# Maven2 基础教程(1) - 环境配置

## 目标

通过本文，您可以

- 了解 Maven2
- 将 Maven2 安装到本地机
- 安装 Maven 的 eclipse 插件
- 在 SVN 上下载一个项目,并使用 Maven2 编译、测试、安装、部属等
- 生成 eclipse 的项目文件,并使用 eclipse 的 IDE 编辑
- 在 eclipse 中使用 Maven 方式开发

## 准备

### 预备知识

本文假设您以掌握如下的知识

- JDK 的安装及使用
- eclipse 的安装及使用
- 简单的命令行方式

### 预备软件

在阅读本文的过程中，会使用下列软件，所有软件都附有下载连接地址

- JDK : jdk-1_5_0_17-windows-i586-p.exe
- eclipse : eclipse-SDK-3.4.1-win32.zip
- Maven : apache-maven-2.0.10-bin.zip

## 安装软件

### 安装 JDK

JDK 的安装,此处省略.

#### JDK 安装注意

JDK 安装完成后请务必设置 JAVA_HOME 环境变量，否则 maven2 无法运行

## 安装 eclipse

eclipse 安装此处省略

## 安装 Maven2

将刚才下载的包 apache-maven-2.0.10-bin.zip 解压到 D:\maven2 下,会出现如下目录结构

```
D:\maven2
     +--bin
     +--boot
     +--conf
     +--bin
```

设置环境变量，在系统环境变量 PATH 中加入 D:\maven2\bin，并且设置环境变量 M2_HOME=d:\maven2
开启命令行方式，输入

```
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

e:\Velcros\velcro7.prototype>*mvn*
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[ERROR] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO]

You must specify at least one goal. Try 'mvn install' to build or 'mvn --help' for options
See http://maven.apache.org for more information.



[INFO] ------------------------------------------------------------------------
[INFO] For more information, run Maven with the -e switch
[INFO] ------------------------------------------------------------------------
[INFO] Total time: < 1 second
[INFO] Finished at: Tue Mar 24 09:45:26 CST 2009
[INFO] Final Memory: 1M/2M
[INFO] ------------------------------------------------------------------------
```

## 配置 Maven2 使用公司内部插件仓库服务器

由于 Maven2 的安装包并不包括插件，所以在使用时会自动到网络上下载需要使用的插件以及依赖包等。将 Maven 配置为使用内部插件服务器，可以加快下载速度
Maven2 的配置文件放在如下位置 $用户目录/.m2/settings.xml 和 $M2_HOME/conf/settings.xml，查找循序为 先查找 $用户目录/.m2/settings.xml，然后查找 $M2_HOME/conf/settings.xml。

### 用户目录

Windows 环境下的用户目录一般为 C:\Documents and Settings\ [Login Name]

Linux 环境下的用户目录一般为/home/[Login Name]

用编辑器打开 settings.xml 文件，找到如下内容

```
<mirrors>
   <!-- mirror
    | Specifies a repository mirror site to use instead of a given repository. The repository that
    | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are used
    | for inheritance and direct lookup purposes, and must be unique across the set of mirrors.
    |
    <mirror>
      <id>mirrorId</id>
      <mirrorOf>repositoryId</mirrorOf>
      <name>Human Readable Name for this Mirror.</name>
      <url>http://my.repository.com/repo/path</url>
    </mirror>
   -->
</mirrors>
```

加入

```
<mirror>
     <id>visionsoft</id>
     <url>http://maven2-repo1.visionsoft.net/repo/</url>
     <mirrorOf>central</mirrorOf>
</mirror>
```

## 配置本地仓库位置

Maven 会将内部或外部服务器下载的插件和库文件放在本地电脑里。缺省位置为$用户目录/.m2/repository，为了能够更好的管理,我们一般调整本地仓库的位置。
打开 settings.xml，找到如下设置

```
<!-- localRepository
 | The path to the local repository maven will use to store artifacts.
 |
 | Default: ~/.m2/repository
<localRepository>/path/to/local/repo</localRepository>
-->
```

加入:

```
<localRepository>E:/maven2-repository</localRepository>
```

本文将本地仓库安装在 E:/maven2-repository 下，仓库可以安装在任何位置，只要工作方便即可

# 使用 Maven 编译一个测试项目

## 由 SVN 仓库中检出测试项目

可以使用任何的 SVN 客户端检出 http://svn.visionsoft.net/test-repo/trunk 下的工程到 E:/velcros/test-repo 下，在此直接使用 SVN 命令行。

