Maven

Prof. Roberto Rocha

Introdução

- História: g++ (manualmente), Make, Ant, Inv,
- Maven: Centralizar informações de Build.
- Ferramenta livre para gestão, construção e compreensão de projetos Java
- Objetivo: Criar projetos com estruturas padronizadas
  - mvn archetype:generate
- Atividades: builds, dependências, releases, distribuição, documentação e outros.
- Site: maven.apache.org

Introdução

- Problemas:
  - Tarefas manuais repetitivas
  - Processos de construção inconsistentes
  - Diferença de ambientes de desenvolvimento (IDE e Sistema Op.)
- Vantagens
  - Maior foco no desenvolvimento
  - Integração com outras ferramentas
  - Automatização de processos
  - Ir para casa mais cedo :)

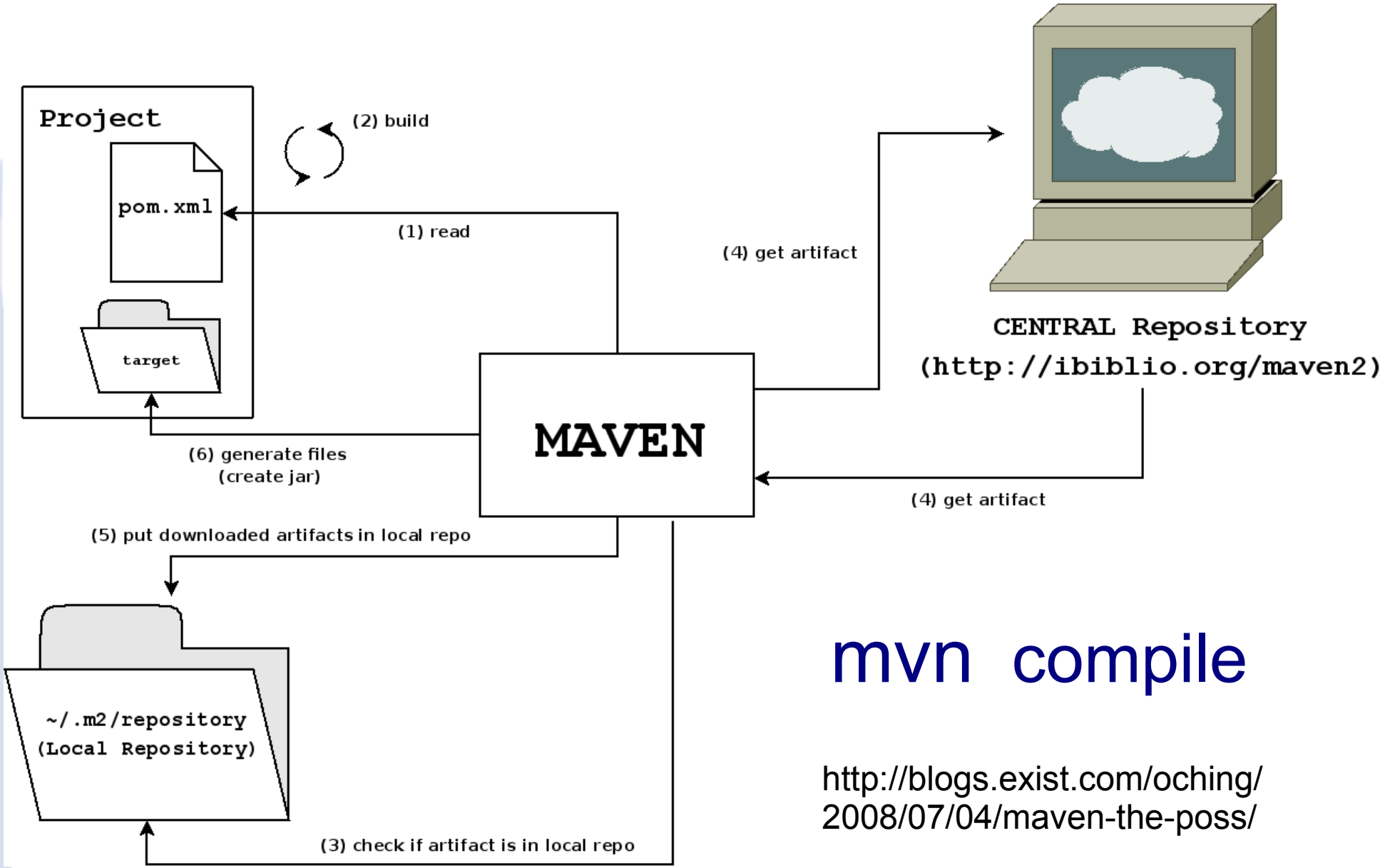
Introdução

- Construção de software:
  - Pré-processamento
  - Compilação e Testes
  - Empacotamento
  - Distribuição
- Administração
  - Geração de relatórios e site web
  - Geração de documentos
  - Fácil comunicação do time

