## [Maven仓库分类](https://www.cnblogs.com/understander/p/5508251.html)

**MAVEN仓库分类**

**Maven仓库分为：本地仓库+远程仓库两大类**

**远程仓库又分为：中央仓库+私服+其它公共远程仓库**

**1，在Maven中，任何一个依赖、插件或者项目构建的输出，都可以称之为构件**

**2，Maven在某个统一的位置存储所有项目的共享的构件，这个统一的位置，我们就称之为仓库。（仓库就是存放依赖和插件的地方）**

**3，任何的构件都有唯一的坐标，Maven根据这个坐标定义了构件在仓库中的唯一存储路径**

**4，本地仓库，就是Maven在本地存储构件的地方(maven的本地仓库，在安装maven后并不会创建，它是在第一次执行maven命令的时候才被创建)**

**5，当maven寻找构件时，首先从本地仓库中寻找，若在本地仓库中找不到，则向远程仓库中寻找，远程仓库若还不能找到则会报错，远程仓库找到就下载到本地再使用。因此随着maven的不断的使用下载构件，它的本地仓库将越来越大**

**6，一个构件只有在本地仓库中之后，才能由其他Maven项目使用**

**7，中央仓库是maven自带的远程仓库，默认地址：http://repol.maven.org/maven2**

**8，私服是架设在本机或者局域网中的一种特殊的远程仓库，通过私服可以方便的管理其它所有的外部远程仓库**

**9，本地仓库默认的地址是：${user:home}/.m2/repository**

**10，在项目中添加远程仓库，当中央仓库找不到所需构件时，我们可以配置pom.xml文件，添加其它远程仓库，其中的id必须唯一（若不唯一，设置为central将覆盖中央仓库的位置 ）**

**11，默认情况下，不管是在windows还是linux上，每个用户在自己的用户目录下都有一个路径名为.m2/repository/的仓库目录。有时候，因为某些原因(例如磁盘空间不足等)，用户会想要自定义本地仓库目录地址，则需要修改settings.xml配置文件。（此时更改后，所有的用户都会受到影响，而且如果maven进行升级，那么所有的配置都会被清除，所以要提前复制和备份M2\_HOME/conf/settings.xml文件，故一般情况下不推荐配置全局的settings.xml）**

[Maven初步搭建 (一)](https://www.cnblogs.com/understander/p/5508156.html)

**什么是maven？**

**也许很多人开始的时候跟我一样，在看了很多工程之后都不知道这个鸟东西到底是用来干嘛用的！:-D**

**一个东西之所以会出现是有其原因的，譬如Linus大神写git。**

[Maven](http://baike.baidu.com/view/336103.htm)**项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件**[项目管理工具](http://baike.baidu.com/view/3169783.htm" \t "_blank)**。**

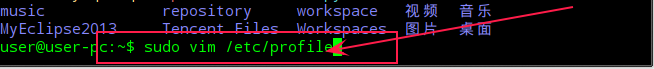
**简单来说，它就是一个项目管理工具。在后续的框架中（Struts，Spring，Hibernate等）将大量使用它。**

**在最开始阶段，我们需要在myeclipse中搭建一个maven仓库。**

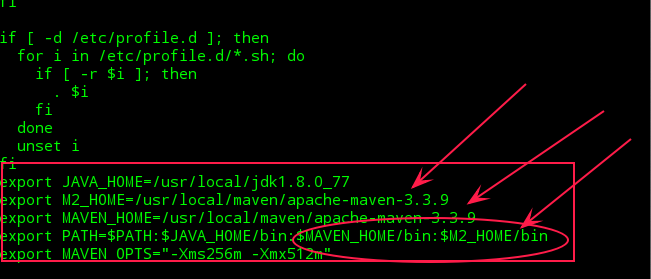
#### **1.可以去官网下载一个**[maven](http://maven.apache.org/download.cgi" \t "_blank)**，解压缩并安装好。（我使用的是maven-3.3.9）**

#### **2.下载之后需要配置maven环境变量路径。**

**linux系统下打开配置文件：**



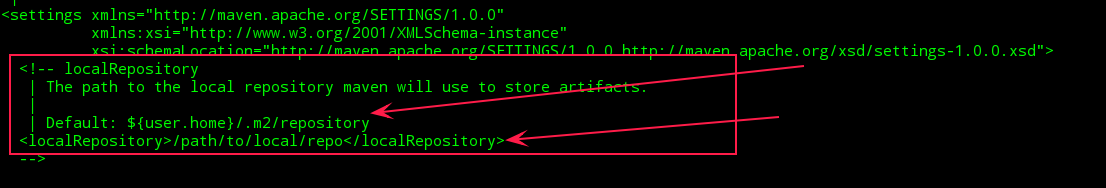
**插入如下路径名称：（JAVA\_HOME是JDK的路径，maven的路径M2\_HOME和MAVEN\_HOME只要使用一个就可以了）**



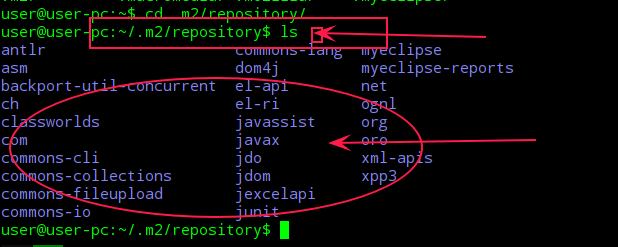
**3.修改maven本地仓库：（不是必要操作，若不进行修改，maven会使用默认的位置作为本地仓库）**

