

# Maven开发实战



张龙

2013-1-9

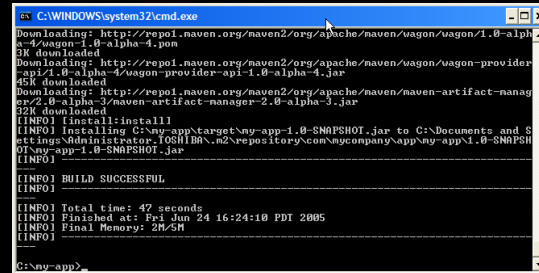
# Maven简介

- **Maven ['meivin]**
  - 内行 专家
- Maven是Apache软件基金会顶级项目
- <http://maven.apache.org>

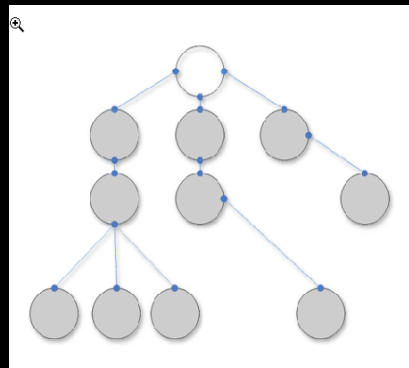


# 何为Maven

- Maven是基于项目对象模型(Project Object Model), 通过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是
  - 构建工具
  - 依赖管理工具
  - 文档工具



```
C:\WINDOWS\system32\cmd.exe
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon-1.0-alpha-4-pon
3K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon-provider-
api/1.0-alpha-4/wagon-provider-api-1.0-alpha-4.jar
45K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-artifact-manag
er/2.0-alpha-3/maven-artifact-manager-2.0-alpha-3.jar
12K downloaded
[INFO] Installing: install
[INFO] Installing C:\my-app\target\my-app-1.0-SNAPSHOT.jar to C:\Documents and S
ettings\Administrator\TOSHIBA\m2\repository\com\mycompany\app\my-app\1.0-SNAPSH
OT\my-app-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESSFUL
[INFO]
[INFO] Total time: 42 seconds
[INFO] Finished at: Fri Jun 24 16:24:10 PDT 2005
[INFO] Final Memory: 2M/5M
[INFO]
```



# Maven的作用

- 构建项目
- 构建项目文档
- 构建项目报告
- 项目依赖关系管理
- 项目配置管理
- 项目发布

# 使用Maven的好处

- 将一系列动作过程自动化
  - 编译
  - 单元测试
  - 打包
  - 部署
  - ...
- 项目依赖关系的处理
  - 防止jar包冲突
  - 自动获取jar包
- 约定优于配置