Entendimento do projeto

- Questões relativas ao projeto:
  - Onde está o código fonte?
  - Quais dependências ele tem?
  - Existem testes? Cobrem todo o código?
  - Existe documentação?
  - O que produz cada subprojeto?
  - Geração de site com informação básica
  - Boas práticas de desenvolvimento como: testes, qualidade de código, etc.

Execução do Maven



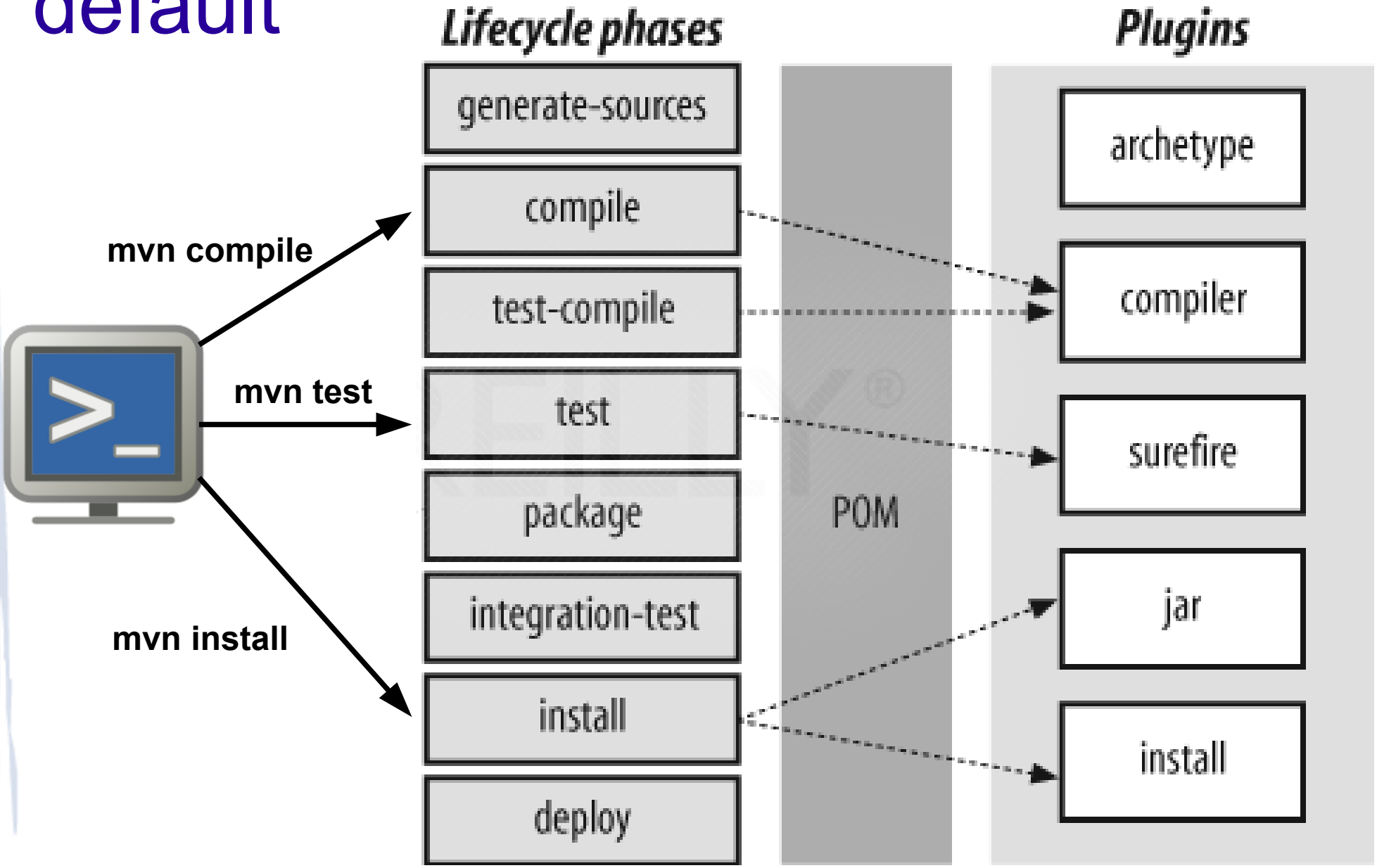
mvn compile

http://blogs.exist.com/oching/2008/07/04/maven-the-poss/

Ciclo de Vida

- Sequência ordenada de fases para a construção do software
- Possui goals (metas) que podem estar relacionadas com uma fase
