Maven 引入外部依赖 Maven 项目文档

Maven 项目模板

Maven 使用 archetype(原型) 来创建自定义的项目结构，形成 Maven 项目模板。

在前面章节我们学到 Maven 使用下面的命令来快速创建 java 项目：

```
mvn archetype:generate
```

什么是 archetype ?

archetype 也就是原型，是一个 Maven 插件，准确说是一个项目模板，它的任务是根据模板创建一个项目结构。我们将使用 quickstart 原型插件创建一个简单的 java 应用程序。

使用项目模板

让我们打开命令控制台，跳转到 C:\> MVN 目录并执行以下 mvn 命令:

```
C:\MVN> mvn archetype:generate
```

Maven 将开始处理，并要求选择所需的原型:

```
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'archetype'.
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Default Project
[INFO]task-segment: [archetype:generate] (aggregator-style)
[INFO] ------------------------------------------------------------------------
[INFO] Preparing archetype:generate
...
600: remote -> org.trailsframework:trails-archetype (-)
601: remote -> org.trailsframework:trails-secure-archetype (-)
602: remote -> org.tynamo:tynamo-archetype (-)
603: remote -> org.wicketstuff.scala:wicket-scala-archetype (-)
604: remote -> org.wicketstuff.scala:wicketstuff-scala-archetype
Basic setup for a project that combines Scala and Wicket,
depending on the Wicket-Scala project.
Includes an example Specs test.)
605: remote -> org.wikbook:wikbook.archetype (-)
606: remote -> org.xaloon.archetype:xaloon-archetype-wicket-jpa-glassfish (-)
607: remote -> org.xaloon.archetype:xaloon-archetype-wicket-jpa-spring (-)
608: remote -> org.xwiki.commons:xwiki-commons-component-archetype
(Make it easy to create a maven project for creating XWiki Components.)
609: remote -> org.xwiki.rendering:xwiki-rendering-archetype-macro
```

```
(Make it easy to create a maven project for creating XWiki Rendering Macros.)
610: remote -> org.zkoss:zk-archetype-component (The ZK Component archetype)
611: remote -> org.zkoss:zk-archetype-webapp (The ZK wepapp archetype)
612: remote -> ru.circumflex:circumflex-archetype (-)
613: remote -> se.vgregion.javg.maven.archetypes:javg-minimal-archetype (-)
614: remote -> sk.seges.sesam:sesam-annotation-archetype (-)
Choose a number or apply filter
(format: [groupId:]artifactId, case sensitive contains): 203:
```

按下 **Enter** 选择默认选项 (203:maven-archetype-quickstart)。

Maven 将询问原型的版本

```
Choose org.apache.maven.archetypes:maven-archetype-quickstart version:
1: 1.0-alpha-1
2: 1.0-alpha-2
3: 1.0-alpha-3
4: 1.0-alpha-4
5: 1.0
6: 1.1
Choose a number: 6:
```

按下 **Enter** 选择默认选项 (6:maven-archetype-quickstart:1.1)

Maven 将询问项目细节。按要求输入项目细节。如果要使用默认值则直接按 Enter 键。你也可以输入自己的值。

```
Define value for property 'groupId': : com.companyname.insurance
Define value for property 'artifactId': : health
Define value for property 'version': 1.0-SNAPSHOT
Define value for property 'package': com.companyname.insurance
```

Maven 将要求确认项目细节，按 **Enter** 或按 Y

```
Confirm properties configuration:
groupId: com.companyname.insurance
artifactId: health
version: 1.0-SNAPSHOT
package: com.companyname.insurance
Y:
```

