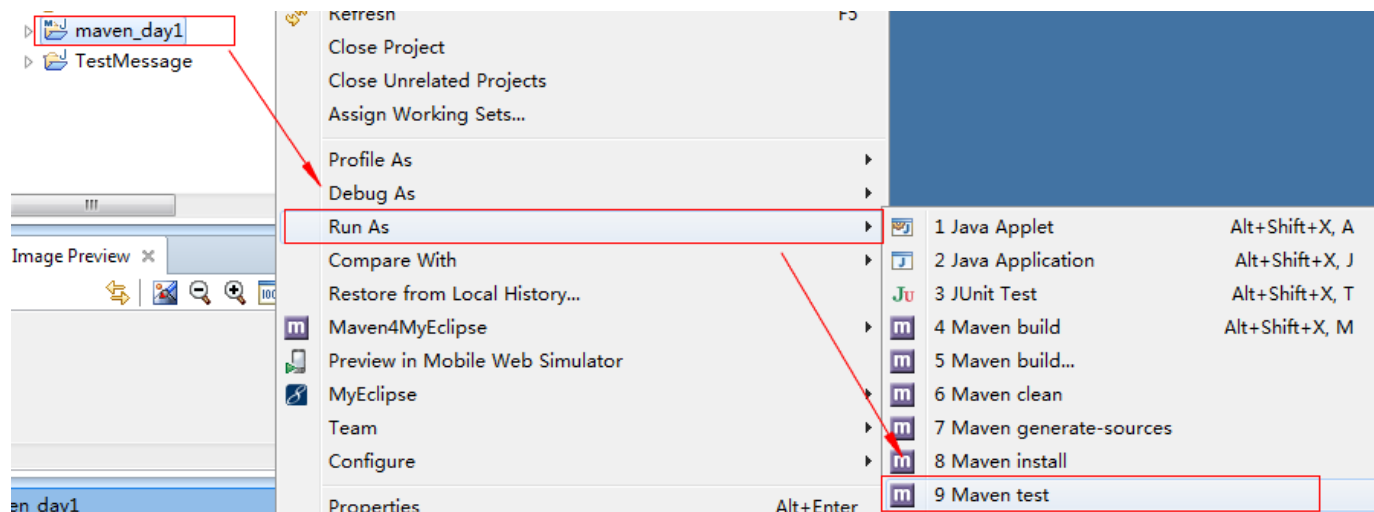
```



7. 目录结构解析

目录结构	描述
/src/main/java	源代码目录
/src/main/resources	源代码配置文件目录
/src/test/java	测试代码目录
/src/test/resources	测试代码配置文件目录
/target	构建后生成.jar 文件的放置位置
/target/classes	构建后生成.class 文件放置位置
/pom.xml	maven 配置文件放置位置

8. 通过eclipse工具运行当期项目



Running `zpark.test.TestHelloWorld`

hello world

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.052 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 7.285s  
[INFO] Finished at: Thu Apr 20 20:26:02 CST 2017  
[INFO] Final Memory: 14M/308M  
[INFO] -----
```

9.maven中管理项目中依赖(jar|插件)

1. maven中获取jar包的方式

maven在全世界维护一个中心库,获取jar应该去相应的中心库获取坐标

2. maven的中心库

The screenshot shows the Maven Repository website for the JUnit artifact. The browser tabs include 'maven repository_百度', 'Maven Repository: junit', and 'jquery, 怎么给多个元素'. The address bar shows 'mvnrepository.com/artifact/junit/junit'. The page features the Maven Repository logo, a search bar with 'junit' entered, and a 'Search' button. On the left, there's a section for 'Indexed Artifacts (6.15M)' with a line graph showing growth from 2004 to 2017. Below that is a 'Popular Categories' list. The main content area displays 'JUnit' with its logo, a description, and a table of metadata. An advertisement for AdMob by Google is also visible. At the bottom, a table lists the versions of JUnit available in the Central repository.