- 3 ciclos de vida
  - Default
  - Site (mvn site)
  - Clean (mvn clean)

Ciclo default

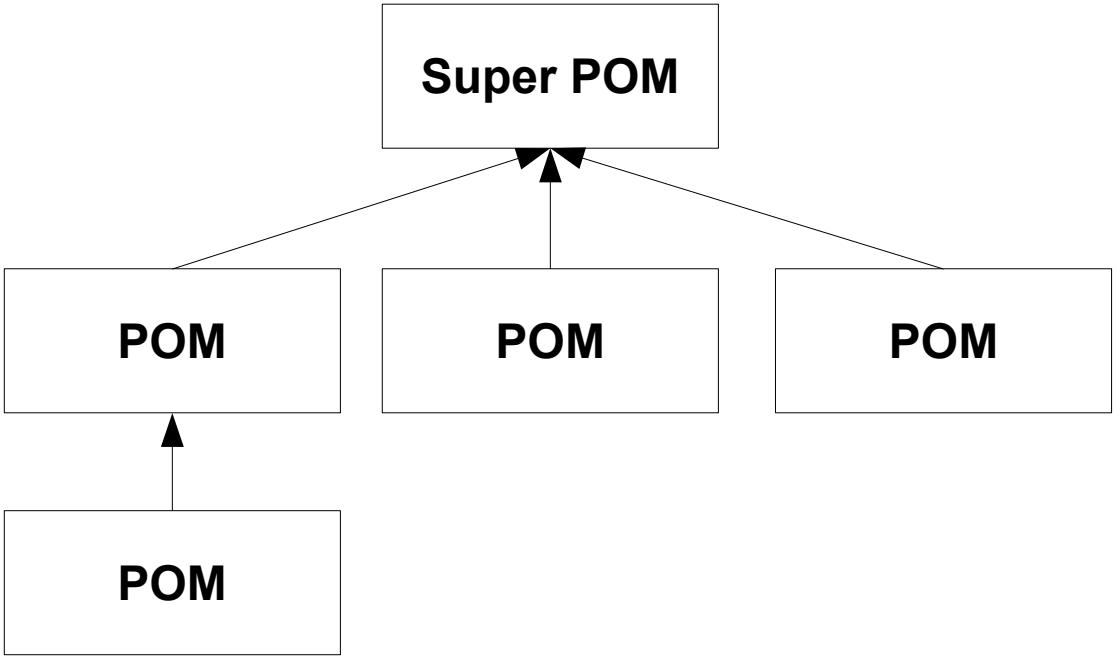


Adaptado de: http://www.oreilly.de/catalog/9780596527938/toc.html

POM.xml (Project Object Model)

- É o ponto central de informação do projeto
- É um arquivo xml onde são definidos:
  - Descrição e propriedades do projeto
  - Dependências de módulos externos
  - Plugins
  - Repositórios de código fonte
- Possui objetivos pré-definidos
- É a unidade básica de trabalho do Maven
- Pode herdar características de outro POM