**在下载好的maven文件目录下有一个conf的配置文件夹，修改其setting.xml文件中的内容：**

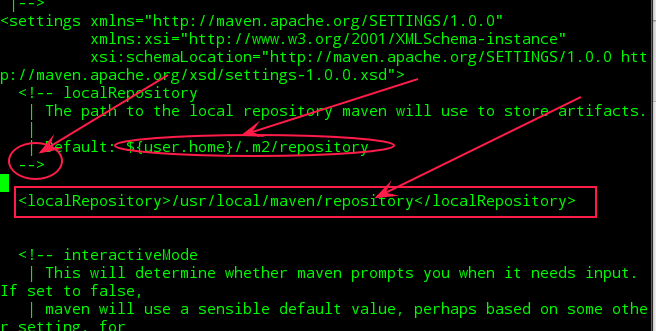
**------>修改前：（可以看到maven使用的是默认位置下的仓库）**



**进入maven的默认仓库，我们会看到有许多文件都保存在其中，而后续的修改了本地仓库，若缺少这些东东，maven则不能正常运行。**

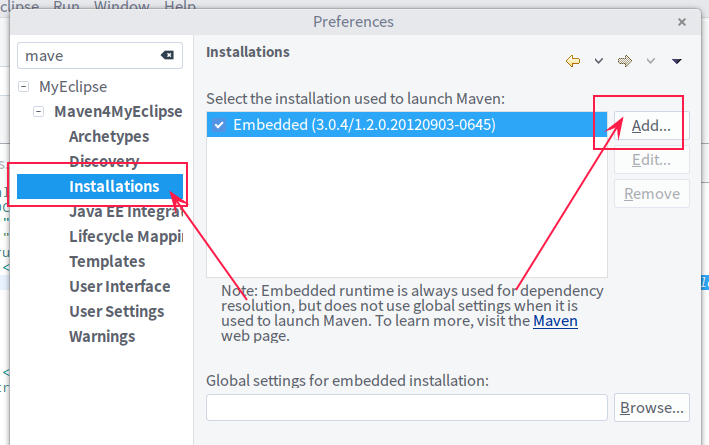


**------>修改后：（需要注意的是修改了自己需要保存的目标路径之后，需要将原来的注释符上移，即为图中第一个圆括号中的内容）**

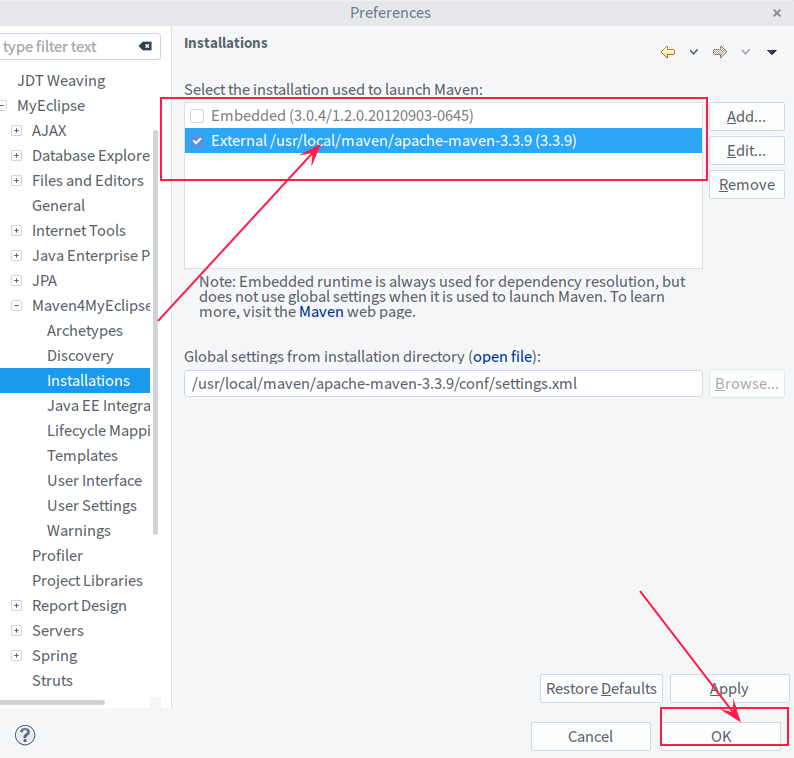


#### **4.MyEclipse中配置Maven插件**

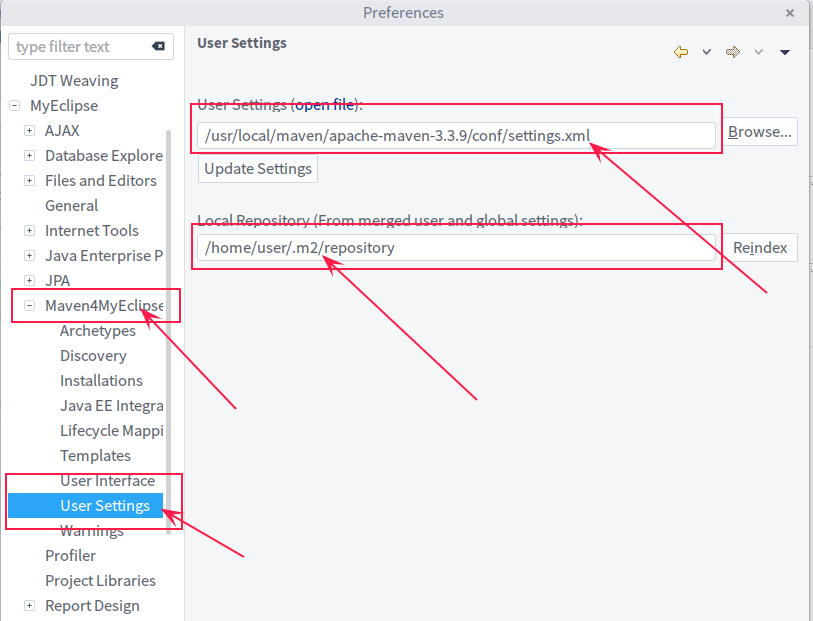
**打开MyEclipse ---> widnow--->preferences --->myeclipse --->maven4MyEclipse --->Installations ---> add --->**



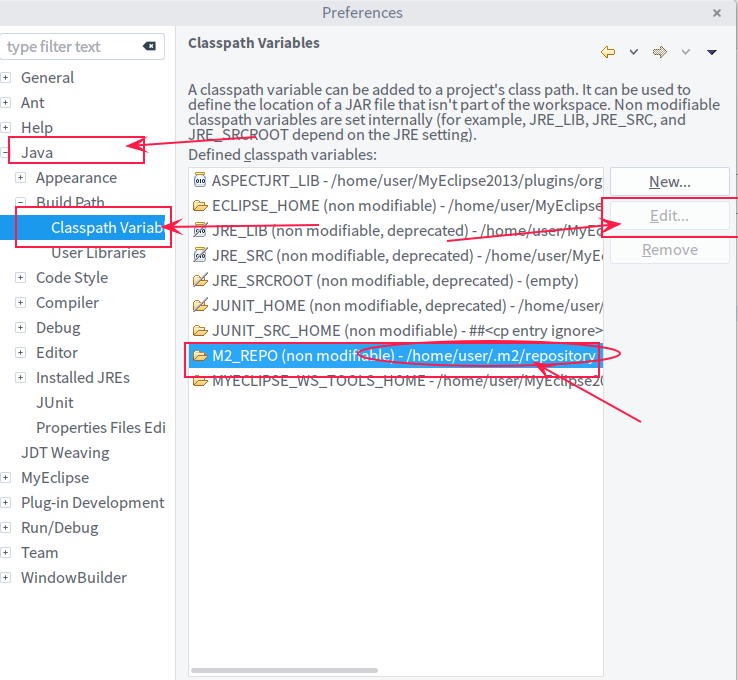
**选择第一步已经解压缩安装好的mavenl --->finish ---> ok**



