UNIPÊ
Pós-graduação

Especialização em Desenvolvimento Web

Gerência de Configuração de Software
Automação com Apache Maven

André Figueiredo

andre.unipe@gmail.com

Introdução

O que é o Maven?

- Ferramenta de automação de build
- Gerencia o processo de desenvolvimento de produtos (artefatos), em especial, Java
- Abordagem declarativa (diferentemente do Ant)
- Convenção sobre Configuração
- Desenvolvido pelo grupo Apache

maven

Introdução

Por que usar o Maven?

- Padronização do processo de desenvolvimento
- Gerenciamento de dependências (bibliotecas)
- Compartilhamento de componentes
  - Diferentemente de compartilhamento de código/build
- Extensível através de plugins
- Testes facilitados
- Documentação facilitada
- Fácil integração com ferramentas de integração contínua e monitoramento da qualidade do código
- Integração com IDE

Instalação

Baixar distribuição

http://maven.apache.org/download.html

Descompactar

Configurar variáveis de ambiente

Windows

M2_HOME=C:\apache-maven-3.0.4 (opcional)

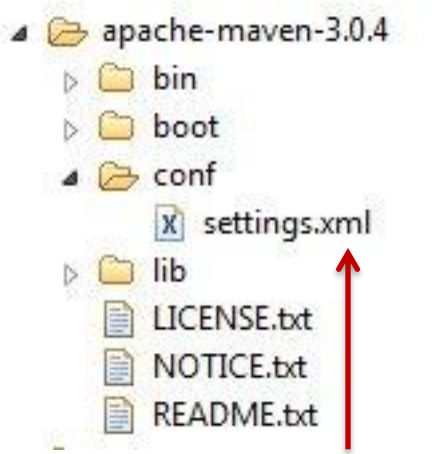
JAVA_HOME=<pasta de instalação do JDK>

PATH=**%M2_HOME%\bin**;%JAVA_HOME%\bin;%PATH%

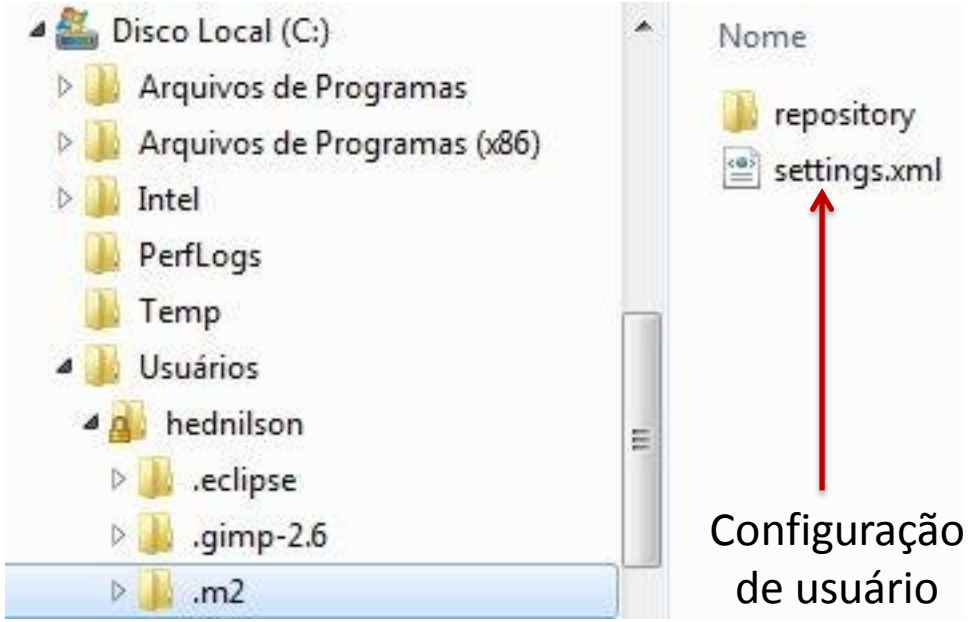
Testar instalação

PROMPT> mvn -version

Configuração



Configuração global



Configuração de usuário

Configuração

settings.xml

Proxies, Autenticação, Repositórios, Mirrors, Profiles, ...