Hierarquia do POM



- $ mvn help:effective-pom

Exemplo de POM.xml

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>br.edu.univas</groupId>
    <artifactId>Library</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>Library project of Univas</name>
    <plugins>
    </plugins>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
        </dependency>
    </dependencies>
</project>
```

Dependências

- São os artefatos (arquivos ou bibliotecas) que o Maven precisa para construir o projeto.
- A maioria das dependências são arquivos jars, wars, rars, zip, etc.
- Todas as dependências necessárias para o projeto devem ser definidas no POM.xml, para serem utilizadas pelo Maven.

Dependências

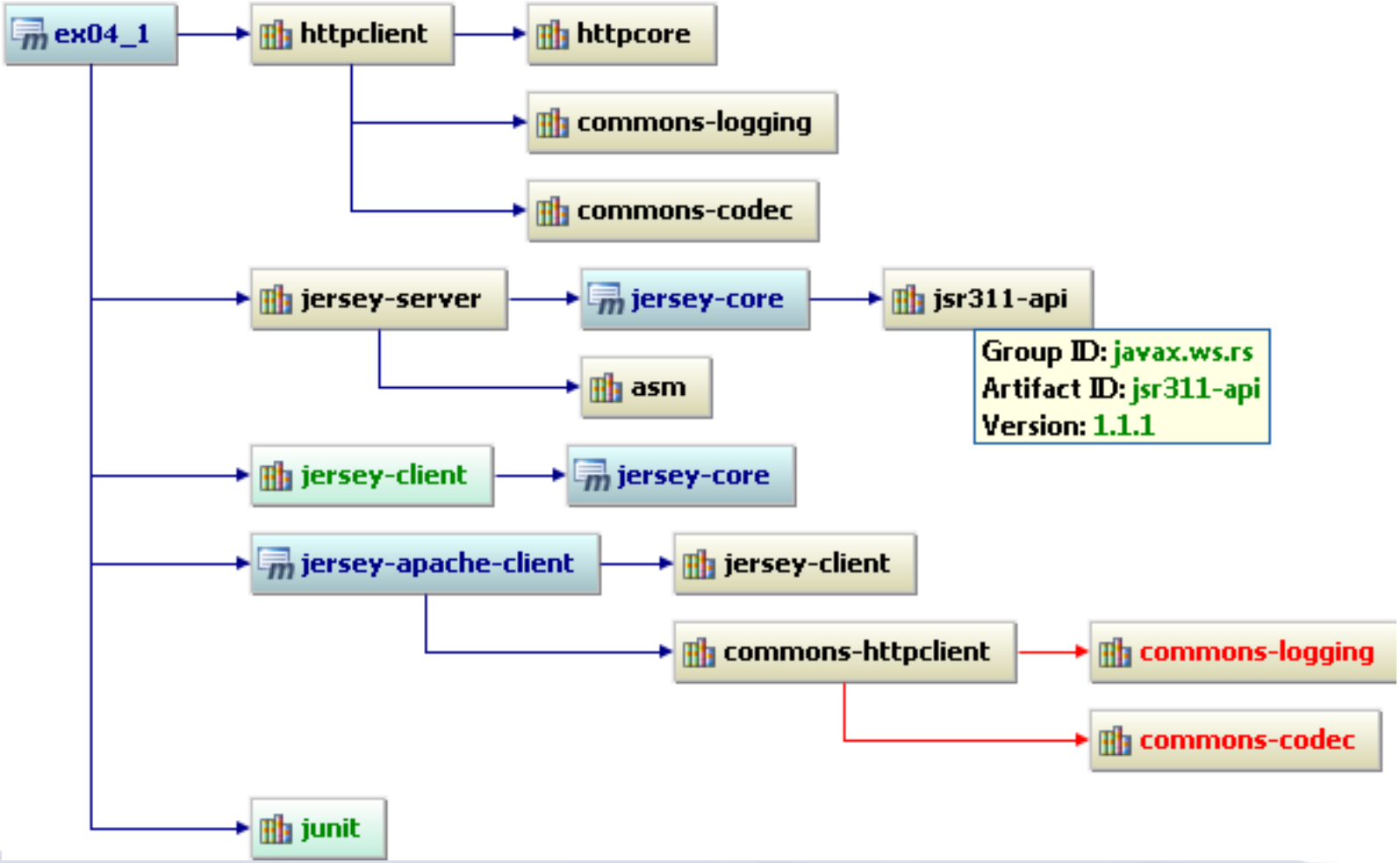
```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>br.edu.univas</groupId>
    <artifactId>Library</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>Library project of Univas</name>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <type>jar</type>
        </dependency>
    </dependencies>
</project>
```

Dependências

- Type: jar (default), war, ear, rar, etc.
- Escope
  - Compile: usada em todas as fases
  - Provided: usada apenas na fase de compilação
  - Runtime: é usada apenas em tempo de execução
  - Test: é usada na fase de testes
  - System: localização especifica do artefato (normalmente o artefato não está em nenhum repositório)

Dependências Transitivas

- São dependências de dependências



http://blogs.jetbrains.com/idea/2010/05/maven-dependencies-diagram/

Repositório

- É uma estrutura de diretórios e arquivos.
- O Maven utiliza para armazenar e buscar dependências do projeto.
- O repositório pode ser:
  - Local: $HOME/.m2
  - Proxy: servidor interno da empresa