#### **5.Maven4MyEclipse --->选择 User Settings ---> 在右侧User Settings栏中选择Maven安装目录下的/conf/settings.xml配置文件 --->Local Repository中选择刚刚从新设置的本地仓库所在文件路径（若没有修改则使用默认的仓库位置）---> ok**



#### **6. 选择Window---> preferences ---> java ---> BuildPath ---> Classpath variables ---> 选择有m2\_repo项 --->edit--->将其修改成你的本地仓库所在路径（若是默认仓库则不需要修改。）---> ok**



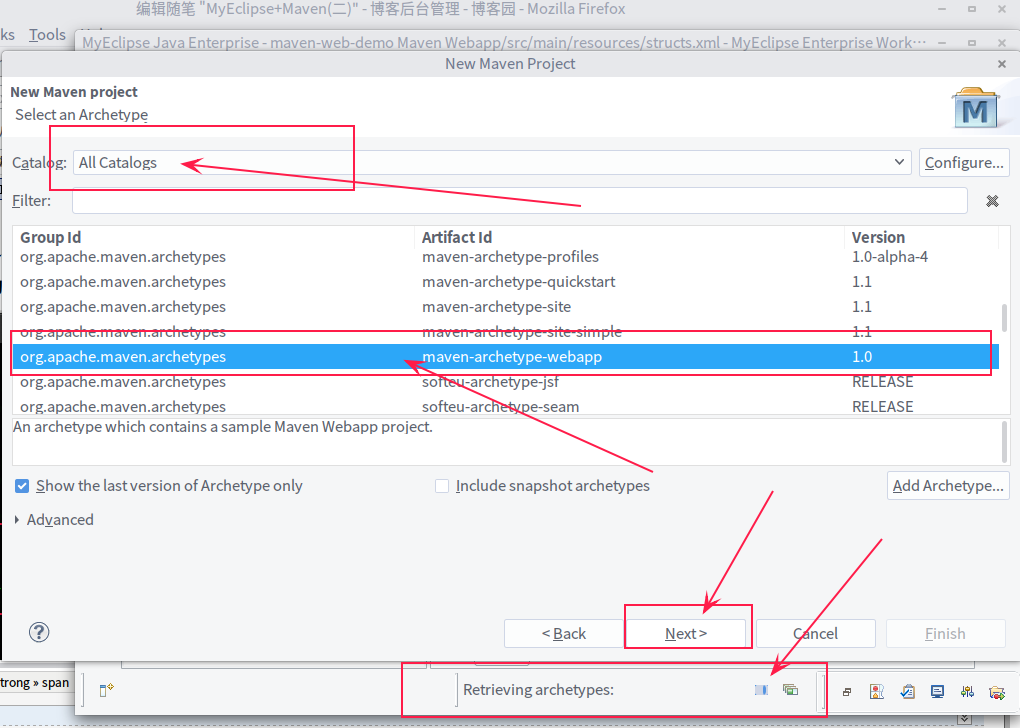
**7.搭建过程完成。**

## [Maven创建webapp(二)](https://www.cnblogs.com/understander/p/5508908.html)

这一节将记录在myeclipse下用maven创建一个简单的webapp项目

web开发maven仓库自动添加组件，故需要需要保持网络的通畅。

打开myeclipse  -->  File  --> project  --> 选取Maven Project  --> next  --> next  -->



我们选择创建一个简单的webapp项目，如果是第一次使用maven，本地仓库需要从远程仓库下载一些基本的常用构建，这个过程很耗费时间(我想说真TMD的耗费时间)。

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

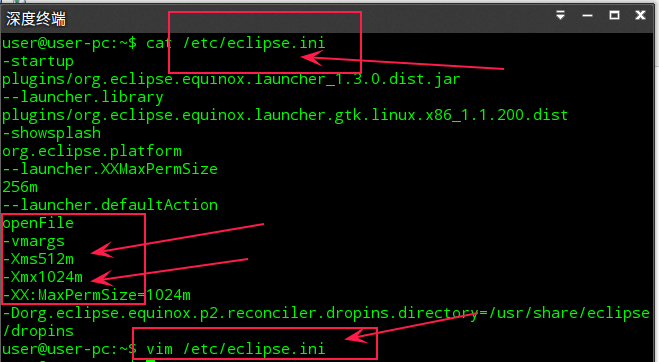
问题：An internal error occurred during: "Retrieving archetypes:".GC overhead limit exceeded

若出现gc overhead limit exceeded eclipse的错误：原因是eclipse默认配置内存太小，需要修改eclipse安装文件下的eclipse.ini文件。

使用vim 打开/etc/eclipse.ini文件

修改以下内容为：-Xms 512m(初始化最小内存)  -Xmx 1024m（最大占有内存）-XX:MaxpermSize=1024m(编译时一直占用最大内存)