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
    <proxies>
        <proxy>
            <id>internet-proxy</id>
            <active>true</active>
            <protocol>http</protocol>
            <username>proxyuser</username>
            <password>proxypass</password>
            <host>proxy.host.net</host>
            <port>80</port>
            <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
        </proxy>
    </proxies>
</settings>
```

Hands-on!

Baixe o projeto inicial e vamos “mavenizá-lo”!

Arquitetura

- Núcleo
  - Leitura de XML
  - Gerenciador de repositório de artefatos
  - Gerenciador de ciclo de vida e plugins

Plugins

As principais funcionalidades do Maven são providas através de plugins

Um plugin pode executar mais de uma atividade

goal implementado por um **Mojo**

Execução de **goals**

mvn plugin:goal

mvn plugin:goal –Dparametro=xyz

Ex:

mvn help:effective-settings

mvn help:describe –Dplugin=compiler

http://maven.apache.org/plugins

Ciclos de vida e fases

Clean

- pre-clean
- clean
- post-clean

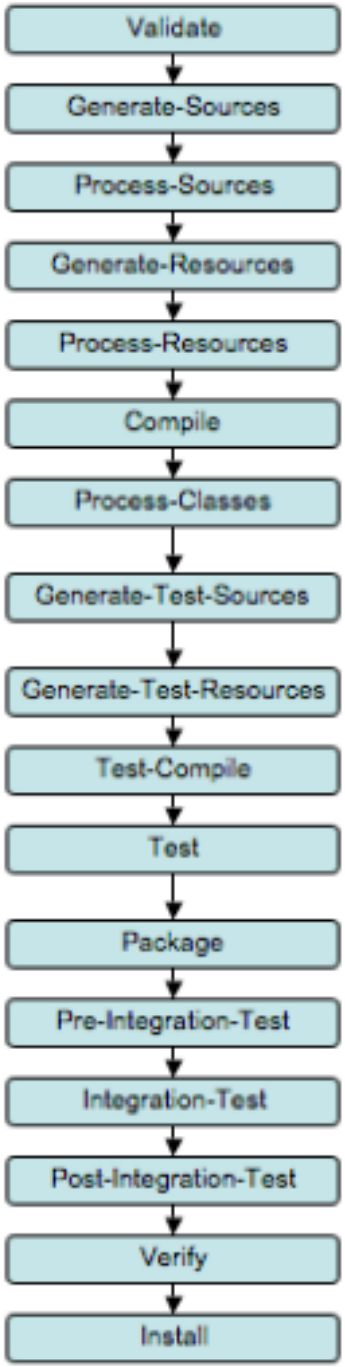
Build (default)

Site

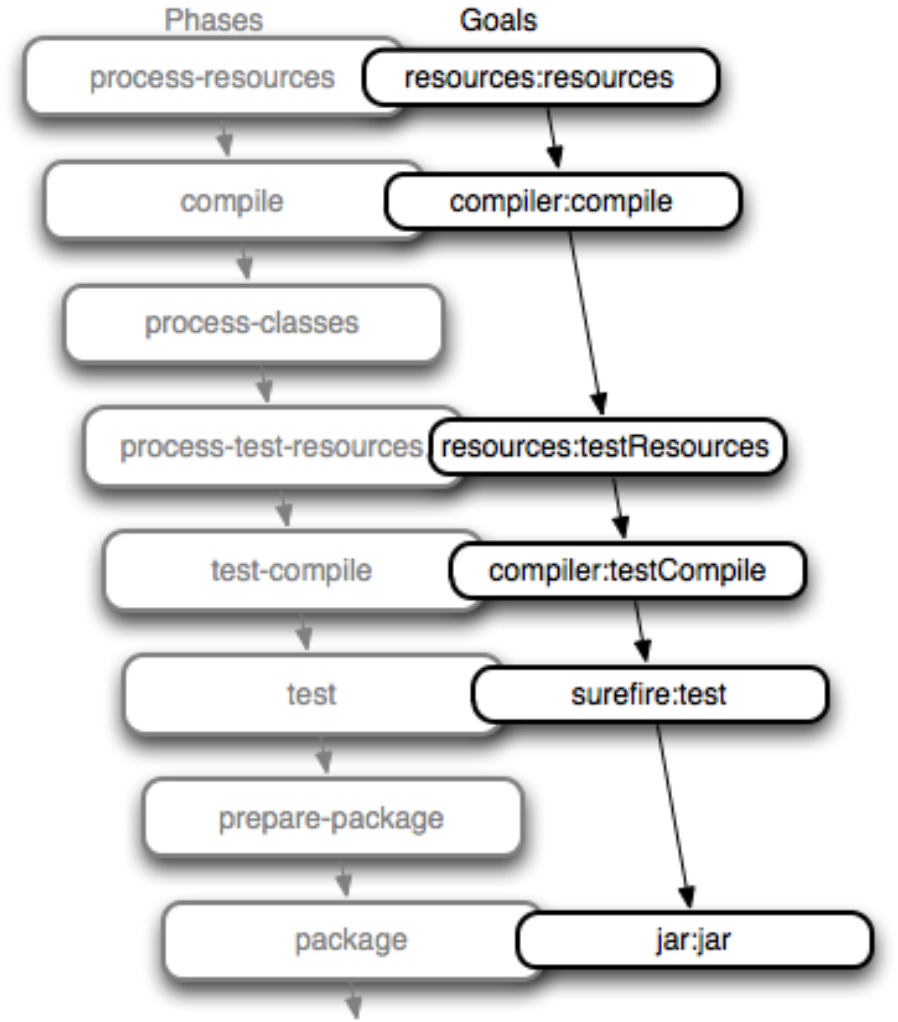
- pre-site
- site
- post-site
- deploy-site

UNIPÊ