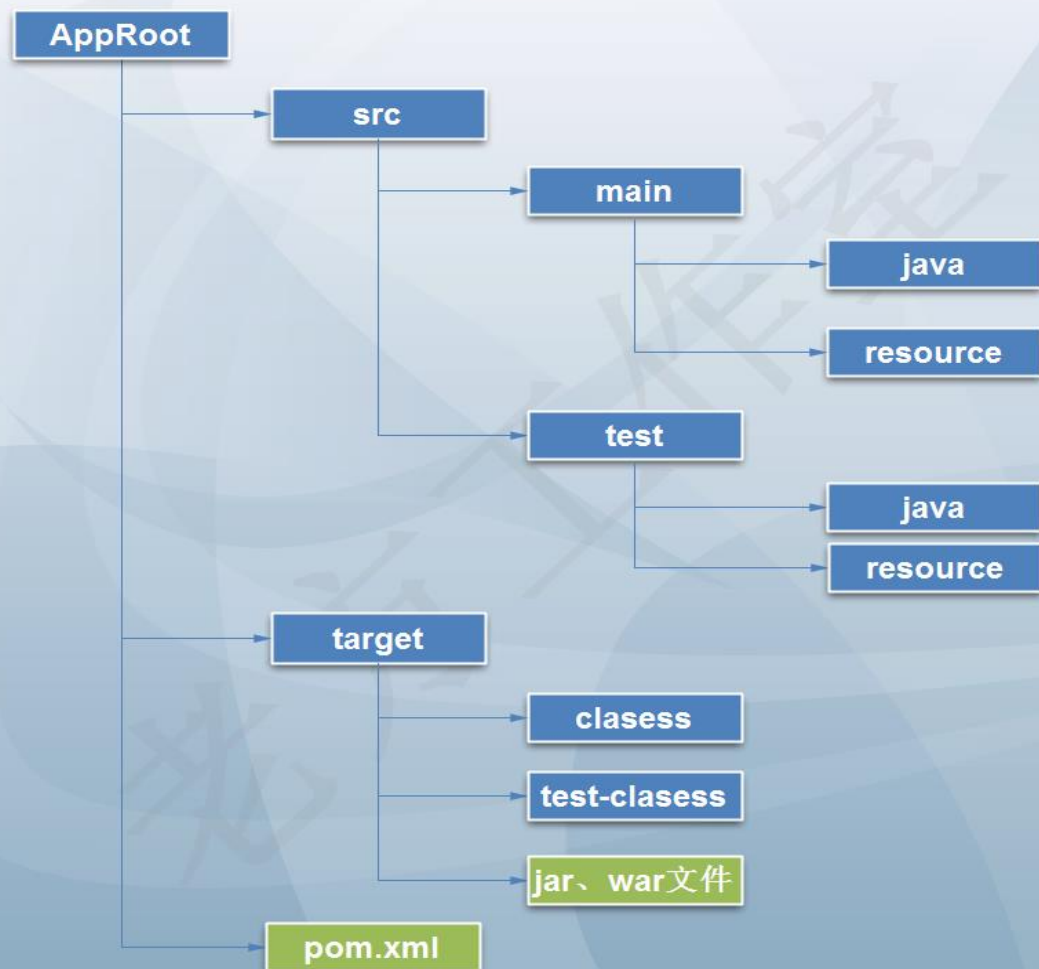
# 使用Maven的好处

- 目前很多项目都是使用Maven管理
  - 获取项目代码
  - 获取项目jar包
- 版本控制系统
  - 不必将jar包导入到版本控制系统中
  - 通过pom.xml文件在本地管理

# Maven 目录结构

- bin
  - maven运行脚本
- boot
  - java类加载器
- conf
  - maven重要的配置文件settings.xml
- lib
  - maven运行时所需的jar包
  - 超级pom文件位于maven-model-builder-3.0.4.jar