重新启动myeclipse。

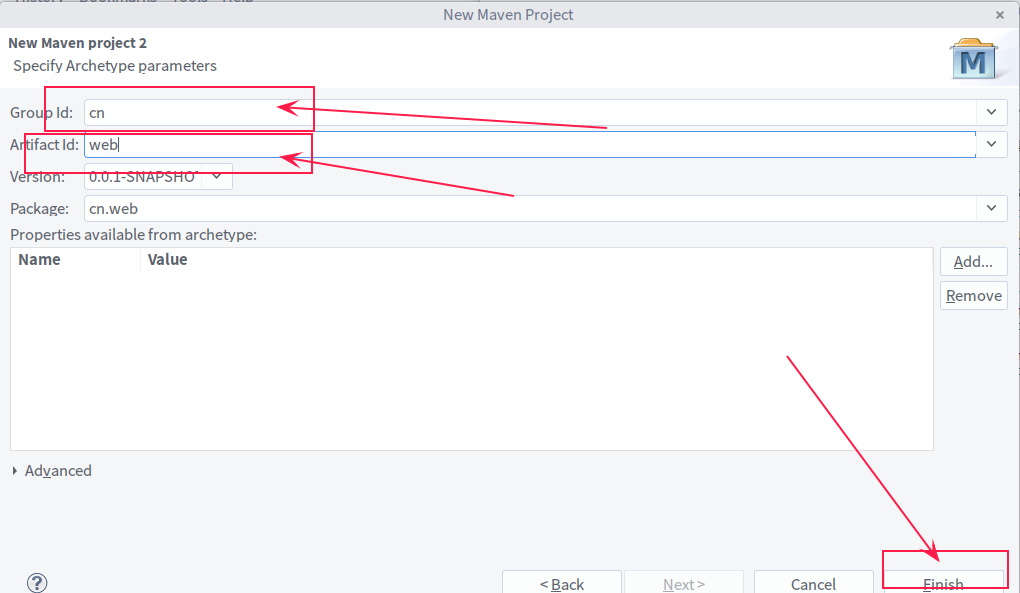


--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

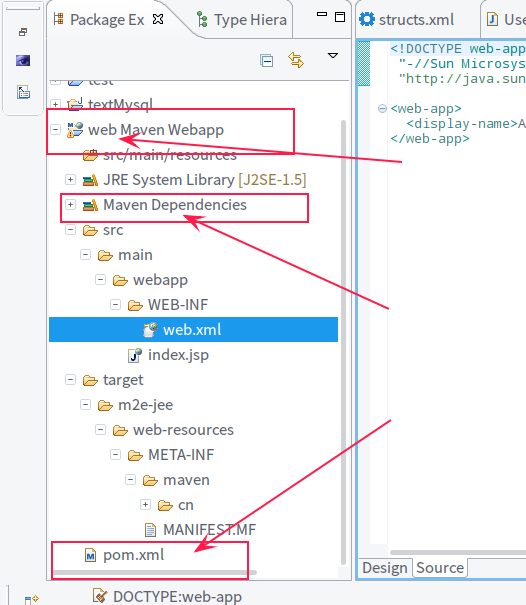
  -->选取一个webapp  --> next  -->

group id:选取一个名字（例如cn）

artifact id:取一个名字（例如web）

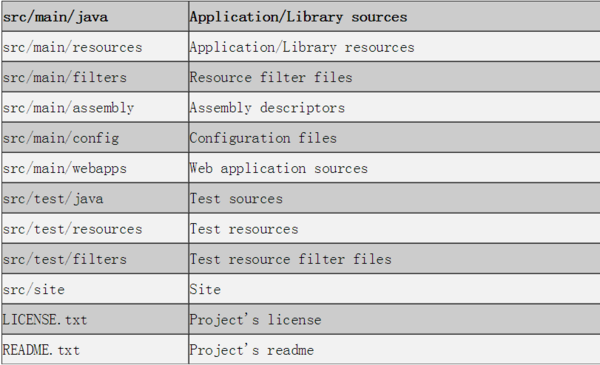


  --> finish之后在工作区可以看到该web项目（如下图）  --> 创建完成



顺便再说一下maven约定的文件结构：

一般的src/main/java    src/main/resources    src/test/java     src/test/resources这四个源文件是Maven提倡使用一个共同的标准目录结构，使开发人员能在熟悉了一个Maven工程后，对其他的Maven工程也能清晰了解。这样做也省去了很多设置的麻烦。



# [Maven创建servlet项目演示（三）](https://www.cnblogs.com/understander/p/5510474.html)

## **上一节用Maven新建了web项目成功后，本文演示在此基础上应用servlet。**

**从对tomcat服务器进行配置可的过程中可以知道，tomcat作为servlet容器运行，负责处理客户请求，把请求传送给servlet并把结果返回给客户。**

**这里的servlet作用是处理该客户请求的最终实体类。**

**servlet实际上是一个有规定接口，属性和方法的java类，tomcat容器与servlet之间的接口是由servlet API定义好的，这个接口定义了tomcat容器在servlet上要调用的方法及传递给servlet的对象。**

**tomcat服务器会在启动时，将在web.xml中配置好的servlet对象实例化，并加载到服务器的内存中，若接收到HTTP请求，服务器将调用相应的servlet对象来进行处理。**

**所有的servlet都必须实现javax.servlet.Servlet接口，但是我们在进行http协议相关的web开发时，一般直接通过继承javax.servlet.HttpServlet类来实现。**

1.首先修改pom.xml文件，添加servlet依赖

[复制代码](javascript:void(0);)

**1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**2 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">**

**3 <modelVersion>4.0.0</modelVersion>**

**4 <groupId>com.lei.demo</groupId>**

**5 <artifactId>maven-web-demo</artifactId>**

**6 <packaging>war</packaging>**

**7 <version>0.0.1-SNAPSHOT</version>**

**8 <name>maven-web-demo Maven Webapp</name>**

**9 <url>http://maven.apache.org</url>**

**10 <dependencies>**

**11 <!-- JUnit配置 -->**

**12 <dependency>**

**13 <groupId>junit</groupId>**

**14 <artifactId>junit</artifactId>**

**15 <version>3.8.1</version>**

**16 <scope>test</scope>**

**17 </dependency>**

**18 <!-- 添加Servlet -->**

**19 <dependency>**

**20 <groupId>javax.servlet</groupId>**

**21 <artifactId>servlet-api</artifactId>**

**22 <version>3.1-b02</version>**

**23 <scope>provided</scope>**

**24 </dependency>**

**25 </dependencies>**

**26 <build>**

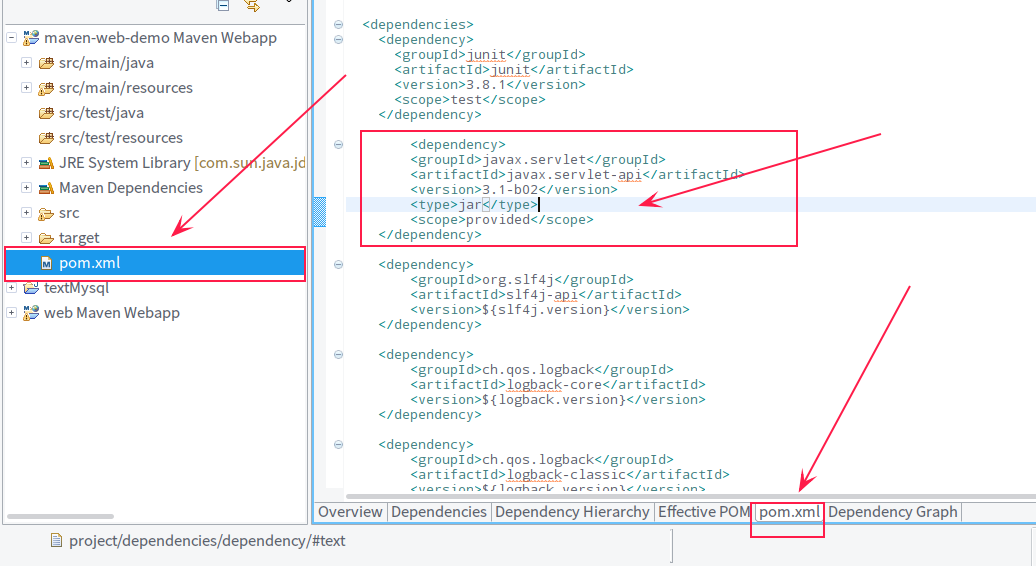
**27 <finalName>maven-web-demo</finalName>**