  - Remoto: servidor HTTP com a mesma estrutura de 'diretórios' a partir de uma URL.
- Declarar uma dependência equivale a uma busca do artefato no repositório.

Repositório

- Exemplo de uma URL do repositório local:

<version>

C:/user/roberto/.m2/org/hibernate/hibernate/3.1.3/hibernate-3.1.3.jar

<URL repo> <groupId> <artifactId> <type>

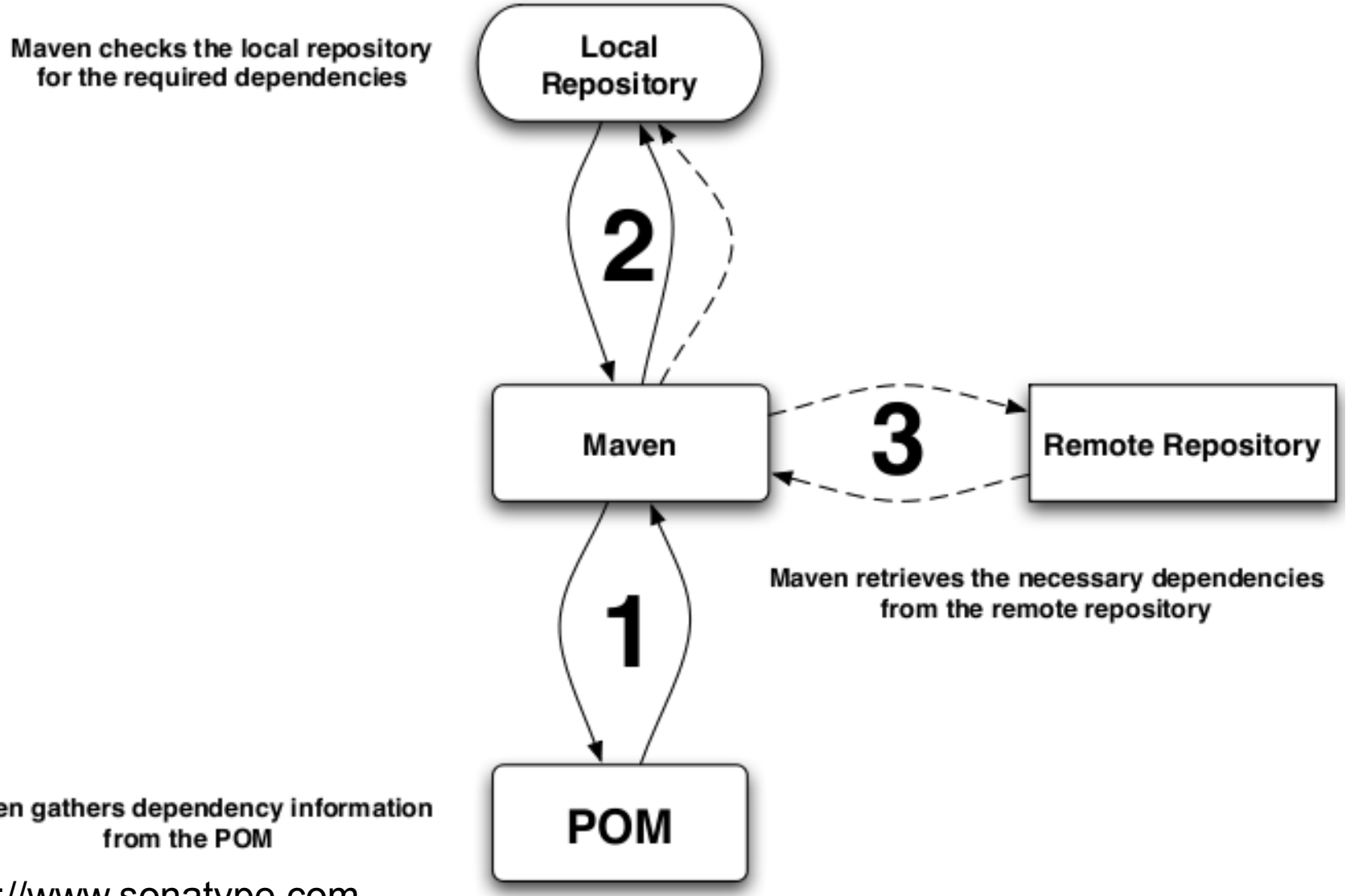
- Exemplo de uma URL do repositório remoto:

<version>

http://ibiblio.org/maven2/org/hibernate/hibernate/3.1.3/hibernate-3.1.3.jar

<URL repo> <groupId> <artifactId> <type>

Busca no Repositório



http://www.sonatype.com

Estrutura de diretórios do **Projeto**

- O Maven fornece uma estrutura padrão de diretórios de acordo com o tipo de projeto: Web, EJB, Web-services, EAR, etc.
- Seguindo a estrutura padrão, não é preciso indicar para o Maven onde fica seus arquivos.

Estrutura de diretórios do projeto

- Estrutura padrão sugerida (não obrigatória)

src	→ Diretório raiz.
src/main/java	→ Código fonte java.
src/main/resources	→ Arquivos de configuração do fonte.
src/main/config	→ Arquivos de configuração.
src/main/webapp	→ Arquivos de aplicação web.
src/test/java	→ Código-fonte de teste.
src/test/resources	→ Arquivos de configuração de teste.
src/site	→ Arquivos do Website.

- Diretório 'target': o Maven usa para armazenar os arquivos gerados no processo de build do projeto.

Multi-projetos

- Utilizado em projetos maiores (ex.: JEE)
- Facilita a centralização do processo de build

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.sample</groupId>
    <artifactId>sample-project</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>
    <modules>
        <module>sample-jar</module>
        <module>sample-ejb</module>
        <module>sample-war</module>
        <module>sample-ear</module>
    </modules>
</project>
```