# Maven项目结构





# pom.xml

- 包含有关项目相关信息的元数据

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.lenovo</groupId>
  <artifactId>roiapp</artifactId>
  <packaging>war</packaging>
  <version>1.0</version>
  <name>Lenovo Social Media ROI APP</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.5</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>3.1.3.RELEASE</version>
    </dependency>
  </dependencies>
</project>
```

# POM关键要素

- **modelVersion:** 指定了当前POM模型的版本，对于Maven2或Maven3来说，只能是4.0.0
- **groupId:** 定义了项目属于哪个组，一般与项目所在公司或组织有关，如：com.lenovo
- **artifactId:** 定义了当前Maven项目在组中唯一的ID，如：rms
- **name:** 项目名称，如GAD RMS APP
- **version:** 项目当前的版本。SNAPSHOT意为快照，说明项目还处于开发中，不是稳定版本
- **packaging:** Maven项目的打包方式，默认为jar；Web项目是war，聚合项目是pom

# 坐标

- groupId, artifactId与version一同构成了Maven坐标，通过坐标我们就可以定位唯一的依赖

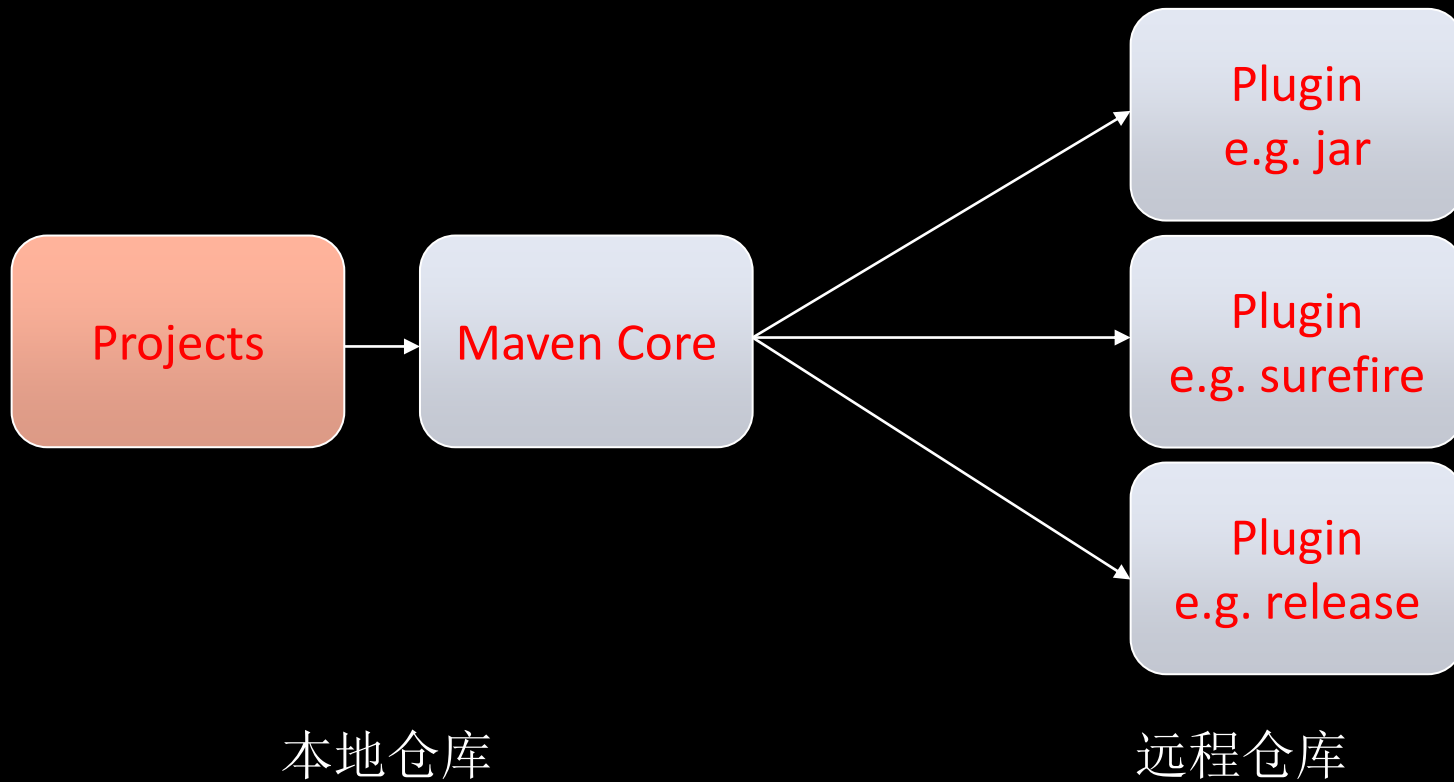
# Maven常用命令

- `mvn archetype:generate` : 创建 Maven 项目
- `mvn compile` : 编译源代码