**28 </build>**

**29 </project>**

[复制代码](javascript:void(0);)

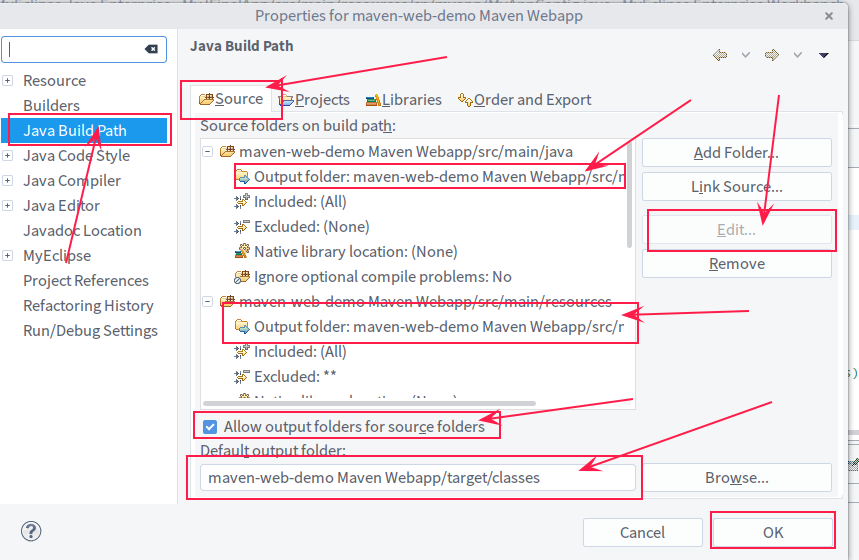
**修改完保存后，项目会从Maven仓库中自动添加servlet-api-3.1-b02.jar包的引用，如果仓库中没有，会自动下载。操作步骤如下：**



**简单说下：source folder和folder区别。因为本人在创建source folder的时候，总是莫名其妙的创建不了，或创建一个source folder结果它就跑到folder里面去了。（好像是一个bug来着，若你也出现同样的问题，请在创建source folder中以source folder，folder，package不同的形式来创建它。）**

**source folder ：存放java源代码的文件夹,当然也包括一些package文件夹,还可以包含其他文件. 项目构建后,source folder里面的java自动编译成class文件到相应的bin文件夹中,其他文件也会移到到相应的目录下.**

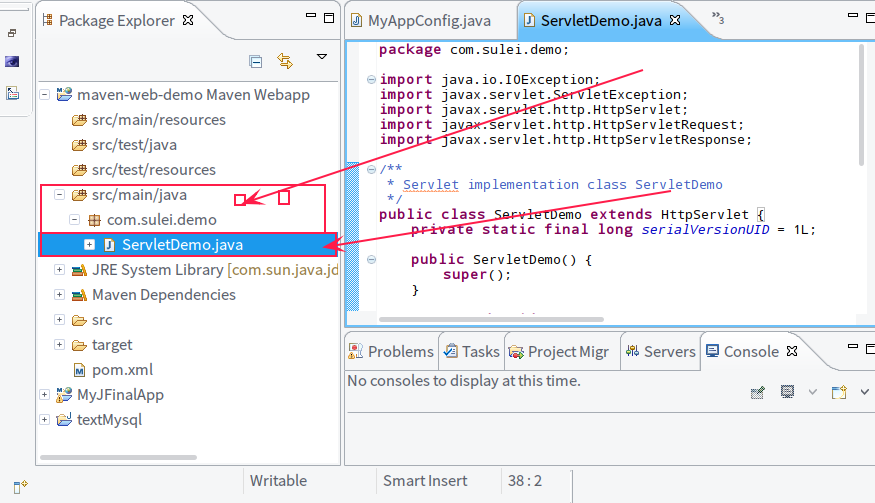
**folder ：里面可以放入任何文件.包括java源文件,jar文件,其他文件(例如,图片,声音等).在此我说明一下,如果里面含有java源文件,不管程序是否 正 确,eclipse都不会报错,把它们当做普通文件处理.但是项目如果要使用这里面的文件,情况就不同了.**



**（很多要在**[Java](http://lib.csdn.net/base/17" \o "Java EE知识库" \t "_blank)**中访问的文件可以放在src下或者便于区分新建特定名字的source folder目录放在下面，这样的会随着编译自动将java文件被编译到default output folder路径下面（若你重新设置了output folder文件路径，则编译到新设置的路径文件下），而不是java类型的文件，则是直接拷贝过去，这一切对于开发者都是透明的。需要注意的是，无论是使用默认或修改了的putput folder，在myeclipse工作目录中都以不可见的形式存在，当然你可以在文件夹中找到它）**

2.新建一个Servlet

在source folder下的src/main/java下创建一个com.sulei.demo包，包下创建一个ServletDemo类。



**一个servlet从产生到结束的流程如下：**

**a. servlet容器创建servlet的一个实例，在需要该servlet处理的请求到时触发。**

**b.容器调用该实例的init（）方法。**

**c. 如果容器对该servlet有请求，则调用此实例的service（）方法。（注意是service而不是servlet）**

**d. 容器在销毁本实例前调用它的destroy（）方法，并还进行垃圾收集。**

**所以servlet生命周期常常被定义为：加载，实例化，初始化，处理客户请求和销毁。一旦清清了一个servlet，就没有办法阻止容器执行一个完整的生命周期了。**

**servlet类代码如下：**

[复制代码](javascript:void(0);)

1 package com.sulei.demo;

2

3 import java.io.IOException;

4 import javax.servlet.ServletException;

5 import javax.servlet.http.HttpServlet;

6 import javax.servlet.http.HttpServletRequest;

7 import javax.servlet.http.HttpServletResponse;

8

9 /\*\*

10 \* Servlet implementation class ServletDemo

11 \*/

12 public class ServletDemo extends HttpServlet {

13 private static final long serialVersionUID = 1L;

14

15 public ServletDemo() {

16 super();

17 }

18 　　//doGet方法处理GET类型的请求

19 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

20 doPost(request, response);

21 }

22 　　//doPost处理POST类型的请求

23 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

24 request.setCharacterEncoding("UTF-8");

25 response.setContentType("text/html;charset=utf-8");

26

27 String action = request.getParameter("action");

28 if("login\_input".equals(action)) {

29 request.getRequestDispatcher("login.jsp").forward(request , response);

