Maven教程

> Maven 翻译为"专家"、"内行"，是 Apache 下的一个纯 Java 开发的开源项目。基于项目对象模型（缩写：POM）概念，Maven利用一个中央信息片断能管理一个项目的构建、报告和文档等步骤。 Maven 是一个项目管理工具，可以对 Java 项目进行构建、依赖管理。 Maven 也可被用于构建和管理各种项目，例如 C#，Ruby，Scala 和其他语言编写的项目。Maven 曾是 Jakarta 项目的子项目，现为由 Apache 软件基金会主持的独立 Apache 项目。
>
> 老师笔记加上自己的一点课堂笔记，如有错误请联系QQ：734229011进行修改：）😁

一，分布式

- 传统项目部署：
  - 新建一个项目，通过不同的包区分不同的模块
  - 把这一个项目发布服务器的 tomcat 中
- 分布式项目部署(适用于高负载情况下)
  - 把一个完整的项目拆分成多个项目,把拆分后的项目分别部署到对应的服务器(tomcat)中的过程就叫做分布式项目部署
  - 把传统项目包换成一个单独的项目

二，Ant

1. Ant 和 Maven 都是项目构建工具
2. 在项目的运行编辑打包等过程都需要依赖于 Ant 构建工
3. eclipse 默认使用的是 Ant，项目根目录下的 `.project` 文件即为 Ant 的配置文件

三，Maven

Maven简介

- 基于 Ant 的构建工具，Ant 有的功能 Maven 都有，额外添加了其他功 能
- 本地仓库：计算机中一个文件夹，自己定义是哪个文件夹
- 中央仓库：网上地址 https://repo1.maven.org/maven2/ (下载速度慢 -> 配置国内镜像)
- 保证 JDK 版本和开发环境一致，如果不配置 1.4 或 1.5
- 坐标
  - 每一 jar 文件都有一个唯一坐标，通过坐标可以精确确定是哪 个 jar
  - 坐标组成

Group ID: 公司名.公司网址倒写

Artifact ID: 项目名

Version: 版本

- pom
  - 英文名称(Project Object Mode)
  - 项目对象模型

    把 project 当作对象看待

    通过 Maven 构建工具可以让对象(项目)和对象(项目)之 间产生关系
- 网址(查看定位坐标)

  http://mvnrepository.com

Maven初始化

1. eclipse 环境默认集成 Maven
2. eclipse (开启，修改) Maven 版本和版本切换：Window -> Preferences -> Maven -> Installations (勾选相应版本即可，如果要额外添加其它版本点击 add)
3. settings.xml (Window -> Preferences -> Maven -> User Settings) 中需要配置的内容
   1. 设置本地仓库(示例语法)

      ```
      <localRepository>D:/maven/r2/myrepository</localRepository>
      ```

   2. 修改 JDK 版本保证 JDK 版本和开发环境一致，如果不配置 1.4 或 1.5(JDK1.7 示例)

      ```
      <profile>
          <id>jdk-1.7</id>
          <activation>
              <activeByDefault>true</activeByDefault>
              <jdk>1.7</jdk>
          </activation>
          <properties>
              <maven.compiler.source>1.7</maven.compiler.source>
              <maven.compiler.target>1.7</maven.compiler.target>
              <maven.compiler.compilerVersion>1.7</maven.compiler.compilerVersion>
          </properties>
      </profile>
      ```

   3. 修改镜像地址(不使用 nexus 时配置)(默认国外镜像，速度慢，修改为阿里云镜像)

      ```
      <mirror>
          <id>alimaven</id>
          <name>aliyunmaven</name>
          <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
          <mirrorOf>central</mirrorOf>
      </mirror>
      ```

创建Maven项目

1. New -> Maven Project
2. 勾选复选框(Create a simple project(skip archetype selection))，表示创建一个简单的 Maven 项目(不应用任何模板)
3. 填写项目信息