Pós-graduação

Ciclo de vida default (build)e suas fases



Ciclo de vida default (build-jar)

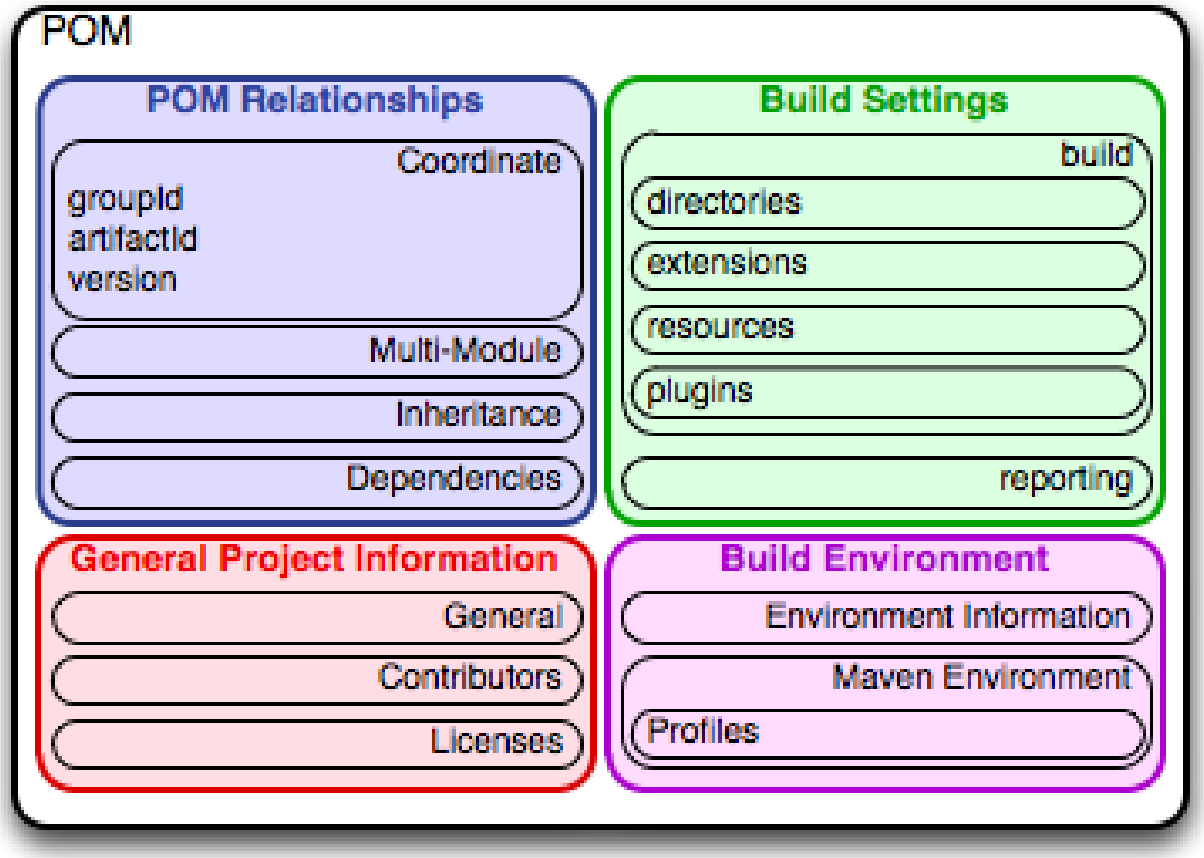


Note: There are more phases than shown above, this is a partial list

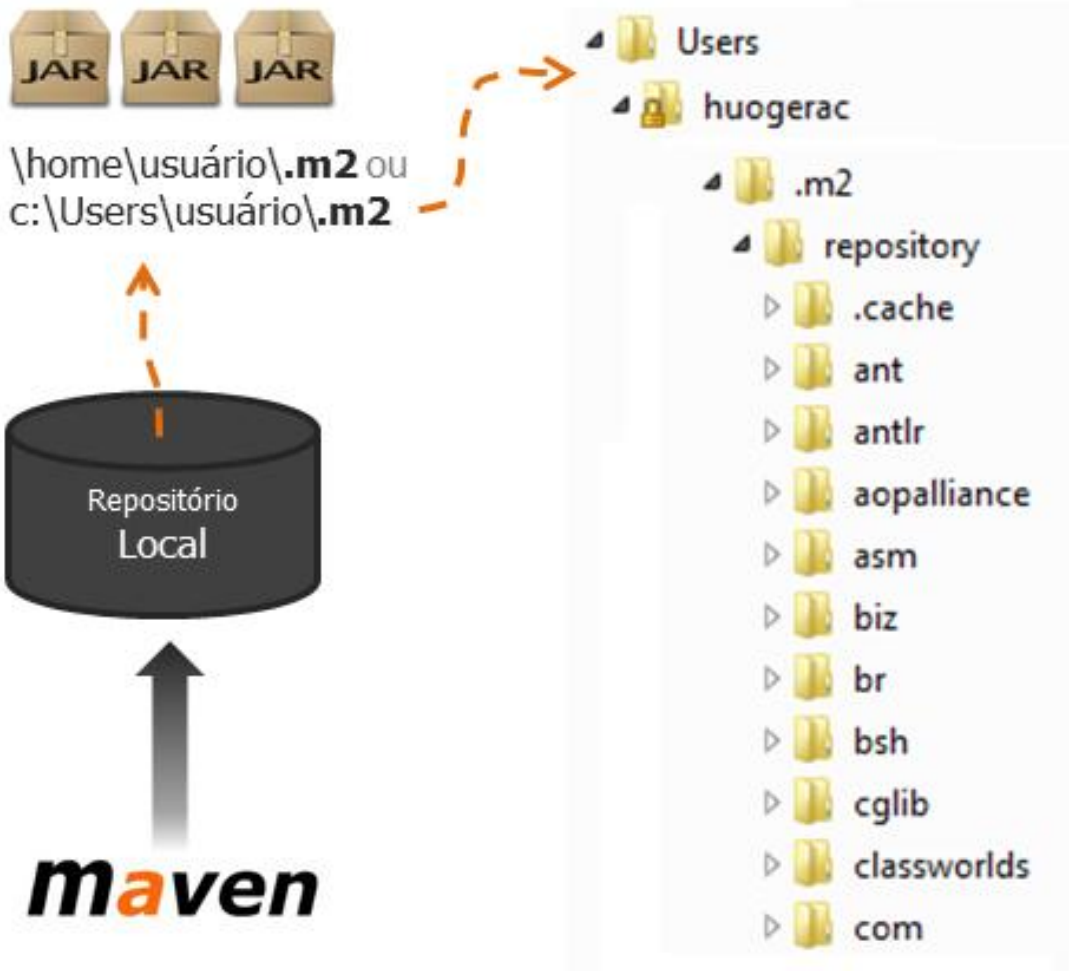
Hands-on!

Voltando!

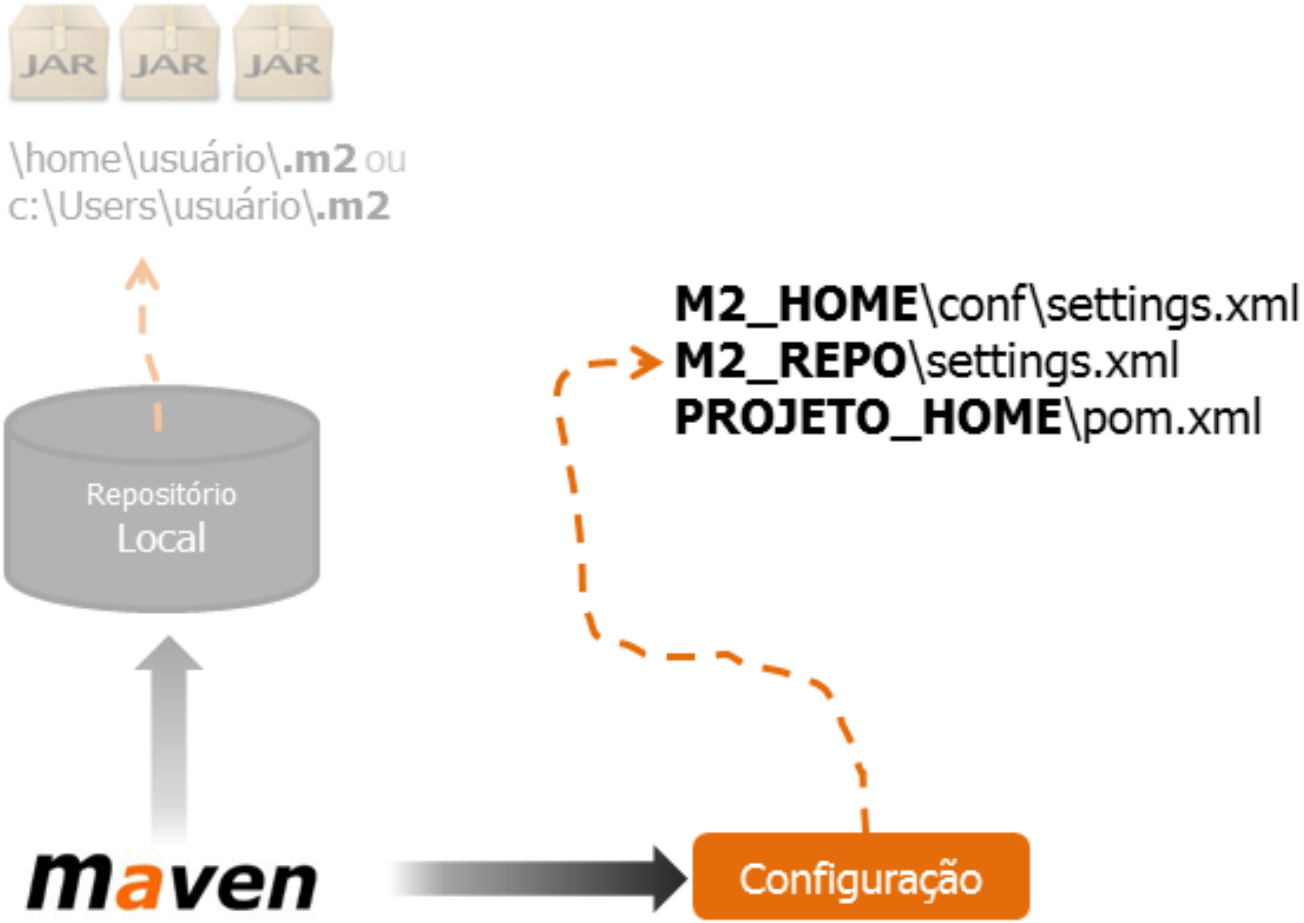