30 } else if("login".equals(action)) {

31 String name = request.getParameter("name");

32 String password = request.getParameter("password");

33

34 System.out.println("name->" + name + ",password->" + password);

35 }

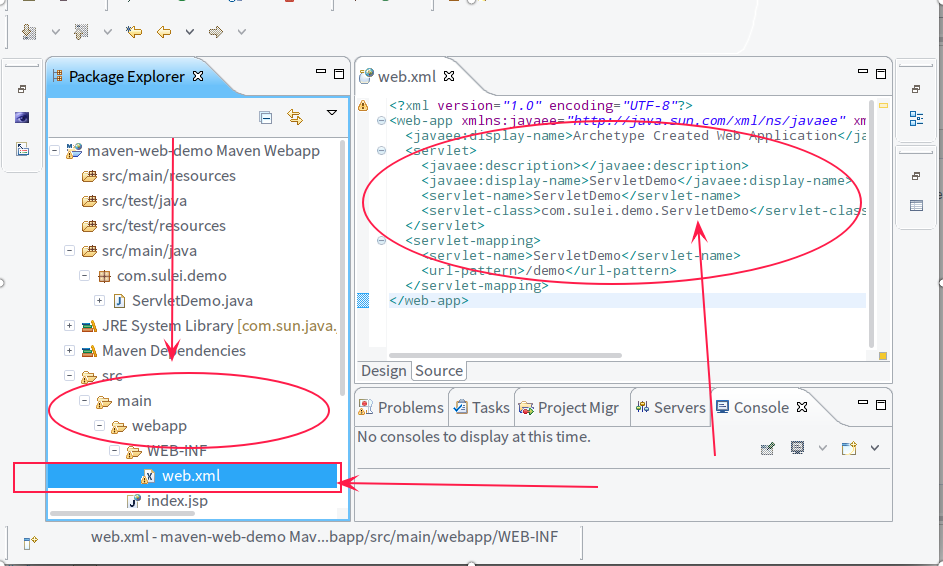
36 }

37

38 }

[复制代码](javascript:void(0);)

**3.修改web.xml配置文件**



**web.xml配置文件的内容为：**

[复制代码](javascript:void(0);)

**1 <?xml version="1.0" encoding="UTF-8"?>**

**2 <web-app xmlns:javaee="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">**

**3 <javaee:display-name>Archetype Created Web Application</javaee:display-name>**

**4 <servlet>**

**5 <javaee:description></javaee:description>**

**6 <javaee:display-name>ServletDemo</javaee:display-name>**

**7 <servlet-name>ServletDemo</servlet-name>**

**8 <servlet-class>com.sulei.demo.ServletDemo</servlet-class>**

**9 </servlet>**

**10 <servlet-mapping>**

**11 <servlet-name>ServletDemo</servlet-name>**

**12 <url-pattern>/demo</url-pattern>**

**13 </servlet-mapping>**

**14 </web-app>**

[复制代码](javascript:void(0);)

4.创建index.jsp

jsp文件内容如下：

[复制代码](javascript:void(0);)

**1 <%@ page language="java" contentType="text/html; charset=UTF-8"**

**2 pageEncoding="UTF-8"%>**

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

**4 <html>**

**5 <head>**

**6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">**

**7 <title>Insert title here</title>**

**8 </head>**

**9 <body>**

**10 <p>用Maven创建web项目，测试Servlet</p>**

**11 <a href="demo?action=login\_input">登录(demo?action=login\_input)</a>**

**12 </body>**

**13 </html>**

[复制代码](javascript:void(0);)

5.创建login.jsp

**jsp文件内容如下：**

[复制代码](javascript:void(0);)

**1 <%@ page language="java" contentType="text/html; charset=UTF-8"**

**2 pageEncoding="UTF-8"%>**

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

**4 <html>**

**5 <head>**

**6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">**

**7 <title>Insert title here</title>**

**8 </head>**

**9 <body>**

**10 <form action="demo?action=login" method="post">**

**11 Name:<input type="text" name="name" />**

**12 Password:<input type="password" name="password" />**

**13**

**14 <input type="submit" value="登录" />**

**15 </form>**

**16 </body>**

**17 </html>**

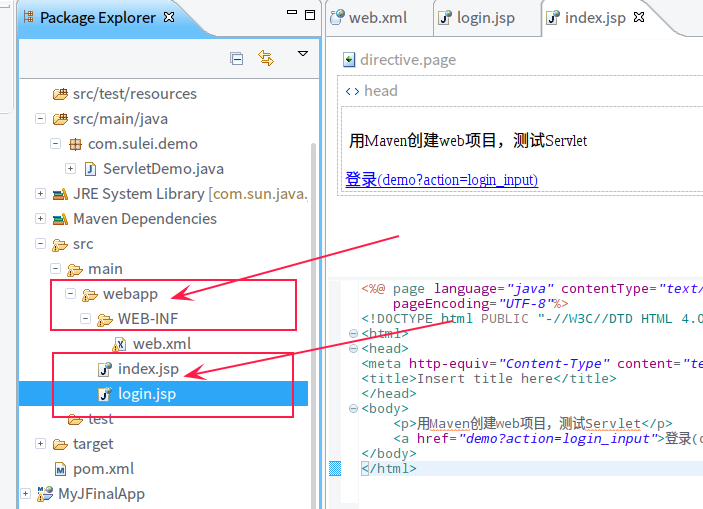
[复制代码](javascript:void(0);)

**扩展内容：JSP是基于servlet的扩展，实际上JSP编译后就是servlet文件，不过使用jsp技术使得网页开发变得更加方便而已。**

**servlet中只能通过out.println()语句输出HTML语法。**

**JSP网页是传统的HTML文件中加入java程序片段和JSP标签构成的，servlet/JSP容器收到客户端的请求时，首先执行网页中的程序代码，然后把结果以HTML格式响应给客户端。**

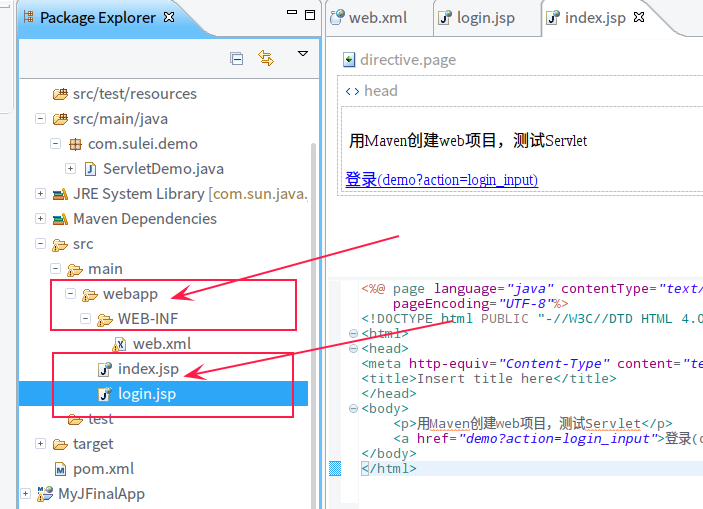
**现在效果图如下：（注意jsp文件存放的位置都是存放在webapp文件夹下）**



