Maven 系列课程

时间: 2017/12/12
作者:袁毅雄
邮箱: 896778954@qq.com

目录

初始 Maven

Maven 的由来

Maven 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。

Maven 除了以程序构建能力为特色之外，还提供高级项目管理工具。由于 Maven 的缺省构建规则有较高的可重用性，所以常常用两三行 Maven 构建脚本就可以构建简单的项目。由于 Maven 的面向项目的方法，许多 Apache Jakarta 项目发文时使用 Maven，而且公司项目采用 Maven 的比例在持续增长。

Maven 这个单词来自于意第绪语（犹太语），意为知识的积累，最初在 Jakata Turbine 项目中用来简化构建过程。当时有一些项目（有各自 Ant build 文件），仅有细微的差别，而 JAR 文件都由 CVS 来维护。于是希望有一种标准化的方式构建项目，一个清晰的方式定义项目的组成，一个容易的方式发布项目的信息，以及一种简单的方式在多个项目中共享 JARs

项目架构

单一架构

垂直架构

Maven 运行原理

远程仓库/本地仓库

图解：

地址：

全球中央仓库：

https://repo.maven.apache.org/maven2
http://mvnrepository.com/
https://search.maven.org/

国内镜像仓库/私服：

http://maven.aliyun.com/nexus/content/groups/public

Maven 的安装

环境变量配置

Maven 的配置

settings.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!--本地中央仓库-->
  <localRepository> </localRepository>

  <!--Maven  插件-->
  <pluginGroups></pluginGroups>

  <!--Maven  代理服务器-->
  <proxies></proxies>

  <!--Maven  私服连接信息-->
  <servers></servers>

  <!--Maven  私服/镜像服务器/远程中央仓库-->
  <mirrors></mirrors>

  <!--Maven  插件-->
  <profiles></profiles>
</settings>
```

阿里巴巴的仓库配置

```
<!--阿里巴巴远程仓库-->
```

```
    <mirror>
      <id>nexus-aliyun</id>
      <mirrorOf>*</mirrorOf>
      <name>Nexus aliyun</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public</url>
    </mirror>
```

jdk 配置

```
<!--JDK 1.8-->
    <profile>
      <id>jdk-1.8</id>
      <activation>
        <activeByDefault>true</activeByDefault>
        <jdk>1.8</jdk>
      </activation>
      <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
      </properties>
      <repositories>
        <repository>
          <id>nexus</id>
          <name>local private nexus</name>
          <url>http://maven.oschina.net/content/groups/public/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>nexus</id>
          <name>local private nexus</name>
          <url>http://maven.oschina.net/content/groups/public/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
```

```
            <snapshots>
              <enabled>false</enabled>
            </snapshots>
          </pluginRepository>
        </pluginRepositories>
      </profile>
```

Maven 常用的命令

maven 命令	作用
mvn archetype:generate	创建 Maven 项目
mvn compile	编译源代码
mvn deploy	发布项目
mvn test-compile	编译测试源代码
mvn test	运行应用程序中的单元测试
mvn site	生成项目相关信息的网站
mvn clean	清除项目目录中的生成结果
mvn package	根据项目生成的 jar
mvn install	在本地 Repository 中安装 jar
mvn eclipse:eclipse	生成 eclipse 项目文件
mvn jetty:run	启动 jetty 服务
mvn tomcat:run	启动 tomcat 服务
mvn clean package -Dmaven.test.skip=true	清除以前的包后重新打包，跳过测试类
mvn dependency:copy-dependencies -DoutputDirectory=lib	导出依赖包到项目的 lib 下

Maven 项目配置

项目骨架

```
src
    -main
        -java
            -package      主代码
        -resource
            -源码配置
    -test
        -java
            -package      测试代码
```

```
                -resource
                    -测试配置
```

maven 坐标体系

组织名：groupId
模块名：artifactId
版　本：version：SNAPSHOT/ RELEASE

maven 依赖作用范围

test	只在项目测试的时候有依赖
compile	会在编译的时候有依赖
runtime	会在运行的时候有依赖
provided	编译有效,运行无效　servlet-api

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/xsd/maven-4.0.0.xsd">

    <groupId>com.bdqn.whhpit</groupId>
    <artifactId>SSM2R</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

</project>
```

Maven 生命周期

Maven 生命周期是为了所有的构建过程进行抽象和统一。Maven 从大量的项目和构建工具中学习和反思,总结了一套高度完善、易扩展的生命周期。这个生命周期包含了项目的清理、初始化、编译、测试、打包、集成测试、验证、部署和站点生成等几乎所有构建步骤

clean(清理)

cleanup(清理所有) 此生命周期旨在给工程做清理工作，它主要包含以下阶段：

pre-clean - 执行项目清理前所需要的工作。

clean - 清理上一次 build 项目生成的文件。

post-clean - 执行完成项目清理所需的工作.

default(默认)

validate - 验证项目是否正确且所有必要的信息都可用。

initialize - 初始化构建工作，如：设置参数，创建目录等。

generate-sources - 为包含在编译范围内的代码生成源代码.

process-sources - 处理源代码, 如过滤值.

generate-resources -

process-resources - 复制并处理资源文件，至目标目录，准备打包。

compile - 编译项目中的源代码.

process-classes - 为编译生成的文件做后期工作, 例如做 Java 类的字节码增强.

generate-test-sources - 为编译内容生成测试源代码.

process-test-sources - 处理测试源代码。

generate-test-resources -

process-test-resources - 复制并处理资源文件，至目标测试目录。

test-compile - 将需测试源代码编译到路径。一般来说，是编译/src/test/java 目录下的 java 文件至目标输出的测试 classpath 目录中。

process-test-classes -

test - 使用合适的单元测试框架运行测试。这些测试代码不会被打包或部署。

prepare-package -

package - 接受编译好的代码，打包成可发布的格式，如 JAR 。

pre-integration-test -

integration-test - 按需求将发布包部署到运行环境。

post-integration-test -

verify -
install -将包安装到本地仓库，给其他本地引用提供依赖。
deploy -完成集成和发布工作，将最终包复制到远程仓库以便分享给其他开发人员。

site(站点)

pre-site - 执行一些生成项目站点前的准备工作。
site - 生成项目站点的文档。
post-site - 执行需完成站点生成的工作，如站点部署的准备工作。
site-deploy - 向制定的 web 服务器部署站点生成文件。[2]

Eclipse 中使用 Maven

IntelliJ IDEA 中使用 Maven