**项目主页：**[http://www.gradle.org](http://www.gradle.org/)

**下载地址：**<http://services.gradle.org/distributions/gradle-1.5-all.zip>

**安装：**

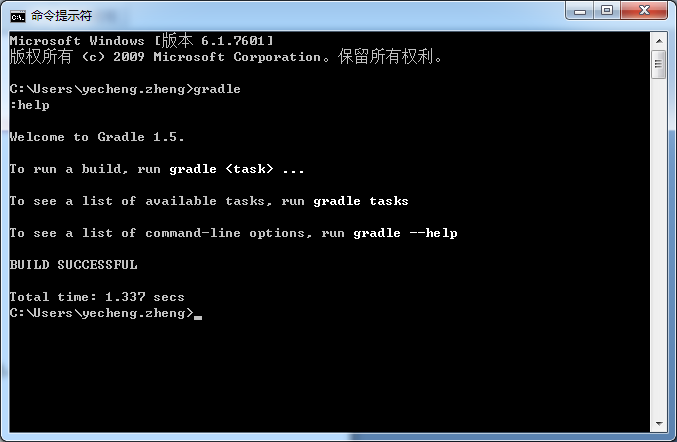
1. 解压到自定义目录（例如：C:\gradle-1.5）
2. 设置环境变量

GRADLE\_HOME = C:\gradle-1.5

Path += %GRADLE\_HOME%\bin

GRADLE\_OPTS = -Dfile.encoding=UTF-8

1. 打开命令行输入：gradle，显示以下类似信息即表示安装成功



**命令：**

1. 编译命令：gradle cleaneclipse eclipse -Pprofile=test
2. 打包命令：gradle clean war -Pprofile=product
3. 一些常用命令：

-b 指定build文件（默认为：build.gradle）

-c 指定settings文件（默认为：settings.gradle）

-x 排除指定任务（例如：-x test）

更多常用命令请输入gradle -h/help查看

加源码：

在build.gradle中加入以下代码：

sourceSets.main.java.srcDirs "gen-java"

web工程：

在build.gradle中加入以下代码：

apply plugin: 'war'

=============

**一些小技巧**

gradle cleanEclipse  eclipse 可以简写为  gradle  cE e

11.2 排除任务

你可以使用 -x参数来排除不需要执行的任务,让我们用上一个示例中的脚本来体验一下

> gradle dist -x test

:compile

compiling source

:dist

building the distribution

BUILD SUCCESSFUL

Total time: 1 secs

可以看到,test任务并没有被执行,即使它被dist任务所依赖,同时test任务所依赖的compileTest也同样没有被执行,

而像complie所依赖的除了test之外的任务仍然正常执行

====================================================

11.3 持续构建-即使发生错误

默认情况下,Gradle在执行时如果发生错误会立即中止执行任务,这能使构建更快完成但无法查看更多其它的失败信息,

此时可以使用

-continue

参数,在一个构建中尽可能捕获更多的失败信息;

当增加此参数是,gradle会在其所依赖的任务成功执行的前提下独立的执行每个任务;例如,如果你使用了

'java'和'checkStyle'插件,即使你的代码规范检查和单元测试失败,gradle仍然可以运行checkStyle,执行单元测试并且构建javadoc;所有的错误都会在

P.S:只有其所依赖的任务成功执行之后任务才会被执行,所以如果当java代码编译失败时,单元测试任务即不会被执行,因为

test任务直接依赖于compileJava任务

=================================================================

11.4 任务名称简写

在执行任务时,只要键入的字符足以唯一区分出一个任务即可省去其它字符.如11.1示例

执行dist任务 输入gradle di即可

驼峰命名式简写.如11.1示例

要执行compileTest任务只需输入 gradle cT即可 当然输入 gradle cmopTest同样可以执行构建

===

**gradle如何在打jar包的时候隔离掉某些文件和目录?**

经过试验， build.gradle里的写法如下：

 jar {

                 from sourceSets.main.resources

                exclude('domesticPnr/\*\*' ,  'pnrconf/\*\*/\*.xml' , "pnrs/\*\*/\*.txt",

    'dotnet/\*\*' , 'espeed' , 'espeed','issue-ticket','testpnr'

                )

}

====

如何引用到某个一个版本的jar， 而不是源码引用?

出于某种原因，（ 固定版本号， 或者你的新项目不在 tz根目录下）， 你会需要用jar包来引用  tz-common里的项目， 做法如下：

首先， 在 tz\tz-common\framework-mongo-support目录下的 gradle.build文件中加入版本号

version = "1.0-SNAPSHOT"

//group = 'com.travelzen'

如果加上这行group， 则可以改变group名

这时候， 你可以运行

gradle  :tz-common:framework-mongo-support:install  -i

把jar包install到你的gradle本地存储。

然后， 在你的新项目的根 build.gradle目录中加入

repositories {

  mavenCentral()

  mavenLocal()

}

主要是需要 mavenLocal(）， 这样才能引用到之前发布的jar包。

然后， 再在你的新项目中

 compile('tz.tz-common:framework-mongo-support:1.0-SNAPSHOT'){

 changing=true

   }

再执行  gradle  eclipse

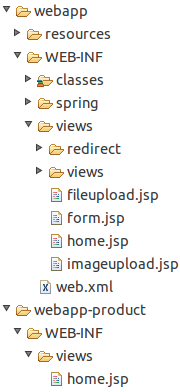
就能够引用到  framework-mongo-support-1.0-SNAPSHOT.jar了

如果你要把jar包发布到私服上， 可以参考以下步骤：

<http://www.gradle.org/docs/current/userguide/maven_plugin.html>

发布私服需要 nexus的用户名密码， 这个承文那边有。

**在总build.gradle中加入了webapp-(profile)的支持**

    afterEvaluate {  
        if (project.plugins.findPlugin('war') && project.hasProperty('profile')) {  
            webapp = project.webAppDirName  
            webtmp = webapp + '-tmp'  
            war.doFirst {  
                copy {  
                    from "$projectDir/" + webapp  
                    into "$projectDir/" + webtmp  
                }  
                copy {  
                    from "$projectDir/" + webapp + '-' + project.profile  
                    into "$projectDir/" + webtmp  
                }  
                project.webAppDirName = webtmp  
            }  
            war.doLast {  
                ant.delete dir: webtmp  
            }  
        }  
    }  
  
  
与resources-(profile)的含义略有不同，webapp-(profile)里的内容只在打war包的时候起作用，平时编译 eclipse是没有影响的  
  
功能：在打war包的时候把webapp与webapp-(profile)里的内容拷贝到临时的webapp-tmp目录里，如果有同名文件 webapp-(profile)将覆盖webapp的内容，然后以webapp-tmp作为新的webapp打成war包  
  
例如：假设项目有webapp和webapp-product两个webapp目录，当用命令gradle (projectPath):war -Pprofile=product打war包的时候，webapp-product里的home.jsp将覆盖webapp里的同名文件  
  
  
注意事项：由于其他webapp目录的文件无法通过eclipse命令直接进行替换测试，请自行对这些文件测试保证其正确性