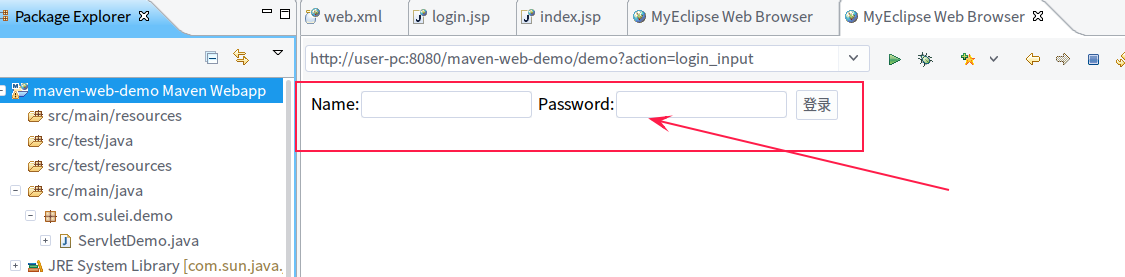
6.运行代码：右键单击工程名称-->run as -->myeclipse service application--> tomcat 7 .x-->ok

各步骤截图如下：

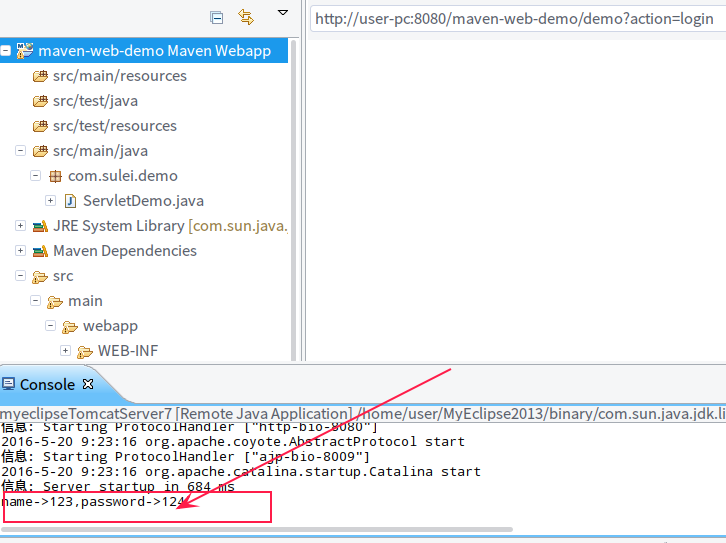
a.进入index页面



**b.进入login页面**



c.在窗口可以看到运行java程序获取jsp页面的name和password结果



# [Hibernate+maven+mysql](https://www.cnblogs.com/understander/p/5523139.html)

**最近在研究hibernate，想建立一个简单的Hibernate+maven+mysql工程，网上找了一大堆的示例，要么看不懂结构，要么就是缺少必要文件。总之都没有成功，结果无意在一个外文网上找了一个实例，惊叹于人家的排版。也不得不感叹的说下浩瀚的百度里，想找到自己想要的东西是多么难啊！**

**以下借鉴人家文章和内容并做部分补充，以供初学者一起分享：**

**（准备工作myeclipse的安装+maven的下载+mysql的配置略过）**

**1.mysql中建一个mkyong的数据库，其下面新建一个stock的空表格。**

[复制代码](javascript:void(0);)

**//启动mysql服务器**

**sudo service mysql start**

**//进入mysql**

**mysql -uroot -p**

**输入mysql密码**

**//创建数据库mkyong**

**>create database mkyong;**

**//使用数据库mkyong**

**>use mkyong;**

**//在数据库mkyong中创建表stock。请直接copy。**

**>create table stock (**

**`STOCK\_ID` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT,**

**`STOCK\_CODE` VARCHAR(10) NOT NULL,**

**`STOCK\_NAME` VARCHAR(20) NOT NULL,**

**PRIMARY KEY (`STOCK\_ID`) USING BTREE,**

**UNIQUE KEY `UNI\_STOCK\_NAME` (`STOCK\_NAME`),**

**UNIQUE KEY `UNI\_STOCK\_ID` (`STOCK\_CODE`) USING BTREE**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8;**

[复制代码](javascript:void(0);)

**2.在myeclipse中创建一个maven工程，自己取一个名字。**

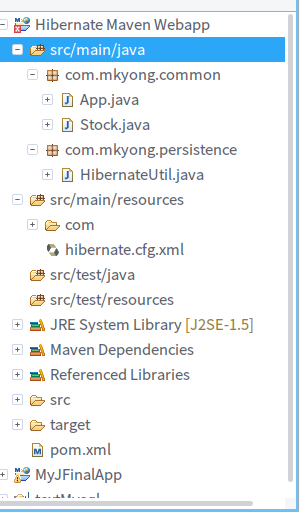
## New --> Project --> Maven Project --> Next --> 在(New Maven Project )中选择创建一个webapp项目

**3.创建源文件（Source Folder）**

**在 “src/main”目录下创建 “/src/main/resources” , 之后所有的Hibernate’s xml配置文件都将放置在该文件下。**

**在“src/main”目录下创建“/src/main/java”文件，其中放置java源代码文件。**

**4.现在该工程的结构看起来如下：**

****

**5.添加Hibernate和Mysql依赖**

**修改项目的pom.xml配置文件，它将支持Hibernate和mysql的连接，Hibernate项目需要dom4j, commons-logging, commons-collections和 cglib文件，mysql需要连接jdbc的文件。将原来的内容更换为如下代码。**

**文件：pom.xml**

[复制代码](javascript:void(0);)

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com</groupId>

<artifactId>Hibernate</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>Hibernate Maven Webapp</name>

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

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- MySQL database driver -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.9</version>

</dependency>

<!-- Hibernate library dependecy start -->

<dependency>

<groupId>dom4j</groupId>

<artifactId>dom4j</artifactId>

<version>1.6.1</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.1.1</version>

</dependency>

<!-- http://mvnrepository.com/artifact/org.hibernate/hibernate -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate</artifactId>