$svn co http://svn.visionsoft.net/test-repo/trunk test-repo

```
A test-repo\testng.xml
A test-repo\src
A test-repo\src\test
A test-repo\src\test\java
A test-repo\src\test\java\com
A test-repo\src\test\java\com\velcro7
A test-repo\src\test\java\com\velcro7\base
A test-repo\src\test\java\com\velcro7\base\TestVelcroObject.java
A test-repo\src\main
A test-repo\src\main\java
A test-repo\src\main\java\com
A test-repo\src\main\java\com\velcro7
A test-repo\src\main\java\com\velcro7\base
A test-repo\src\main\java\com\velcro7\base\VelcroObject.java
A test-repo\src\main\java\com\velcro7\base\VelcroObjectType.java
A test-repo\src\main\java\com\velcro7\base\VelcroValidation.java
A test-repo\src\main\java\com\velcro7\base\VelcroUnionObject.java
A test-repo\src\main\java\com\velcro7\base\VelcroObjectID.java
A test-repo\src\main\java\com\velcro7\base\exception
A test-repo\src\main\java\com\velcro7\base\exception\VelcroObjectCannotRemoveException.java
A test-repo\src\main\java\com\velcro7\base\exception\VelcroObjectUnsupportTypeException.java
A test-repo\src\main\java\com\velcro7\base\exception\VelcroGeneralException.java
A test-repo\src\main\java\com\velcro7\base\exception\VelcroObjectInvalidNameException.java
A test-repo\src\main\java\com\velcro7\base\exception\VelcroObjectDuplicateException.java
A test-repo\src\main\java\com\velcro7\base\exception\VelcroObjectIllegalInheritException.java
A test-repo\src\main\java\com\velcro7\base\exception\VelcroObjectNotFoundException.java
A test-repo\pom.xml
```

取出版本 4。

## 编译、测试项目

在命令行方式下输入如下命令，进行编译，及测试

```
$cd test-repo
$mvn test
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building velcro7-base
[INFO] task-segment: [test]
[INFO] ------------------------------------------------------------------------
Downloading:
http://maven2-repo1.visionsoft.net/repo//org/apache/maven/plugins/maven-resources-plugin/2.3/maven-resources-plugin-
2.3.pom
4K downloaded

...

Downloading: http://maven2-repo1.visionsoft.net/repo//org/apache/maven/maven-artifact/2.0/maven-artifact-2.0.jar
76K downloaded
[INFO] Surefire report directory: E:\Velcros\test-repo\target\surefire-reports
-------------------------------------------------------
T E S T S
-------------------------------------------------------
Running TestSuite
PASSED: testProperty
PASSED: testCheckName
PASSED: testConstruct
PASSED: testToString

===============================================
base
Tests run: 4, Failures: 0, Skips: 0
===============================================

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.329 sec
Results :
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 49 seconds
[INFO] Finished at: Tue Mar 24 11:05:27 CST 2009
[INFO] Final Memory: 10M/18M
[INFO] ------------------------------------------------------------------------
```

至此，编译、测试完成

## 生成发布文件

使用 maven 生成发布可以采用如下命令

$mvn package

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building velcro7-base
[INFO] task-segment: [package]
[INFO] ------------------------------------------------------------------------
[INFO] [resources:resources]
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory E:\Velcros\test-repo\src\main\resource
[INFO] [compiler:compile]
[INFO] Nothing to compile - all classes are up to date
[INFO] [resources:testResources]
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory E:\Velcros\test-repo\src\test\resource
[INFO] [compiler:testCompile]
[INFO] Nothing to compile - all classes are up to date
[INFO] [surefire:test]
[INFO] Surefire report directory: E:\Velcros\test-repo\target\surefire-reports

-------------------------------------------------------
T E S T S
-------------------------------------------------------
Running TestSuite
PASSED: testProperty
PASSED: testCheckName
PASSED: testConstruct
PASSED: testToString

===============================================
base
Tests run: 4, Failures: 0, Skips: 0
===============================================

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.329 sec

Results :

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

[INFO] [jar:jar]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
```

```
[INFO] Total time: 4 seconds
[INFO] Finished at: Tue Mar 24 11:30:29 CST 2009
[INFO] Final Memory: 11M/22M
[INFO] ------------------------------------------------------------------------
```

然后再\target 目录下可以找到 velcro7-base.jar 文件。

## 安装 eclipse 的 maven 插件

要在 eclipse 中使用 maven，首先需要安装 maven 插件。可以使用 eclipse 的更新管理器，更新地址为
http://m2eclipse.sonatype.org/update/

## 生成 eclipse 配置文件

Maven2 可以直接生成 eclipse 的项目配置文件，通过这种方式，可以将项目的依赖关系等一起设定完成。
在项目目录下输入