Version	Repository	Usages	Date
4.12	Central	18,578	(Dec, 2014)

3. 项目中获取jar包

```

<groupId>com.baizhi.shiro</groupId>
<artifactId>maven_day1</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

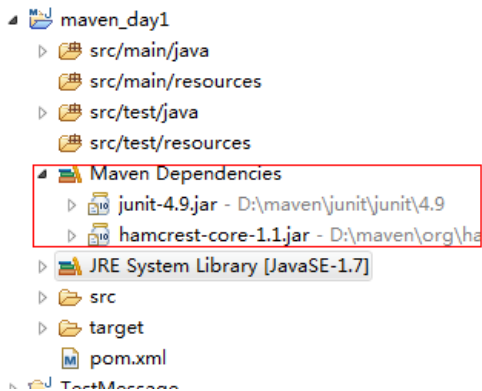
<name>maven_day1</name>
<url>http://maven.apache.org</url>

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

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.9</version>
    <scope>test</scope>
  </dependency>
</dependencies>

```

4. 项目中在maven dependency



10.maven依赖中的scope属性

-在maven项目放入依赖时可以指定scope scope属性的默认值为 **provided**

1、枚举各个属性值的含义

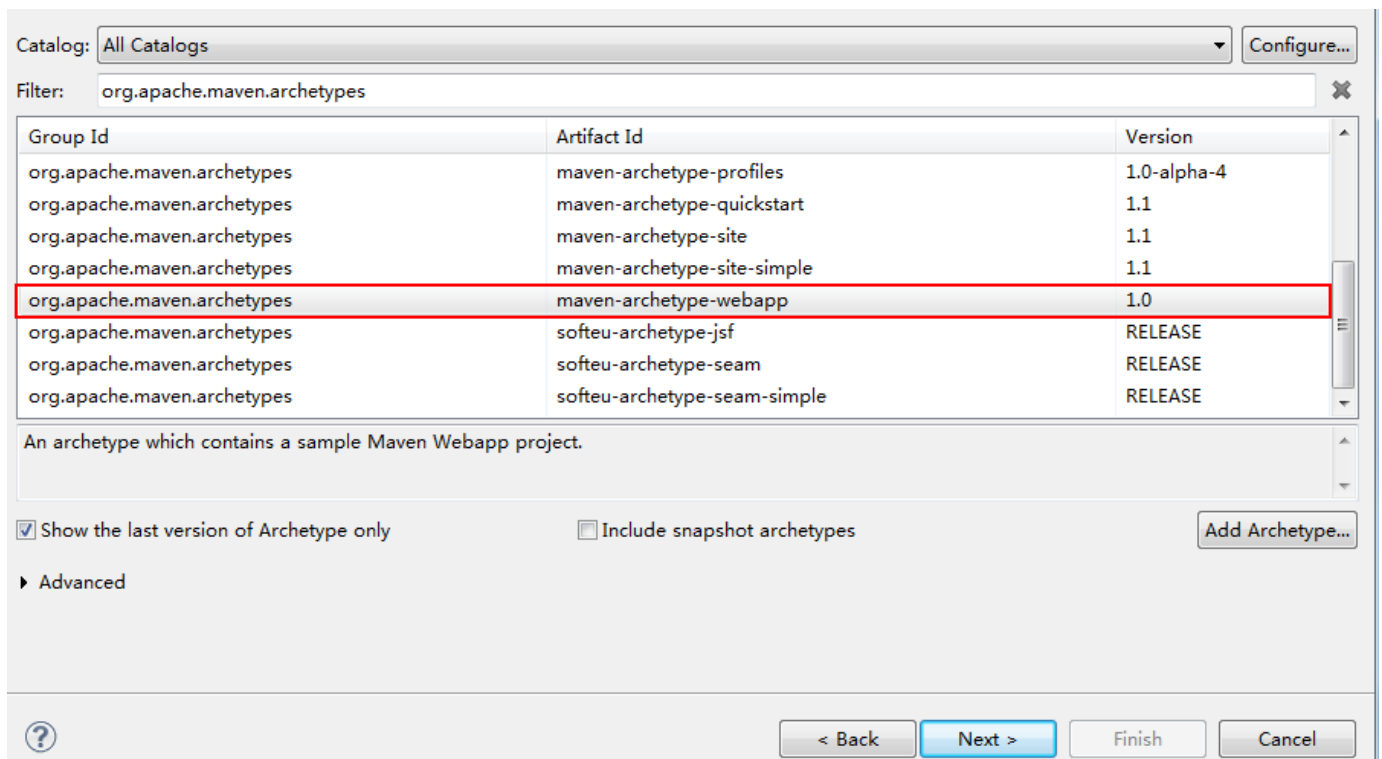
- compile, 缺省值, 适用于所有阶段, 会打包进项目。
- provided, 类似compile, 期望JDK、容器或使用者会提供这个依赖。
- runtime, 只在运行时使用, 如DBC驱动, 适用运行和测试阶段。
- test, 只在测试时使用, 用于编译和运行测试代码。不会随项目发布。
- system, 类似provided, 需要显式提供包含依赖的jar, Maven不会在Repository中查找它。