- `mvn test-compile` : 编译测试代码
- `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` : 清除目标目录中的生成结果
- `mvn package` : 依据项目生成 打包文件
- `mvn install` : 在本地 Repository 中安装 打包结果
- `mvn deploy` : 将jar包发布到远程仓库
- `mvn eclipse:eclipse` : 生成 Eclipse 项目文件

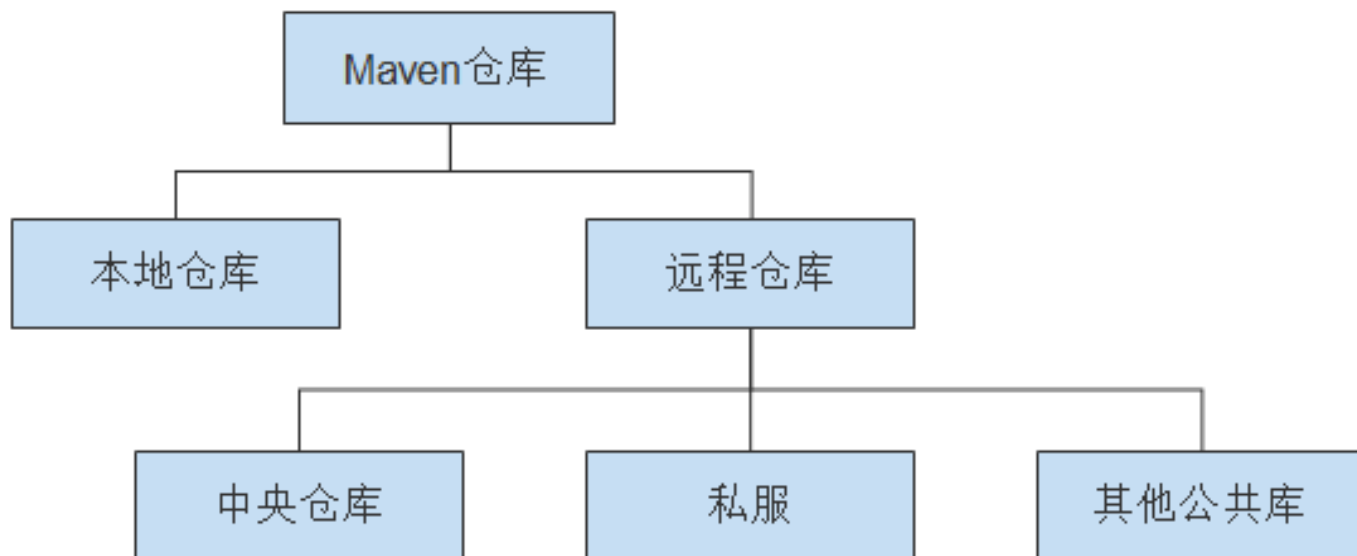
# Maven仓库

- Maven仓库就是放置所有JAR文件（WAR，ZIP，POM等等）的地方，所有Maven项目可以从同一个Maven仓库中获取自己所需要的依赖JAR

# Maven仓库

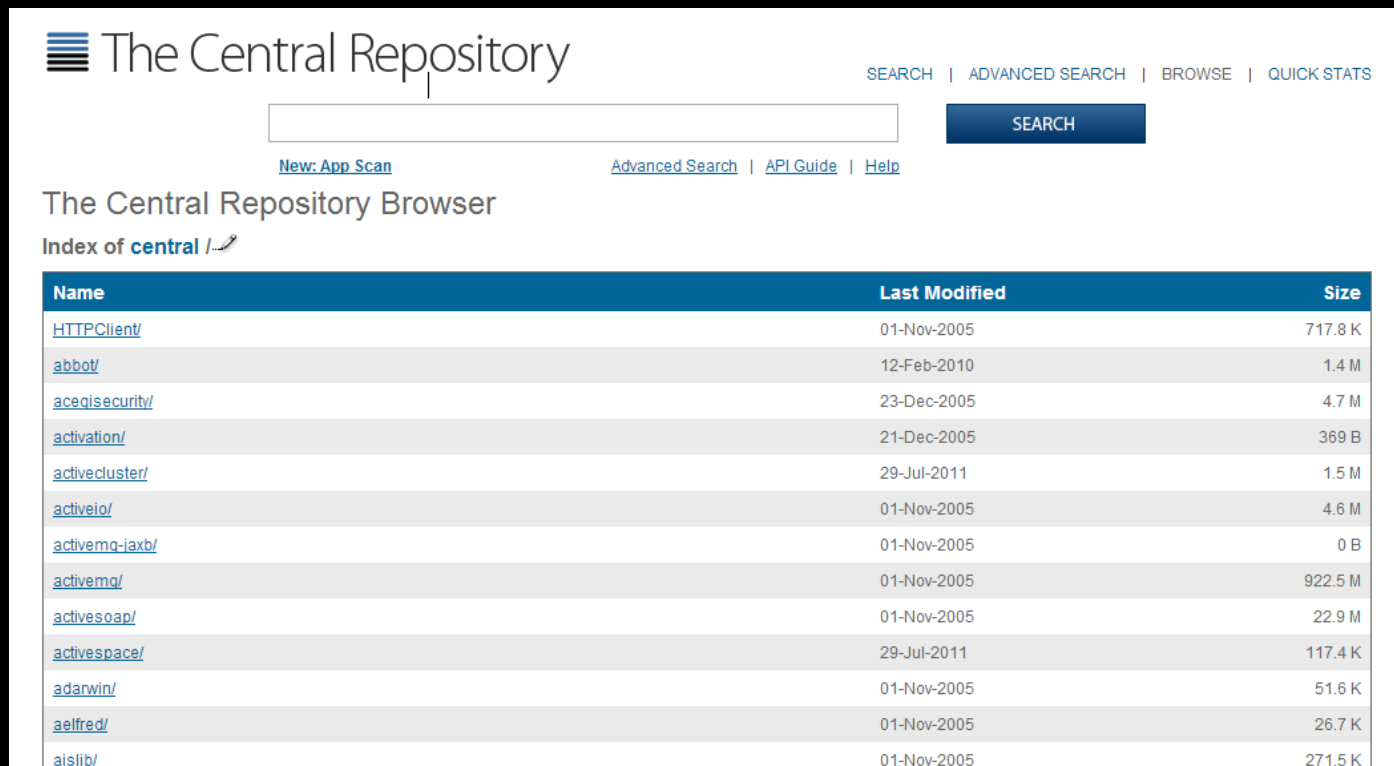


# Maven仓库分类



# 中央仓库

- 包含了世界上大多数流行的开源Java构件
- <http://search.maven.org/>



The screenshot displays the 'The Central Repository' website. At the top, there is a navigation bar with links for 'SEARCH', 'ADVANCED SEARCH', 'BROWSE', and 'QUICK STATS'. Below this is a search input field and a 'SEARCH' button. The main heading is 'The Central Repository Browser', followed by 'Index of central'. A table lists various artifacts with columns for 'Name', 'Last Modified', and 'Size'.

Name	Last Modified	Size