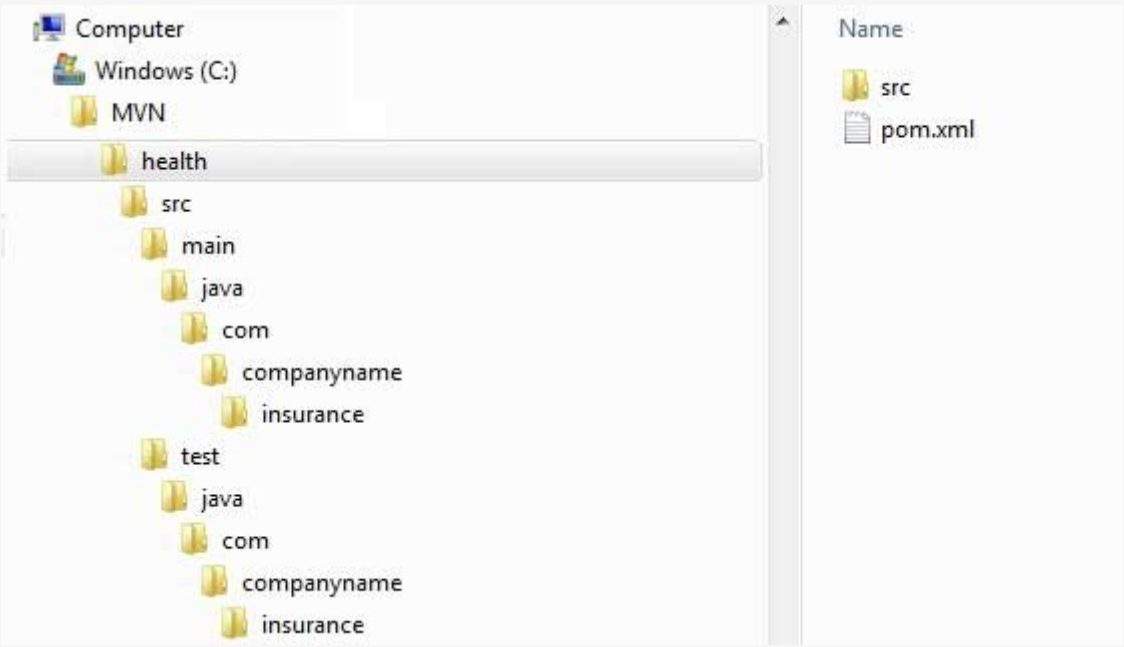
现在 Maven 将开始创建项目结构，显示如下:

```
[INFO] ------------------------------------------------------------------------
[INFO] Using following parameters for creating project
from Old (1.x) Archetype: maven-archetype-quickstart:1.1
```

```
[INFO] ----------------------------------------------------------------------
[INFO] Parameter: groupId, Value: com.companyname.insurance
[INFO] Parameter: packageName, Value: com.companyname.insurance
[INFO] Parameter: package, Value: com.companyname.insurance
[INFO] Parameter: artifactId, Value: health
[INFO] Parameter: basedir, Value: C:\MVN
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\MVN\health
[INFO] ----------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ----------------------------------------------------------------------
[INFO] Total time: 4 minutes 12 seconds
[INFO] Finished at: Fri Jul 13 11:10:12 IST 2012
[INFO] Final Memory: 20M/90M
[INFO] ----------------------------------------------------------------------
```

创建的项目

现在转到 C:\ > MVN 目录。你会看到一个名为 health 的 java 应用程序项目，就像在项目创建的时候建立的 artifactId 名称一样。 Maven 将创建一个有标准目录布局的项目，如下所示:



创建 pom.xml

Maven 为项目自动生成一个 pom.xml文件，如下所示:

```
<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/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.insurance</groupId>
<artifactId>health</artifactId>
```

```
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
<name>health</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>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

App.java

Maven 会自动生成一个测试的 java 文件 App.java。

路径：**C:\MVN\consumerBanking\src\main\java\com\companyname\bank**

```
package com.companyname.insurance;
/**
* Hello world!
*
*/
public class App
{
public static void main( String[] args )
{
System.out.println( "Hello World!" );
}
}
```

AppTest.java

Maven 会自动生成一个 java 文件 AppTest.java。

路径为： **C:\MVN\consumerBanking\src\test\java\com\companyname\bank**

```
package com.companyname.insurance;
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
/**
* Unit test for simple App.
*/
public class AppTest
extends TestCase
{
/**
* Create the test case
```

```
*
* @param testName name of the test case
*/
public AppTest( String testName )
{
super( testName );
}
/**
* @return the suite of tests being tested
*/
public static Test suite()
{
return new TestSuite( AppTest.class );
}
/**
* Rigourous Test :-)
*/
public void testApp()
{
assertTrue( true );
}
}
```

就这样。现在你可以看到 Maven 的强大之处。你可以用 maven 简单的命令创建任何类型的项目，并且可以启动您的开发。

Maven 引入外部依赖 Maven 项目文档