2、其它类型的属性值都比较容易理解, 这里重点比较一下compile和runtime之间的区别:

- (1) 先描述一个简单的例子: 模块A依赖X, 此时X的scope设置的值为runtime;
- (2) 另一模块B依赖A, 则B在编译时不会依赖X (编译时不会有任何问题);

11.maven构建web项目

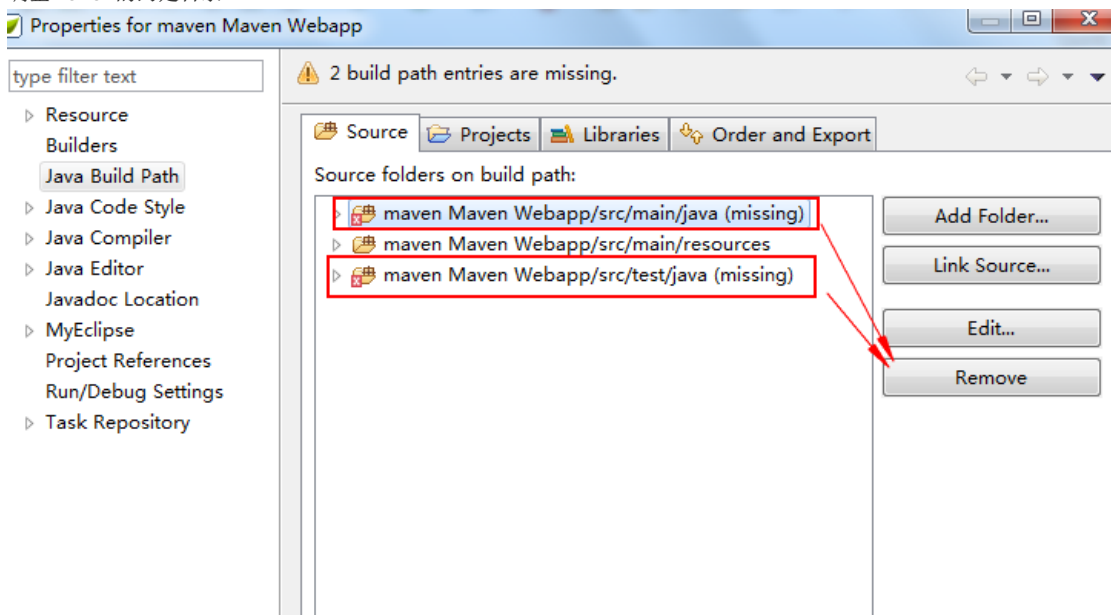
1. 在工作空间中创建maven的项目



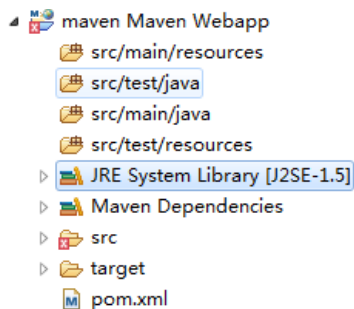
2. 创建完成的项目



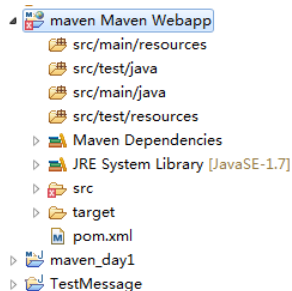
3. 调整maven的约定目录



4. 创建现约定目录

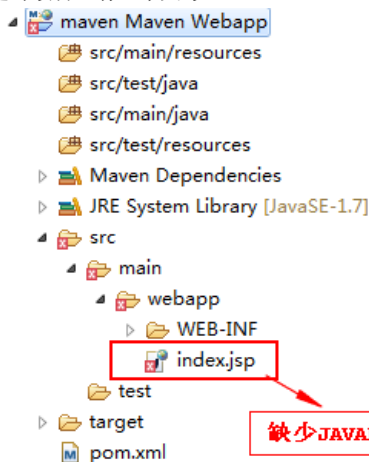


5. 修改编译的环境



```
19 <finalName>maven</finalName>
20 <plugins>
21 <plugin>
22   <groupId>org.apache.maven.plugins</groupId>