<a href="#">HTTPClient/</a>	01-Nov-2005	717.8 K
<a href="#">abbot/</a>	12-Feb-2010	1.4 M
<a href="#">acegisecurity/</a>	23-Dec-2005	4.7 M
<a href="#">activation/</a>	21-Dec-2005	369 B
<a href="#">activecluster/</a>	29-Jul-2011	1.5 M
<a href="#">activeio/</a>	01-Nov-2005	4.6 M
<a href="#">activemq-jaxb/</a>	01-Nov-2005	0 B
<a href="#">activemq/</a>	01-Nov-2005	922.5 M
<a href="#">activesoap/</a>	01-Nov-2005	22.9 M
<a href="#">activespace/</a>	29-Jul-2011	117.4 K
<a href="#">adarwin/</a>	01-Nov-2005	51.6 K
<a href="#">aelfred/</a>	01-Nov-2005	26.7 K
<a href="#">aislib/</a>	01-Nov-2005	271.5 K

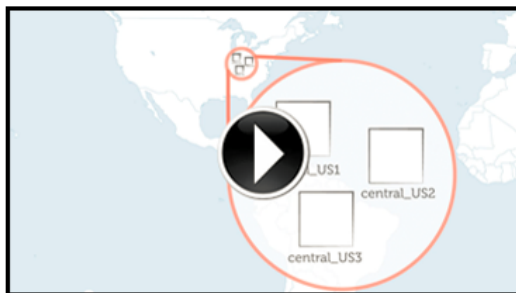


# 如何查找jar包

- 通过仓库搜索引擎
  - <http://www.jarvana.com/jarvana/>
  - <http://mvnrepository.com/>
  - <http://www.mvnbrowser.com/index.html>

# 私服

- 一种特殊的远程仓库，它是架设在局域网内的仓库服务，私服代理广域网上的远程仓库，供局域网内的Maven用户使用
- 最为流行的Maven私服是Nexus
  - <http://www.sonatype.org/nexus/>



Watch: [Proxy Repositories with Nexus](#)