   Artifact

   - Group Id：一般公司名.公司网址倒写(例如：com.lls)
   - Artifact Id：项目名(例如：test)
   - Version：版本，`0.0.1-SNAPSHOT` 为 0.0.1 快照版，可以忽略，也可以修改为版本(`0.0.1`)，因为一般快照版(快照机制)只在私服的时候有作用
   - Packaging：项目的类型,最终会被打包成什么类型

     jar：java 项目

     war：web 项目

     pom：逻辑父项目，只要一个项目有子项目必须是 pom 类型
   - Name：项目名字，一般只在远程发布的时候有作用(一般忽略)
   - Description：项目描述信息，一般只在远程发布的时候有作用(一般忽略)

   Parent Project：一般只在创建子项目的时候才需要配置

   Advanced：一般忽略

Maven项目目录结构(jar 类型)

- `src/main/java` ： 真实目录(src/main/java)的快捷目录，写 java 代码
- `src/main/resources` ： 真实目录(src/main/resources)的快捷目录
  - 存放配置文件
  - 虽然看见 resources 但是里面所有配置文件最终会被编译放入到 classes 类路径(java 项目会被编译放入项目根目录)
- `src/test/java` ： 写测试 java 代码(无关紧要)
- `src/test/resources` ： 测试的配置文件夹(无关紧要)
- pom.xml：Maven 的配置文件

  当前项目所依赖的其他项目或 jar 或插件等

注意：无论是 java 视图还是 javaee 视图，src/main/java 下都有 main 和 resources ，只是 java 视图看不见(默认隐藏)

Maven项目之间的关系

- 依赖关系
  - 标签 `<dependency>` 把另一个项目的 jar 引入到当过前项目
  - 自动下载另一个项目所依赖的其它项目
- 继承关系
  1. 父项目是 pom 类型

2. 子项目jar或war，如果子项目还是其他项目的父项目，子项目也 是 pom 类型
3. 有继承关系后，子项目中出现 `<parent>` 标签

   如果子项目和 `<groupId>` 和 `<version>` 与父项目项目，在子项目中可以不配置 `<groupId>` 和 `<version>`
4. 父项目 pom.xml 中是看不到有哪些子项目，在逻辑上具有父子 项目关系.

```
<parent>
    <groupId>com.bjsxt</groupId>
    <artifactId>parent</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</parent>
```

- 聚合关系
  1. 前提是继承关系，父项目会把子项目包含到父项目中
  2. 子项目的类型必须是 MavenModule 而不是 mavenproject
  3. 新建聚合项目的子项目时，点击父项目右键新建 Maven Module
  4. 具有聚合关系的父项目，在 pom.xml 中 `<modules>`

```
<modules>
    <module>child2</module>
</modules>
```

  5. 具有聚合关系的子项目，在 pom.xml 中 `<parent>`

```
<parent>
    <groupId>com.bjsxt</groupId>
    <artifactId>parent</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</parent>
```

- 聚合项目和继承项目区别
  - 在语意上聚合项目父项目和子项目关系性较强
  - 在语意上单纯继承项目父项目和子项目关系性较弱
- `<dependencyManagement>` 写在父项目
  - 作用：声明可能使用到的所有 jar
  - 子项目中只需要有坐标的 `<groupid>` 和 `<artifactid>`， `<version>` 继承父项目
  - 在父项目中 `<properties>` 把所有版本好进行统一管理
  - 父项目 pom.xml

    `<properties>` 子标签名称自定义

    ${名字} 引用标签的值

```
<properties>
    <spring-version>4.1.6.RELEASE</spring-version>
</properties>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring-version}</version>
        </dependency>
    </dependencies>
</dependencyManagement>
```

- 子项目

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
    </dependency>
</dependencies>
```

创建 war 类型项目

1. 创建 mavenproject 时选择 packaging 为 war
2. 在 webapp (类似普通项目的 webContent 文件夹) 文件夹下新建 META-INF 和 WEB-INF/web.xml (不需要创建 lib 因为Maven 项目的依赖包统一由 pom.xml 进行管理)
3. 在 pom.xml 中添加 javaee 相关的三个 jar (基础的 web 包)
    - `<scope>` jar有效范围 provided 表示编译期生效，不会打包发布到 tomcat 中，因为 tomcat 中默认有，打包过去的话会有冲突