23   <artifactId>maven-compiler-plugin</artifactId>
24   <version>2.3.2</version>
25   <configuration>
26     <target>1.7</target>
27     <source>1.7</source>
28   </configuration>
29 </plugin>
```

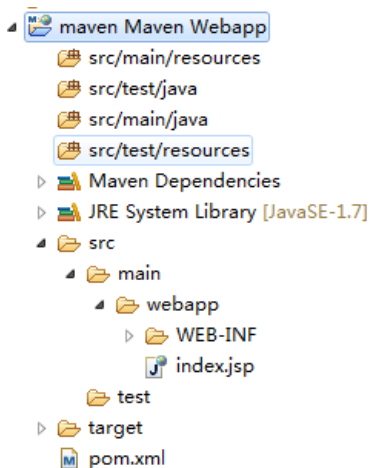
6. 这时项目还有一个叉号



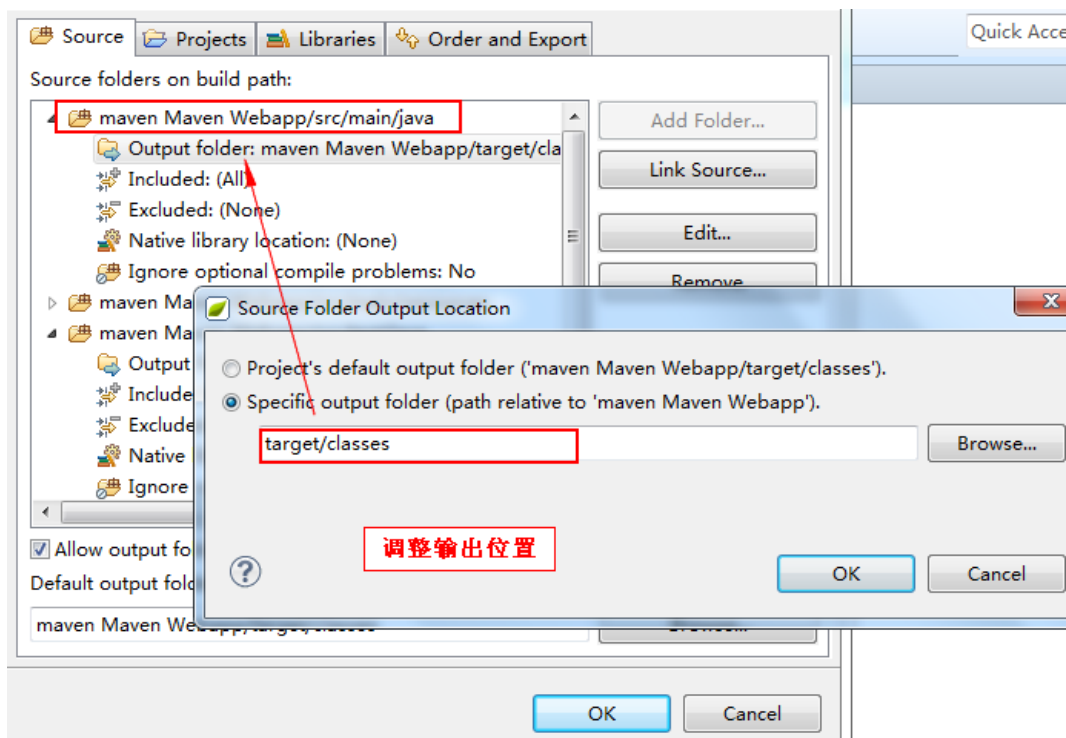
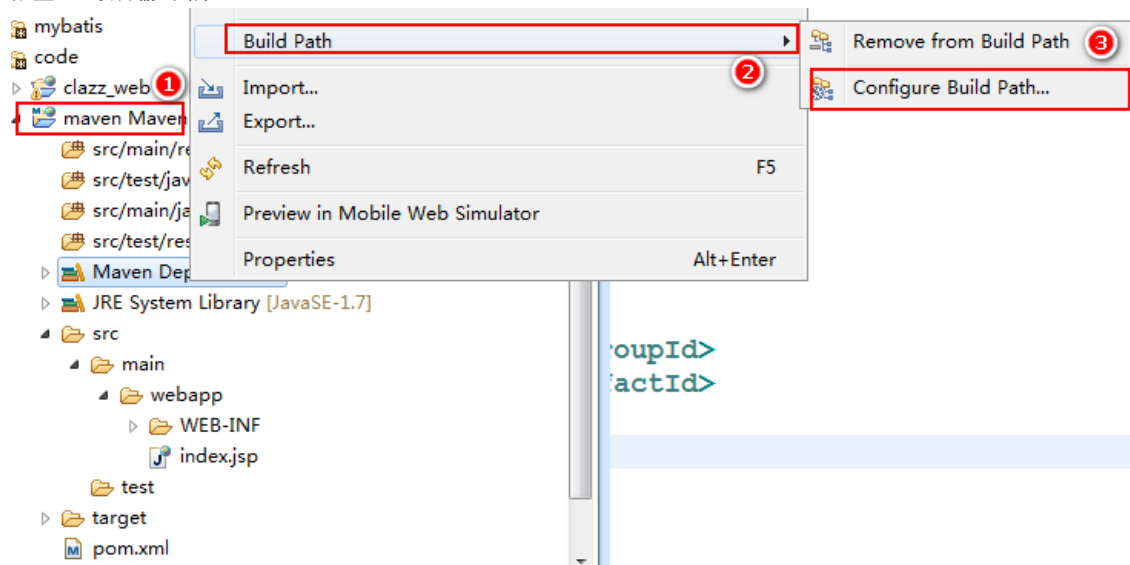
7. 引入web规范 servelt-api jsp-api jstl-jar

```
<dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.0</version>
  <scope>test</scope>
</dependency>
```