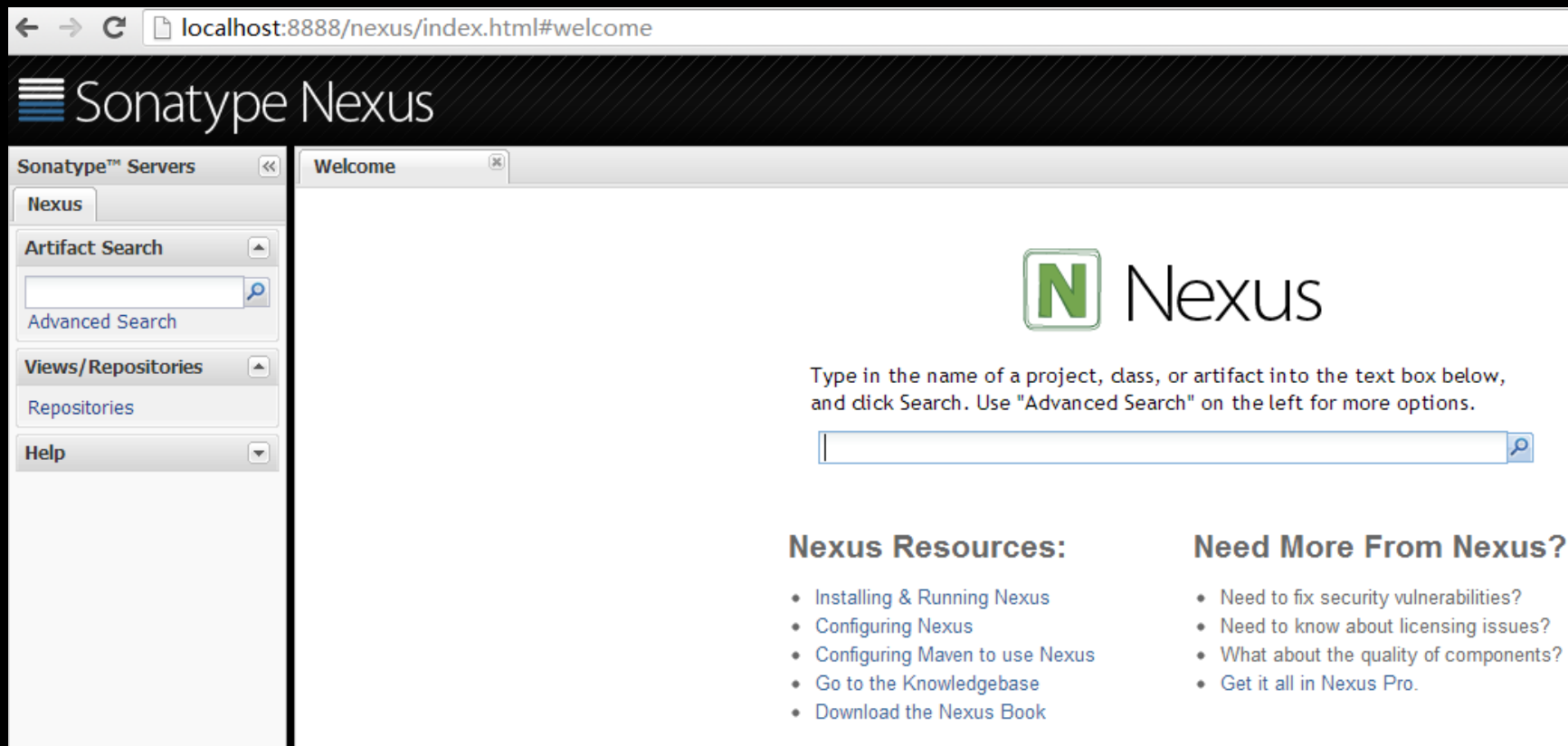
## Sonatype Nexus: Manage Artifacts

Sonatype Nexus sets the standard for repository management providing development teams with the ability to proxy remote repositories and share software artifacts. Download Nexus and gain control over open source consumption and internal collaboration.

DOWNLOAD NEXUS OSS

TRY NEXUS PRO

# Nexus运行截图



# 私服的优势

- 节省自己的外网带宽
- 加速Maven构建
- 部署第三方构件
- 提高稳定性，增强控制
- 降低中央仓库的负荷

# 依赖

- dependencies节点包含多个dependency
- 每个dependency又包含
  - groupId、artifactId、version;
  - type, 依赖的类型, 默认为jar;
  - scope, 依赖的范围;
  - optional, 是否可选;
  - exclusions, 排除传递依赖

# scope

- **compile**（默认范围） 编译依赖范围
  - 对编译、测试、运行三种classpath都有效
- **test** 测试依赖范围
  - 只对于测试classpath有效，如：junit;
- **provided** 已提供依赖范围
  - 编译和测试classpath有效，运行时无效，如：  
servlet-api
- **runtime** 运行时依赖范围
  - 测试、运行classpath有效，编译时无效，如JDBC驱动

# 传递性依赖

- 传递性依赖定义
  - A依赖于B，B依赖于C，A对于C是传递性依赖
- 非Maven项目