POM – Project Object Model



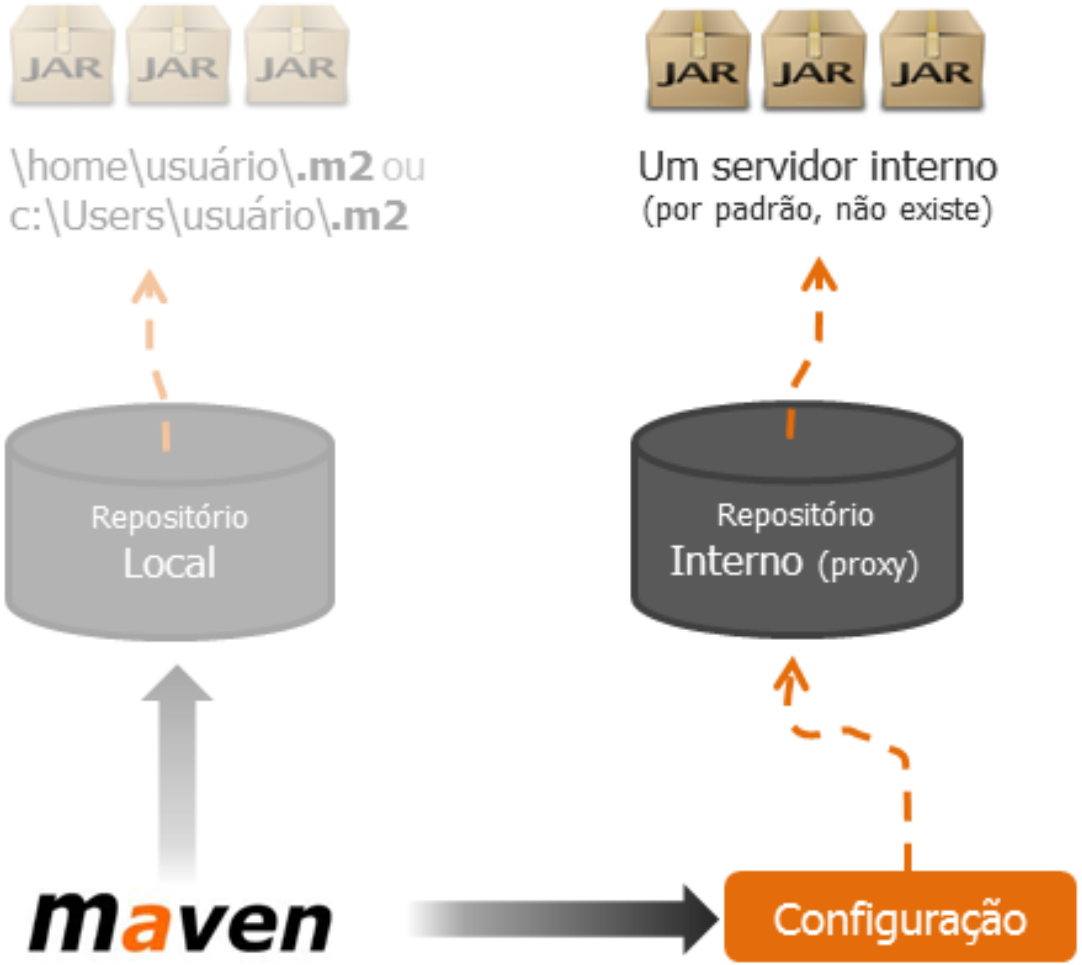
Repositório local



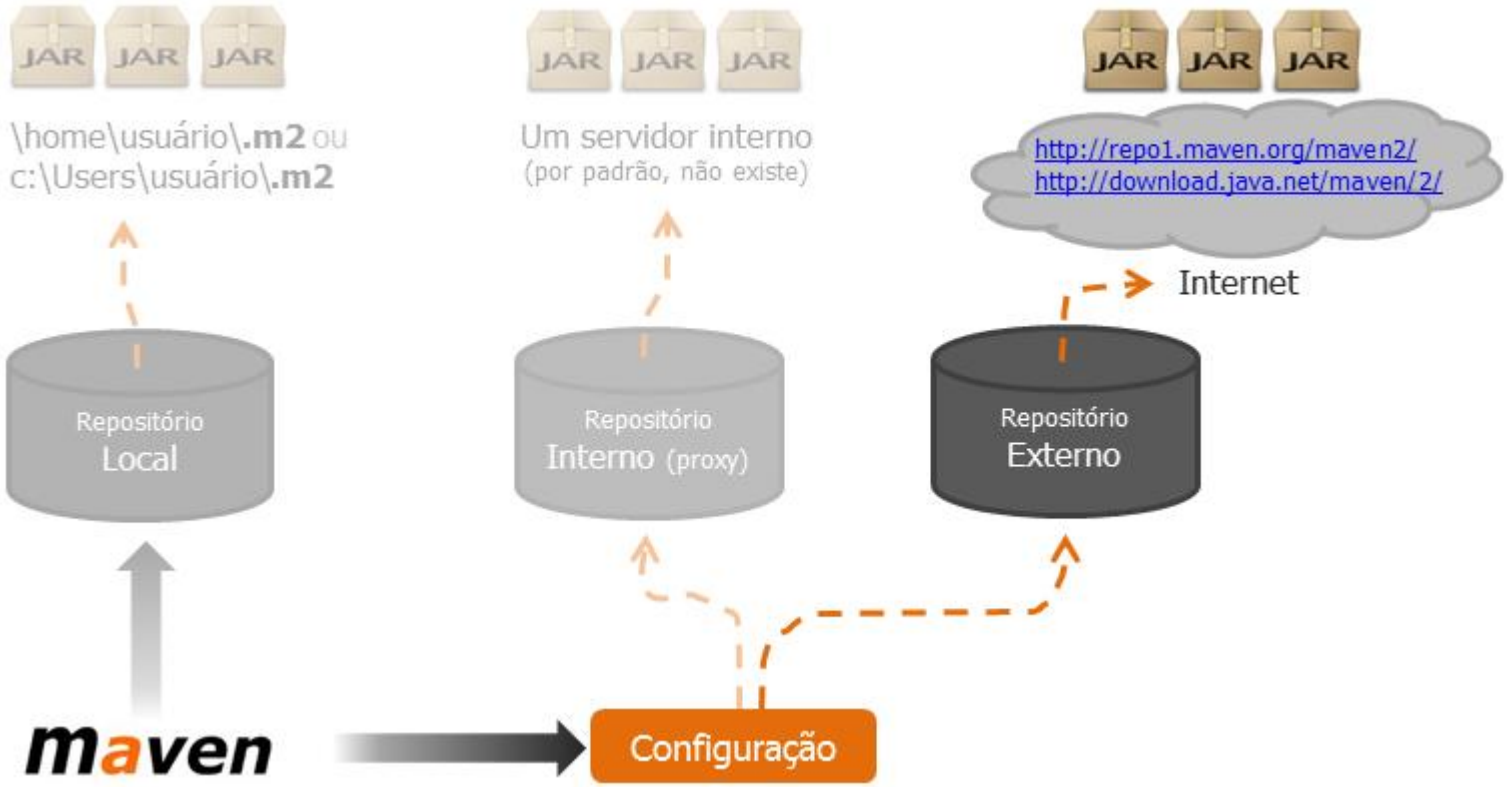
Outros repositórios



Repositórios internos



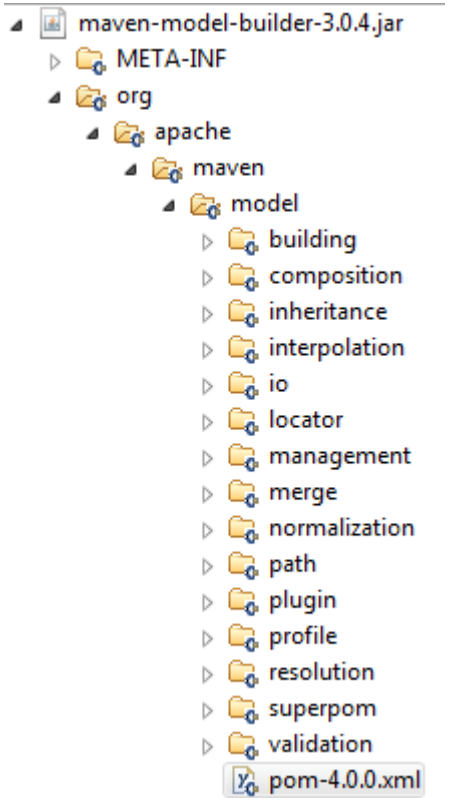
Repositórios externos



Hands-on!

Voltando!

Super POM



pom-4.0.0.xml