<version>3.2.6.ga</version>

</dependency>

<dependency>

<groupId>commons-collections</groupId>

<artifactId>commons-collections</artifactId>

<version>3.2.1</version>

</dependency>

<dependency>

<groupId>cglib</groupId>

<artifactId>cglib-nodep</artifactId>

<version>3.1</version>

</dependency>

<!-- Hibernate library dependecy end -->

<dependency>

<groupId>javax.transaction</groupId>

<artifactId>jta</artifactId>

<version>1.1</version>

</dependency>

</dependencies>

<build>

<finalName>Hibernate</finalName>

</build>

</project>

[复制代码](javascript:void(0);)

**6.创建Hibernate类表映射文件**

**新建一个Stock.hbm.xml映射文件，路径为：“src/main/resources/com/mkyong/common/Stock.hbm.xml“**

**文件：Stock.hbm.xml**

[复制代码](javascript:void(0);)

<?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>

<class name="com.mkyong.common.Stock" table="stock" catalog="mkyong">

<id name="stockId" type="java.lang.Integer">

<column name="STOCK\_ID" />

<generator class="identity" />

</id>

<property name="stockCode" type="string">

<column name="STOCK\_CODE" length="10" not-null="true" unique="true" />

</property>

<property name="stockName" type="string">

<column name="STOCK\_NAME" length="20" not-null="true" unique="true" />

</property>

</class>

</hibernate-mapping>

[复制代码](javascript:void(0);)

**7.创建一个Stock的POJO  java文件，路径为：“src/main/java/com/mkyong/common/Stock.java”**

**文件：Stock.java**

[复制代码](javascript:void(0);)

package com.mkyong.common;

/\*\*

\* Model class for Stock

\*/

public class Stock implements java.io.Serializable {

private static final long serialVersionUID = 1L;

private Stock stock;

private Integer stockId;

private String stockCode;

private String stockName;

public Stock() {

}

public Stock(String stockCode, String stockName) {

this.stockCode = stockCode;

this.stockName = stockName;

}

public Integer getStockId() {

return this.stockId;

}

public void setStockId(Integer stockId) {

this.stockId = stockId;

}

public String getStockCode() {

return this.stockCode;

}

public void setStockCode(String stockCode) {

this.stockCode = stockCode;

}

public String getStockName() {

return this.stockName;

}

public void setStockName(String stockName) {

this.stockName = stockName;

}

public Stock getOper() {

return stock;

}

public void setOper(Stock stock) {

this.stock = stock;

}

}

[复制代码](javascript:void(0);)

**8.创建Hibernate配置文件**

**创建一个Hibernate配置文件，路径为：“src/main/resources/hibernate.cfg.xml“。（注意修改其中的“数据库密码”，mkyong即为数据库的名称）**

**文件：hibernate.cfg.xml**

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.bytecode.use\_reflection\_optimizer">false</property>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.password">**数据库的密码**</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/**mkyong**</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="person">true</property>

<mapping resource="com/mkyong/common/Stock.hbm.xml"></mapping>

</session-factory>

</hibernate-configuration>

[复制代码](javascript:void(0);)

**9.创建Hibernate会话文件**

**创建一个java类型的Hibernate会话文件，路径为：“src/main/java/com/mkyong/persistence/HibernateUtil.java”。  
文件：HibernateUtil.java**

[复制代码](javascript:void(0);)

package com.mkyong.persistence;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static final SessionFactory sessionFactory = buildSessionFactory();

private static SessionFactory buildSessionFactory() {

try {

// Create the SessionFactory from hibernate.cfg.xml

return new Configuration().configure().buildSessionFactory();

}

catch (Throwable ex) {

// Make sure you log the exception, as it might be swallowed

System.err.println("Initial SessionFactory creation failed." + ex);

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

public static void shutdown() {

// Close caches and connection pools

getSessionFactory().close();

}

}

[复制代码](javascript:void(0);)

**10.创建测试文件**

**创建一个名为App.java的测试文件，路径为：“src/main/java/com/mkyong/common/App.java”.**

**文件：App.java**

[复制代码](javascript:void(0);)

package com.mkyong.common;

import org.hibernate.Session;

import com.mkyong.persistence.HibernateUtil;

public class App

{

public static void main( String[] args )

{

System.out.println("Maven + Hibernate + MySQL");

Session session = HibernateUtil.getSessionFactory().openSession();

session.beginTransaction();

Stock stock = new Stock();

stock.setStockCode("4715");

stock.setStockName("GENM");

session.save(stock);

session.getTransaction().commit();

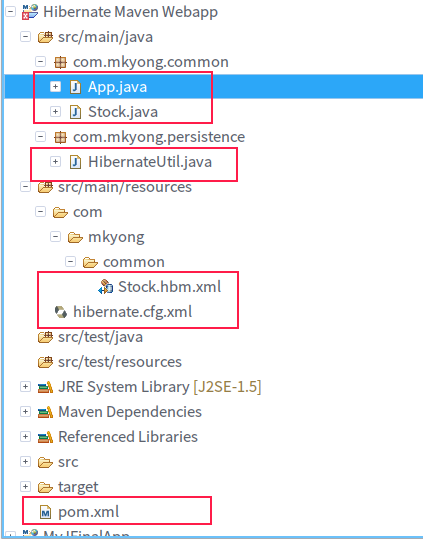
}

}

[复制代码](javascript:void(0);)

**11.再来看看工程的结构**

**创建和修改的六个文件结构如下：**



**12.运行结果**

**运行App.java这个程序，它将会插入一条数据到mysql数据库中去。我们可以在myeclipse中看到该结果，也可以去mysql数据库中查看是否已经插入成功。以下是运行结果：**

[复制代码](javascript:void(0);)

Picked up \_JAVA\_OPTIONS: -Dawt.useSystemAAFontSettings=gasp

Maven + Hibernate + MySQL

.......

五月 24, 2016 5:31:56 上午 org.hibernate.impl.SessionFactoryImpl <init>

信息: building session factory

五月 24, 2016 5:31:56 上午 org.hibernate.impl.SessionFactoryObjectFactory addInstance

信息: Hibernate: insert into mkyong.stock (STOCK\_CODE, STOCK\_NAME) values (?, ?)

[复制代码](javascript:void(0);)

 参考文件链接：[Maven+Hibernate+Mysql Example(XML Mapping)](http://www.mkyong.com/hibernate/quick-start-maven-hibernate-mysql-example/)

<http://www.mkyong.com/hibernate/quick-start-maven-hibernate-mysql-example/>

     源文件下载：[Maven+Hibernate+Mysql Example(XML Mapping)](http://www.mkyong.com/wp-content/uploads/2009/12/Maven_Hibernate_MySQL_Example.zip)

<http://www.mkyong.com/wp-content/uploads/2009/12/Maven_Hibernate_MySQL_Example.zip>

 原文链接：<https://www.cnblogs.com/understander/p/5523139.html>