```
$mvn eclipse:eclipse -DdownloadSources=true
```

打开 eclipse
选择 File->Import
项目类型选择 General\Maven Projects
在弹出对话框中:
Root Directory 中选择 E:\Velcros\test-repo
Projects 中选择 /pom.xml - com.velcryo7.base:velcro7-base:0.1-PROTOTYPE:jar

## 在 eclipse 中使用 maven

安装 eclipse 的 maven 插件后，可以在 eclipse 中直接使用菜单调用常用的 maven 命令。方法为

1. 选中要使用的项目
2. 单击 Run As 按钮，
3. 在弹出的对话框中选择要执行的 maven 动作即可
   如果需要的动作没有出现，也可以在 Run Configuration 中进行相应的配置。

# Maven2 基础教程(2) - 常用命令

## 目的

本文用以介绍一些在项目开发中常用的 maven2 命令，通过本文，您可以了解到

1. 建立一个新项目
2. 编译一个项目
3. 编译及测试一个项目
4. 生成文档
5. 生成项目站点
6. 发布项目

## Maven2 常用命令介绍

除了建立 maven2 项目外，其他所有的命令都需要在所在的项目目录下执行。

### 建立项目

建立项目的命令为

mvn archetype:generate -DarchetypeArtifactId=\[项目类型] -DgroupId=\[组识别] -DartifactId=\[项目名称] -Dversion=\[版本] -Dpackage=\[包]

项目类型(archetypeArtifactId)可以是下列选项

1. maven-archetype-j2ee-simple (J2EE 项目)
2. maven-archetype-portlet (门户)
3. maven-archetype-quickstart (一般 Java project)
4. maven-archetype-site (复杂站点)
5. maven-archetype-site-simple (简单站点)
6. maven-archetype-webapp (Java Web 站点)

groupId 用于项目的分组
artifactId 项目的名称，也就是包名称
version 版本
package 主要设定目录的层次
下面是一个例子

```
#mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart -DgroupId=com.velcro7.framework
-DartifactId=velcro7-base -Dversion=0.1-PROTOTYPE -Dpackage=com.velcro7.base
```

接下来会建立如下目录结构

```
velcro7-base
  + src
  | + main
  | | + java
  | |   + com
  | |     + velcro7
  | |       + base
  | |         + App.java
  | + test
  |   + java
  |     + com
  |        + velcro7
  |         + base
  |           + AppTest.java
  + pom.xml
```

## 编译项目

编译项目的命令为

```
#mvn compile
```

## 执行单元测试

执行单元测设的命令为

```
#mvn test
```

在执行单元测试前，首先会执行编译动作
测试报告放在 target\site\surefire-reports 下

## 生成文档

生成文档的命令为

```
#mvn javadoc:javadoc
```

生成的文档会在 target\site\apidocs 下

## 生成项目站点

生成项目站点的命令为

```
#mvn site:site
```

生成站点放在\target\site 下

### 清除项目生成文件

清除项目生成文件的命令为

```
#mvn clean:clean
```

### 打包文件

打包文件的命令为

```
#mvn package
```

生成的包文件在 target 目录下

### 将项目安装到本地仓库

安装到本地仓库的命令为

```
#mvn install
```

### 发布项目

发布项目的命令为

```
#mvn deploy
```

# Maven2 基础教程(3) - pom.xml 文件简介

## 目标

本文用以说明如何修改 maven2 的主要配置文件 pom.xml 在适应我们的项目需要，通过本文您可以了解到

1. 如何设定编译参数
2. 设定编译环境为 UTF-8 编码
3. 添加依赖项
4. 添加 TestNG 框架支持

# pom.xml 简介

如下是一个最基础的 pom.xml

```
<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.velcro7.framework</groupId>

  <artifactId>velcro7-base</artifactId>

  <packaging>jar</packaging>

  <version>0.1-PROTOTYPE</version>

  <name>velcro7-base</name>

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

  <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>3.8.1</version>

      <scope>test</scope>

    </dependency>

  </dependencies>

</project>
```

说明了项目的名称，以及依赖于 junit 的项目。接下来，我们要调整一下编译参数

# 修改 pom.xml

## 调整编译参数

编译参数，主要通过使用设定 maven-compile-plugin 来实现

我们加入如下配置信息

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
              <encoding>utf-8</encoding>
              <source>1.5</source>
              <target>1.5</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

如上，可以设定编译使用 UTF-8 编码，源码为 JDK1.5 的版本，目标也为 JDK1.5 的版本。

## 设定使用 UTF-8 编码

除了编译外，还有资源文件、javadoc 等都需要告诉 maven 使用 UTF-8 编码，我们可以设定如下两个插件