```
<project>
  <modelVersion>4.0.0</modelVersion>

  <repositories>
    <repository>
      <id>central</id>
      <name>Central Repository</name>
      <url>http://repo.maven.apache.org/maven2</url>
      <layout>default</layout>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>

  <pluginRepositories>
    <pluginRepository>
      <id>central</id>
      <name>Central Repository</name>
      <url>http://repo.maven.apache.org/maven2</url>
      <layout>default</layout>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <releases>
        <updatePolicy>never</updatePolicy>
      </releases>
    </pluginRepository>
  </pluginRepositories>
```

Super POM

pom-4.0.0.xml

```
<build>
  <directory>${project.basedir}/target</directory>
  <outputDirectory>${project.build.directory}/classes</outputDirectory>
  <finalName>${project.artifactId}-${project.version}</finalName>
  <testOutputDirectory>${project.build.directory}/test-classes</testOutputDirectory>
  <sourceDirectory>${project.basedir}/src/main/java</sourceDirectory>
  <scriptSourceDirectory>src/main/scripts</scriptSourceDirectory>
  <testSourceDirectory>${project.basedir}/src/test/java</testSourceDirectory>
  <resources>
    <resource>
      <directory>${project.basedir}/src/main/resources</directory>
    </resource>
  </resources>
  <testResources>
    <testResource>
      <directory>${project.basedir}/src/test/resources</directory>
    </testResource>
  </testResources>
  <pluginManagement>
    <!-- NOTE: These plugins will be removed from future versions of the super POM -->
    <!-- They are kept for the moment as they are very unlikely to conflict with lifecycle
    <plugins>
      <plugin>
        <artifactId>maven-antrun-plugin</artifactId>
        <version>1.3</version>
      </plugin>
      <plugin>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>2.2-beta-5</version>
      </plugin>
      <plugin>
        <artifactId>maven-dependency-plugin</artifactId>
        <version>2.1</version>
      </plugin>
      <plugin>
        <artifactId>maven-release-plugin</artifactId>
        <version>2.0</version>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

Hands-on!

Voltando!

Integração com Eclipse

- Plugin Maven:
  - maven-eclipse-plugin
    - Gera arquivos do eclipse a partir do pom.xml
      - .project, .classpath, .settings, ...
    - Comando de linha
      - mvn eclipse:eclipse
      - Podem ser criados launchers
    - http://maven.apache.org/plugins/maven-eclipse-plugin/
- Plugin Eclipse:
  - m2eclipse
    - Configura projeto no eclipse a partir do pom.xml
    - http://www.sonatype.org/m2eclipse

Instalação do m2eclipse