```
<dependencies>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.0.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>2.2</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>

    </dependency>
```

```
</dependencies>
```

4. 配置 tomcat 插件(Maven 从仓库加载的一个独立 tomcat)，父项目中 `<plugins>` 也可以用 `<pluginManagement>` 括起来表示不立即引入到子项目中，子项目如果需要的话可以再单独引入，子项目引入时的版本号标签要去掉，也起到了父项目统一管理子项目插件版本的目的

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.2</version>
            <configuration>
                <!-- 控制 tomcat 端口号 -->
                <port>80</port>
                <!-- 项目发布到 tomcat 后的名称 -->
                <!-- / 相当于把项目发布名称为 ROOT -->
                <!-- /abc -->
                <path>/</path>
            </configuration>
        </plugin>
    </plugins>
</build>
```

5. 右键项目 -> run as -> maven build -> Goals 中输入

   `clean tomcat7:run`

   注意：不注明版本 7 默认是版本 6

资源拷贝插件

1. maven 默认只把 src/main/resources 里面的非.java 文件进行编译到 classes 中，而 src/main/java 下的 Mapper.xml 或者其它配置文件就会解析不到，只配置 src/main/java 的解析的话，src/main/resource 又会解析不到，所以需要两个(存在配置文件的目录)全部配置
2. 如果希望 src/main/java 下的文件也被编辑到 classes 中,在 pom.xml 中配置(下面内容配置在 `<build>` 中)

```
<resources>
    <resource>
        <directory>src/main/java</directory>
        <includes>
            <include>**/*.xml</include>
        </includes>
    </resource>
    <resource>
        <directory>src/main/resources</directory>
        <includes>
            <include>**/*.xml</include>
            <include>**/*.properties</include>
        </includes>
    </resource>
</resources>
```

热部署

简介：通过热部署配置，可以便捷地把本地项目发布到远程的 tomcat (发布会不用重启 tomcat 就能看到效果)

1. 修改 tomcat/conf/tomcat-users.xml 添加 tomcat 角色(确保 tomcat 中有自带的那个 manager 项目)

```
<rolerolename="manager-gui"/>
<rolerolename="manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui,manager-script"/>
```

2. 在 Maven 项目的 pom.xml 中 tomcat 插件的 `<configuration>` 里配置

```
<plugins>
    <plugin>
        <groupId>org.apache.tomcat.maven</groupId>
        <artifactId>tomcat7-maven-plugin</artifactId>
        <version>2.2</version>
        <configuration>
            <!-- 控制 tomcat 端口号 指 Maven 里的 tomcat 端口，热部署时这项不生效 -->
            <port>80</port>
            <!-- 项目发布到 tomcat 后的名称 -->
            <!-- / 相当于把项目发布名称为 ROOT -->
            <!-- /abc -->
            <path>/jqk</path>
            <username>tomcat</username>
            <password>tomcat</password>
            <url>http://192.168.139.128:8080/manager/text</url>
        </configuration>
    </plugin>
</plugins>
```

3. 右键项目 -> run as -> maven build (以前写过,选择第二个) -> 输入
    1. `tomcat7:deploy` ： 第一次发布(目标 tomcat 没有同名项目)
    2. `tomcat7:redeploy` ： 非第一次发布(目标 tomcat 有同名项目，会覆盖原来的同名项目)

打包项目成war包

1. 右键要打包的项目，选择 Run As -> Maven install (也可以先运行 Maven clean 命令清一下缓存)
2. 控制台提示打包成功后，会显示打包后 war 包的路。径地址
3. 刷新工程，在 target 中发现打包好的 war 包，并且复制 war 包，放入tomcat 的 webapps 中，因为 war 包的名字过长，放入 webapp 时可以修改其名字

补充知识

当 `@RequestMapping("/")` 控制器注解无法进入时，可尝试在 web.xml 中做如下配置

```
<welcome-file-list>
    <welcome-file></welcome-file>
</welcome-file-list>
```

强行解释：不配置欢迎列表内容，项目默认走 index.jsp ，所以需要配置欢迎页，把默认页改为空，这样根路径请求才能走控制器