```
<plugin>
    <artifactId>maven-resources-plugin</artifactId>
    <configuration>
        <encoding>UTF-8</encoding>
    </configuration>
</plugin>
<plugin>
    <artifactId>maven-javadoc-plugin</artifactId>
    <configuration>
        <encoding>UTF-8</encoding>
    </configuration>
</plugin>
```

## 添加 TestNG 的测试框架支持

由于自动生成的项目为使用 JUnit 的测试框架，但是我们的项目使用 TestNG 的测试框架，所以我们需要调整一下项目的依赖关系，并且设定项目使用的 TesgNG 配置文件。
首先删除对于 JUnit 的依赖

```
<dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
</dependency>
```

然后加入如下内容

```
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>5.8</version>
    <scope>test</scope>
    <classifier>jdk15</classifier>
</dependency>
```

由于 TestNG 需要不同的包支持 JDK15 和 JDK14 所以在此我们要特别指定<classifier>属性。
如果设定了<version>属性，maven 会自动下载依赖项的对应版本，如果没有设置<version>属性，Maven 会自动下载最新版本。由于我们项目的开发周期比较长，所以需要指定版本，防止开发过程中由 Maven 自动更换我们使用的依赖库。
<scope>属性，设定了依赖项的使用范围。如果设定为 test 表示近测试时使用，在打包时不会打包该文件。
接下来，我们使用插件 maven-surfire-plugin 设定 testNG 的配置文件位置，如下

```
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
        <suiteXmlFiles>
            <suiteXmlFile>testng.xml</suiteXmlFile>
        </suiteXmlFiles>
    </configuration>
</plugin>
```

如上，表示使用 testng.xml 作为 testNG 的配置文件。

# 结束语

如上配置，我们最后的配置文件为

```
<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.velcro7.framework</groupId>
    <artifactId>velcro7-base</artifactId>
    <packaging>jar</packaging>
    <version>0.1-PROTOTYPE</version>
    <name>velcro7-base</name>
    <url>http://maven.apache.org</url>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-resources-plugin</artifactId>
```

```
            <configuration>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
        <plugin>
            <artifactId>maven-javadoc-plugin</artifactId>
            <configuration>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <encoding>utf-8</encoding>
                <source>1.5</source>
                <target>1.5</target>
            </configuration>
        </plugin>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <configuration>
                <suiteXmlFiles>
                    <suiteXmlFile>testng.xml</suiteXmlFile>
                </suiteXmlFiles>
            </configuration>
        </plugin>
    </plugins>
</build>
<dependencies>
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>5.8</version>
        <scope>test</scope>
     <classifier>jdk15</classifier>
    </dependency>
</dependencies>
</project>
```

# Maven2 基础教程(4) - 设置 Snapshot 和 Release 仓库

## 目标

本文说明

1. Snapshots 和 Release 仓库的区别
2. 如何设置 Snapshots 和 Release 仓库

## Snapshots 和 Release 仓库的区别

Snapshots 是快照仓库，用于保存开发过程中不稳定版本的程序包。 Release 仓库是保存发行版本的程序包的仓库。
如果模块的版本号最后带有-SNAPSHOT，则表示这是一个快照版本。在发布时，Maven2 会根据模块的版本号中时候带有-SNAPSHOT 来确定往那个仓库上传。
在本地编译是，Maven 会根据版本号来判断是否需要检测最新版本。如果某个模块的依赖模块的版本号中以-SNAPSHOT 结尾，则 Maven 会将 Snapshots 仓库中的程序包和本地仓库中的程序包进行比较。如果 Snapshots 仓库中比较新，会自动下载到本地仓库中。如果版本号中不以-SNAPSHOT 结尾，那么 Maven 不会和中央仓库比较，只要本地仓库中已经存在程序包，就不会上中央仓库下载

## Snapshots 和 Release 仓库的配置

首先在模块的 pom.xml 文件中加入如下定义

```
<distributionManagement>
        ...
        <repository>
            <id>VelcroRelease</id>
            <name>maven2.velcrosoft.net-releases</name>
            <url>http://maven2.velcrosoft.net/velcro-releases-local</url>
        </repository>
        <snapshotRepository>
            <id>VelcroSnapshot</id>
            <name>maven2.velcrosoft.net-snapshots</name>
            <url>http://maven2.velcrosoft.net/velcro-snapshots-local</url>
        </snapshotRepository>
        ...
    </distributionManagement>
```

repository 元素设定 Release 仓库 snapshotRepository 元素设定 Snapshot 仓库

接下来在本地的 setting.xml 中加入相应的仓库定义

```
<servers>
    ....
 <server>
```

```
 <id>VelcroSnapshot</id>
 <username>username</username>
 <password>pasword</password>
</server>
<server>
 <id>VelcroRelease<id>
 <username>username</username>
 <password>password</password>
</server>
   ...
 </servers>
```

注意：pom.xml 和 setting.xml 中的服务器 ID 要一致。

如果 Maven2 仓库设定了要使用用户名和密码访问，可以在 server 元素中添加 username 元素和 password 元素来设定。