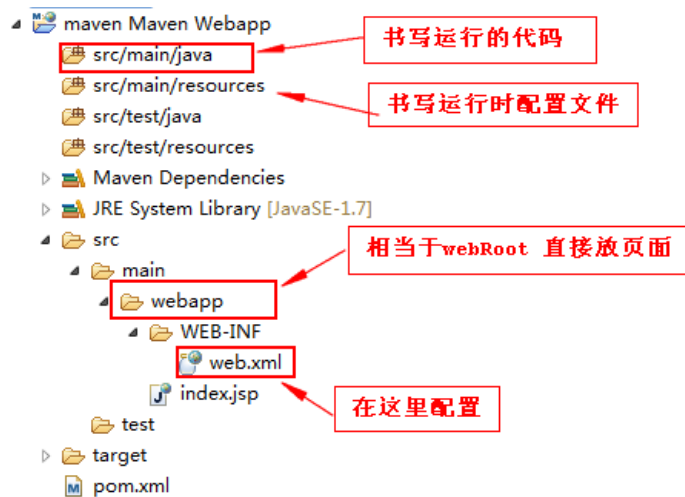
8. 更新项目即可



9. 配置web项目输出路径



10. 解析web项目的目录



12.将驱动jar包打入到本地的仓库

1. maven的中心库中oracle的驱动jar包无法下载需要经过授权才行,手动下载

ojdbc6.jar

2. 手动安装到maven中 执行如下命令:

```
mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc6 -Dversion=6.0 -Dpackaging=jar -Dfile=D:/ojdbc6.jar
```

注:

- 1) 若想执行上述语句, 首先需要配置Java的环境变量和Maven的环境变量;
- 2) 请注意上述的版本号和ojdbc.jar路径;

```
[INFO] Installing C:\Users\Administrator\AppData\Local\Temp\mvninstall13427444107
269683675.pom to D:\maven\com\oracle\ojdbc6\6.0\ojdbc6-6.0.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.757s
[INFO] Finished at: Fri Apr 21 00:45:44 CST 2017
[INFO] Final Memory: 9M/245M
[INFO] -----
```