Relatórios

- O Maven automatiza a geração de diversos relatórios
- Eles permitem:
  - Monitorar a saúde do código (qualidade)
  - Obter métricas
  - Conhecer Cobertura de código
  - Ver o código fonte como página web
  - Acompanhar as mudanças
  - Disponibilizar a documentação do projeto

Relatórios - Cobertura

- Incluir este fragmento no pom.xml

```
<reporting>
    <plugins>
      <plugin>
<!-- use mvn cobertura:cobertura to generate cobertura reports -->
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>cobertura-maven-plugin</artifactId>
          <version>2.4</version> <!-- update properly -->
      </plugin>
    </plugins>
</reporting>
```

  - mvn cobertura:cobertura

Relatórios – CheckStyle

- Incluir este fragmento no pom.xml, junto com o plugin de Cobertura

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-checkstyle-plugin</artifactId>
    <version>2.17</version> <!-- update properly -->
</plugin>
```

  - mvn site

Relatórios – CheckStyle

- Incluir este fragmento no pom.xml, junto com o plugin de Cobertura

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-checkstyle-plugin</artifactId>
   <version>2.17</version> <!-- update properly -->
</plugin>
```

  - mvn site

Archetypes

- São templates de projetos (modelos)
- Define o conteúdo do pom.xml e a estrutura dos diretórios para cada tipo de projeto
- Execução:
  - mvn archetype:generate
  - mvn archetype:generate \
    -DarchetypeGroupId=org.apache.maven.archetypes \
    -DarchetypeArtifactId=maven-archetype-**webapp** \
    -DgroupId=br.edu.univas \
    -DartifactId=bmi-webapp

Plugins Úteis

- mvn eclipse:eclipse
- mvn eclipse:configure-workspace

- mvn dependency:tree
- mvn dependency:resolve

- mvn archetype:create-from-project
- mvn archetype:generate

Conclusão

- Desenvolvedor mais focado em seu trabalho
- Evita erros simples cometidos através do excesso de repetições
- As ferramentas de *build* são base para outros sistemas, por exemplo, integração contínua
- O aprimoramento das ferramentas deixam o aprendizado muito mais simples

Exemplo