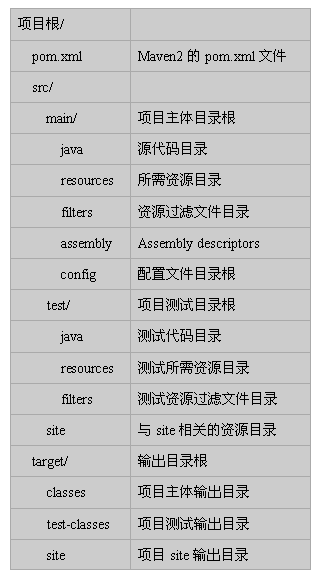
# Maven目录结构

好的目录结构可以使开发人员更容易理解项目，为以后的维护工作也打下良好的基础。Maven2根据业界公认的最佳目录结构，为开发者提供了缺省的标准目录模板。Maven2的标准目录结构如下：



|  |  |
| --- | --- |
| src/main/java | Application/Library sources |
| src/main/resources | Application/Library resources |
| src/main/filters | Resource filter files |
| src/main/assembly | Assembly descriptors |
| src/main/config | Configuration files |
| src/main/scripts | Application/Library scripts |
| src/main/webapp | Web application sources |
| src/test/java | Test sources |
| src/test/resources | Test resources |
| src/test/filters | Test resource filter files |
| src/site | Site |
| LICENSE.txt | Project's license |
| NOTICE.txt | Notices and attributions required by libraries that the project depends on |
| README.txt | Project's readme |

使用目录模板，可以使 pom.xml 更简洁。因为 Maven2 已经根据缺省目录，预定义了相关的动作，而无需人工的干预。以 resources 目录为例：

* src/main/resources，负责管理项目主体的资源。在使用Maven2执行compile之后，这个目录中的所有文件及子目录，会复制到target/classes目录中，为以后的打包提供了方便。
* src/test/resources，负责管理项目测试的资源。在使用Maven2执行test-compile之后，这个目录中的所有文件及子目录，会复制到target/test-classes目录中，为后续的测试做好了准备。

这些动作在 Maven1 中，是需要在 maven.xml 中使用<preGoal>或<postGoal>来完成的。如今，完全不需要在pom.xml中指定就能够自动完成。在src和 test都使用resources，方便构建和测试，这种方式本就已是前人的经验。通过使用Maven2，使这个经验在开发团队中得到普及。

创建标准目录模板，可以通过如下命令：

mvn archetype:create -DgroupId=com.codeline.commons -DartifactId=codelineCommons

groupId和artifactId的含义与Maven1中的含义一样，参数artifactId的值会作为项目根目录的名字。除了建立相应的目录之外，Maven2还会创建缺省的pom.xml。

Maven2也考虑到：不同类型的项目需要拥有不同的目录结构。如创建web项目，可以使用命令：

mvn archetype:create -DgroupId=com.mycompany.app

-DartifactId=my-webapp

-DarchetypeArtifactId=maven-archetype-webapp

**Maven 生命周期**

Maven生命周期已经在另一篇博客中介绍过了(<http://www.cnblogs.com/haippy/archive/2012/07/04/2576453.html>)，这里引用IBM developerworks 的文章再一次讨论Maven 的生命周期。

在Maven2中有了明确的生命周期概念，而且都提供与之对应的命令，使得项目构建更加清晰明了。主要的生命周期阶段：

* validate，验证工程是否正确，所有需要的资源是否可用。
* compile，编译项目的源代码。
* test-compile，编译项目测试代码。
* test，使用已编译的测试代码，测试已编译的源代码。
* package，已发布的格式，如jar，将已编译的源代码打包。
* integration-test，在集成测试可以运行的环境中处理和发布包。
* verify，运行任何检查，验证包是否有效且达到质量标准。
* install，把包安装在本地的repository中，可以被其他工程作为依赖来使用
* deploy，在整合或者发布环境下执行，将最终版本的包拷贝到远程的repository，使得其他的开发者或者工程可以共享。
* generate-sources，产生应用需要的任何额外的源代码，如xdoclet。

如果要执行项目编译，那么直接输入：mvn compile即可，对于其他的阶段可以类推。阶段之间是存在依赖关系（dependency）的，如test依赖test-compile。在执行 mvn test时，会先运行mvn test-compile，然后才是mvn test。