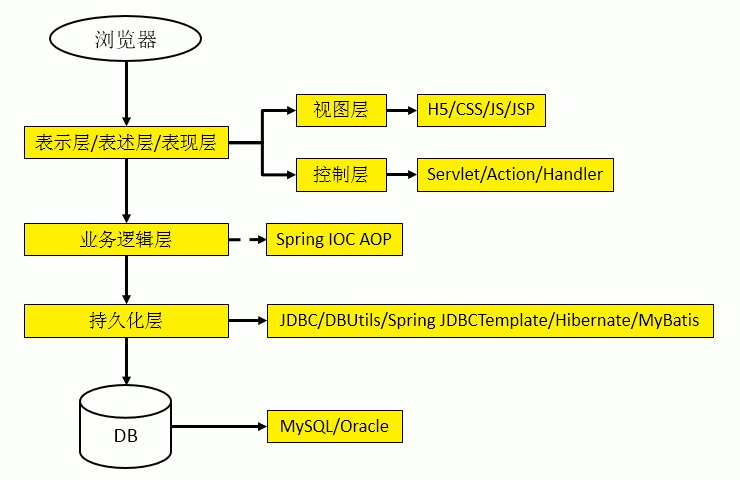
# Maven

1. 目前的前后端架构



1. Maven简介

用于构建Java工程，Maven将jar包引用到工程中，不需要将jar包复制到工程。规范下载jar包，用统一的规范存放在Maven中央仓库。Maven会自动导入jar包所依赖的包，不需要自己手动导入。

1. 构建工具：

Make->Ant->Maven->Gradle

1. 构建工程的步骤：

* 构建：

用java原文件、框架配置文件、JSP、HTML、图片等，去搭建、部署一个可以运行的工程；

* 清理：

将之前旧的class字节码文件删除；

* 编译：

源代码->可执行文件；

* 打包：

动态Web工程打成war包，Java工程打成jar包；

* 安装：

Maven的概念，将打包的得到的文件，复制到仓库中指定的位置；

* 部署：

将动态Web工程生成的War包，复制到Servlet容器中，使其可以运行。

1. 使用Maven需要部进行环境变量配置：

先下载Maven-bin文件，在配置环境变量（Maven是不需要安装的）

（网址：<https://maven.apache.org/download.cgi>）

1. **Maven核心概念：**

* **目录结构**



* **POM：**

Project Object Model 项目对象模型

在Maven中对应pom.xml，用于配置Maven工程。相当于动态Web工程的web.xml。

（DOM：Document Object Model 文档对象模型）

* **坐标：**groupid、artifactid、version

使用下面的三个向量定位唯一一个Maven工程

（1）**组织名：**<groupid>org.springframework</groupid>

（2）**模块名：**<artifactid>spring-core</artifactid>

（3）**版本：**<version>4.0.0.RELEASE</version>

对应的路径为：org/springframework/spring-core/4.0.0.RELEASE/spring-core-4.0.0.RELEASE.jar

* **依赖：**<dependencies>xxx</dependencies>

**作用：解决工程之间jar包冲突的问题（最近原则选取相同的jar包）**

（1）使用mvn install xxx（依赖包），解决找不到依赖的问题。

（2）依赖的范围：<scope>xxx</scope>

* **compile:**对主程序、测试程序有效，参与打包到jar包、部署；
* **test：**仅对测试程序有效，不参与打包、部署；

eg：junit

* **provided**：对主程序、测试程序有效，不参与打包、部署；

eg：servlet.jar

（3）依赖的传递性：

**仅作用于compile范围，test、provided范围仅影响自身。**

在pom.xml中添加依赖Maven，会自动在当前打开的工程中，添加对应的包，不需要在不同工程中重复执行添加操作。

* **仓库**

1. **分类**：

* **本地仓库：**当前电脑上的仓库；
* **远程仓库：**
* 私服：架设在局域网环境下的仓库，为局域网内的Maven工程服务；
* 中央仓库：架设在Internet上，为全世界服务；
* 中央镜像：分担中央仓库的流量；

1. **仓库中的内容：**Maven工程、Maven插件、第三方框架、jar包；

* **生命周期、插件、目标**

（1）生命周期：Clean lifeCycle、Default LifeCycle、

* **继承**

**eg：**junit，由于junit存在test范围，因此不能传递。这就会造成不同模块中的junit版本不一样。

**解决：**在Maven父工程中指定junit版本，打包方式为pom。在子工程中，声明对父工程的引用。**（注意：先安装父工程，否则子工程无法使用）**

**<!-- 子工程中：声明引用父工程 -->**  
<parent>  
 <groupId>com.atguigu.maven</groupId>  
 <artifactId>Parent</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
  **<!-- 以当前文件为基准的父工程pom.xml文件的相对路径 -->**  
 <relativePath>../Parent/pom.xml</relativePath>  
</parent>

**<!-- 父工程中：定义依赖 -->**  
**<!-- 配置依赖的管理 -->**  
<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.9</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</dependencyManagement>

* **聚合**

**作用：**一键安装各个模块工程。

**使用：**在父类工程（总工程）的pom.xml，配置如下命令即可。

**<!-- 配置聚合 -->**  
<modules>  
 **<!-- 指定各个子工程的相对路径 -->**  
 <module>../HelloFriend</module>  
 <module>../MakeFriends</module>  
 <module>../Hello</module>  
</modules>

1. **管理版本号：**

<properties>

**<!--自定义名字：atguigu.spring.version-->**  
 <**atguigu.spring.version**>4.1.1.RELEASE</atguigu.spring.version>   
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
</properties>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>

**<!--引用标签:${ },该方法可用于任何数据的引用-->**  
 <version>**${atguigu.spring.version}**</version>  
 <scope>compile</scope>  
</dependency>

# 常用命令

**注意：执行Maven命令时，需要在pom.xml根目录下执行。Maven命令执行时，如果找不到依赖，会自动联网下载对应的包。**

1. mvn clean：清理，清除生成的jar包和target文件
2. mvn complie：编译主程序
3. mvn test-complie：编译测试程序
4. mvn test：执行测试
5. mvn package：打包，生成jar包
6. mvn install：安装
7. mvn site：生成站点
8. mvn test：生成测试报告
9. mvn deploy：部署，将最终的包复制到远程仓库，共享或者部署到服务器上运行。