* 学习目标

- *能够理解Maven相关的概念
 - * 仓库管理:本地仓库,远程仓库(中心远程仓库,私服)
 - * 生命周期: clean,default(compile,test,install),site
 - * Maven插件
- *能够掌握Maven和idea结合开发方式
 - * java项目, javaweb项目
- *能够掌握Maven多模块的开发
 - * 按开发方式: MVC和Web三层架构:shopping--domain,dao,service,web,common,....
 - * 按功能模块: smartcar: common,customer,employee,pay,...
- *能够掌握Maven私服的部署和使用
 - * nexus

- *回顾
- * 文件上传: FileUpload, Servlet3.0, ...
- * Maven
- * Apache 项目
- * 项目管理工具:项目构建,依赖的管理,项目信息的管理
- * 项目构建:clean--compile--test-package--install--deploy
- * 依赖管理:denpency-->modelVersion---groupid--artifactid--version--package--scope(compile,test,runtime,provided)
 - * Maven约定目录:hellomaven--src------ main --- java

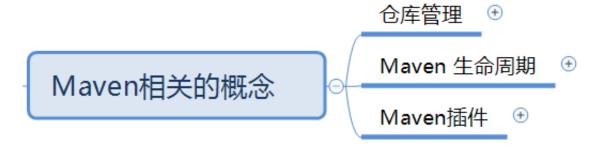
resources

target test --pom.xml

* 如何更换本地仓库, 远程中心的仓库,

- *自动构建java项目, javaweb项目, idea构建java项目,
- * 学习目标
 - *能够理解Maven相关的概念
 - *能够掌握Maven和idea结合开发方式
 - *能够掌握Maven多模块的开发
 - *能够掌握Maven私服的部署和使用

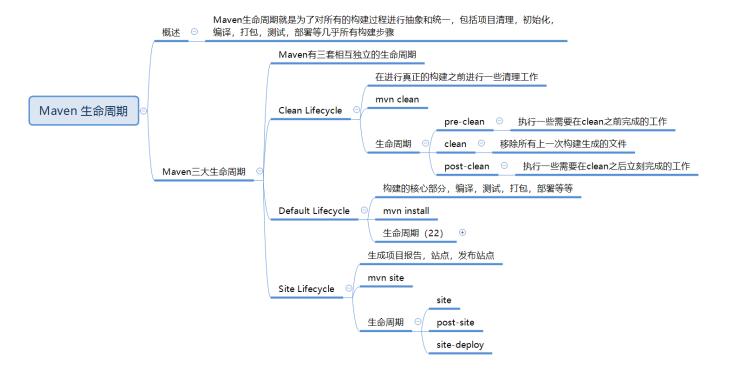
*能够理解Maven相关的概念



* Maven的仓库



* Maven的生命周期



* Default Lifecycle

validate generate-sources process-sources generate-resources process-resources compile process-classes generate-test-sources process-test-sources generate-test-resources process-test-resources test-compile process-test-classes test prepare-package package pre-integration-test integration-test post-integration-test verify install deploy

生命周期 (22)

^{*} Maven 插件



Maven插件

每个插件都能实现多个功能,每个功能就是一个插件目标

Maven的生命周期与插件目标相互绑定,以完成某个具体的构建任务,例如compile就是插件maven-compiler-plugin的一个插件目标

*能够掌握Maven和idea结合开发方式

概述

- *参考: 01-Maven和idea结合开发方式
- *能够掌握Maven多模块的开发



* 聚合

聚合

如果我们想一次构建多个项目模块,那我们就需要对多个项目模块进行聚合

聚合配置代码

<modules>

<module>模块A</module>

<module>模块B</module>

<module>模块C</module>

</modules>

其中module的路径为相对路径

* 继承

继承为了消除重复,我们把很多相同的配置提取出来,例如: grouptId, version等

继承

继承配置代码

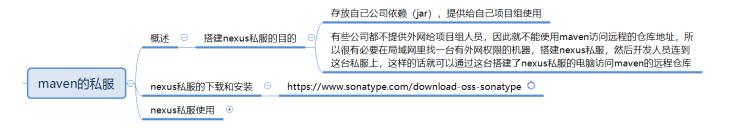
```
1 1 继承配置代码
 2 <parent>
 3
           <groupId>com.lg.maven
 4
           <artifactId>ParentProject</artifactId>
 5
           <version>0.0.1-SNAPSHOT</version>
           <relativePath>../ParentProject/pom.xml</relativePath>
 6
 7 </parent>
8 2 继承代码中定义属性
  properties>
      project.build.sourceEncoding>UTF-8/project.build.sourceEncoding>
10
      <junit.version>4.9</junit.version>
11
      <maven.version>0.0.1-SNAPSHOT</maven.version>
12
13 </properties>
14 访问属性的方式为${junit.version}
15 <dependency>
      <groupId>junit
16
      <artifactId>junit</artifactId>
17
      <version>${junit.version}</version>
18
      <scope>test</scope>
19
20 </dependency>
21 3 父模块用dependencyManagement进行管理
  <dependencyManagement>
22
      <dependencies>
23
24
      <dependency>
          <groupId>junit
25
          <artifactId>junit</artifactId>
26
          <version>${junit.version}</version>
27
28
          <scope>test</scope>
      </dependency>
29
      <dependency>
30
              <groupId>com.lg.maven
31
32
              <artifactId>helloMaven</artifactId>
              <version>${maven.version}</version>
33
34
              <type>jar</type>
              <scope>compile</scope>
35
36
         </dependency>
       </dependencies>
37
38 </dependencyManagement>
   这样的好处是子模块可以有选择的继承, 而不需要全部继承
39
```

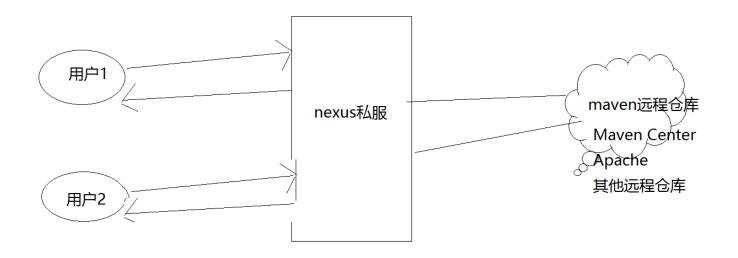
* 实战

聚合主要为了快速构建项目,继承主要为了消除重复



- * 按MVC模式和三层架构分Module
 - *参考: 01-Maven结合MVC模式和三层架构分Module
- * 按功能模块来分Module
 - *参考: 01-Maven结合功能模块来分Module
- *能够掌握Maven私服的部署和使用
 - * Maven私服的概述





* Maven私服具体操作

* 参考: <u>01-Maven私服具体操作</u>