  - 下载核心包后，还需手动下载相关依赖，非常麻烦，还很容易出错
- Maven项目
  - maven可以通过项目中的pom.xml文件加载相关依赖；
  - 再通过依赖的项目中的pom文件继续加载相关依赖；
  - 直到所有依赖加载完成；

# 传递性依赖与依赖范围

	compile	test	provided	runtime
compile	compile	-	-	runtime
test	test	-	-	test
provided	provided	-	provided	provided
runtime	runtime	-	-	runtime



# 依赖调解 (Dependency Mediation)

- 第一原则：路径最近者优先
- 第二原则：第一声明者优先
- 第三原则：和版本无关

# 可选依赖<optional>

- 场景
  - 比如B是一个持久层的工具包，他支持多种数据库，包括mysql、mssql等，在构建这个工具包的时候，需要两种数据库的驱动程序，但在使用这个工具包的时候，只会依赖一种数据库；
- 可选依赖不会被传递，必须显示地声明
- 在绝大多数情况下，我们不应该使用可选依赖（根据单一职责原则）

# 排除依赖<exclusions>

- 排除依赖——阻断依赖关系
- <exclusions>节点可包含多个<exclusion>
- 每个exclusion只包含groupId和artifactId，因为同一个项目不可能出现groupId和artifactId相同但version不同的两个依赖；
- 主要用途是控制SNAPSHOT版本，一个依赖的不稳定可能会引起整个项目的不稳定

# Eclipse插件——m2eclipse

- 和nexus一样，是sonatype的开源工具；
- 基于Eclipse的插件；
- 官方站点地址<http://www.eclipse.org/m2e/>

# m2eclipse功能

- 创建导入maven项目
- 管理依赖
- 自动下载依赖
- 自动解析依赖的sources与javadoc
- 使用模板创建maven项目
- 浏览远程仓库
- 与svn集成
- ...