Centro Universitário de
Anápolis - UniEVANGÉLICA
Av. Universitária Km 3,5 Cx. Postal 122 e 901
CEP: 75 070-290 Anápolis/GO
Fones: (062) 3310 – 6658 e 3310 – 6690

Acadêmicos:
Jônatas Gabriel da Silva Santos;
José Francisco de Oliveira Júnior;
Melquisedeque Semais de Morais;
Newton Rodrigues da Silva Júnior.

Gerência de Configuração de Software

AULA 9

PORTIFÓLIO – MAVEN

O que é o Maven?

Apache Maven, ou Maven, é uma ferramenta de automação de compilação utilizada primariamente em projetos Java. Ela é similar à ferramenta Ant, mas é baseada em conceitos e trabalhos diferentes em um modo diferente. Também é utilizada para construir e gerenciar projetos escritos em C#, Ruby, Scala e outras linguagens. O projeto Maven é hospedado pela Apache Software Foundation, que fazia parte do antigo Projeto Jakarta.

O Maven utiliza um arquivo XML (POM) para descrever o projeto de software sendo construído, suas dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e plug-ins necessários. Ele vem com objetivos pré-definidos para realizar certas tarefas bem definidas como compilação de código e seu empacotamento.

Benefícios

O maven automatiza o sistema de build e gerencia as dependências do seu projeto.

Na questão das dependências, imagina que quando você faz isso manualmente, tem que baixar e depois adicionar as libs ao seu projeto. Seu projeto fica grande, quando você for mandar pra outro pessoal ou subir no github, teria que mandar as libs também, ou então mandar uma lista das libs que precisa.

Sem falar que as bibliotecas as vezes dependem de outras, e você tem que saber disso para que tudo no seu projeto funcione. No fim das contas tudo pode ficar uma bagunça se você gerenciar isso manualmente.

Com o Maven você declara tudo em um arquivo e ele se encarrega de adicionar as bibliotecas que o seu projeto depende. Ele se encarrega de baixar as dependências e dependências das dependências.

Da pra fazer umas coisas bem legais, do tipo dizer que uma biblioteca deve estar disponivel apenas quando rodar os testes.

Quando você mandar pra outra pessoa ou por no github, basta enviar esse arquivo onde as configurações estão declaradas e a outra pessoa quando executar o maven ele irá configurar tudo.

Enfim, algumas tarefas que poderiam ser um inferno caso você fosse gerenciar manualmente.

Na parte de build facilidade você gerar o seu .jar ou .war por exemplo. Tem outras coisas também.

Mais uma coisa: também vejo a questao da IDE como vantagem. Você bao precisa mais criar um projeto Eclipse ou Netbeans, você pode simplesmente criar um projeto maven e as ides hoje todas irão suportar sem qualquer problema. Isso facilita em uma equipe os devs trabalharem com IDEs diferentes, por exemplo.

Modelo de Objetos de Projetos (POM)

Project Object Model (literalmente "projeto modelo de objeto"), ou POM, é a peça fundamental de um projeto do Apache Maven.

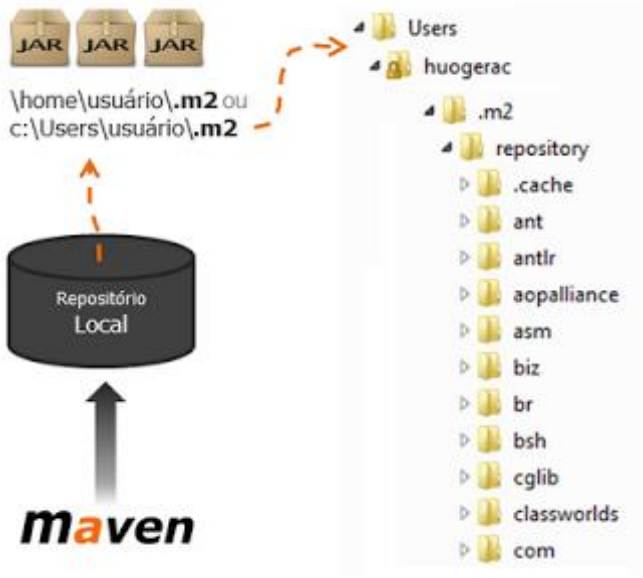
Um POM possui as infomações básicas de um projeto, bem como as diretivas de como o artefato final deste projeto deve ser construido.

A versão 1.0 do Maven utiliza o arquivo project.xml para definição do POM. Na versão 2.0, este arquivo passa a se chamar pom.xml .

Repositório

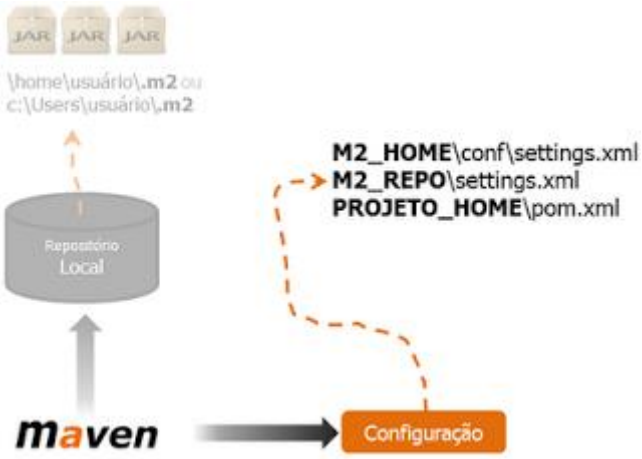
1 - Repositório Local:

Lendo o arquivo pom.xml do projeto, o maven tenta encontrar localmente as dependências. Quando esta busca não encontra um determinado JAR, o maven tenta obter nos próximos repositórios (mais detalhes adiante) uma cópia para o repositório local. Desta maneira, em uma próxima execução o JAR será encontrado. Em outras palavras, o repositório local deverá conter todos JARs, os quais o seu projeto faz referência. Mas isto não significa que você precisa criá-lo manualmente.



Para procurar em outros repositórios, existe uma configuração que especifica os mesmos.

Existem pelo menos 3 formas de fazer esta configuração:
- Configurar no diretório de instalação do Maven (M2_HOME) o arquivo settings.xml
- Configurar no diretório do repositório local do usuário (M2_REPO) também o arquivo settings.xml
- Ou por último, diretamente no pom.xml do projeto.



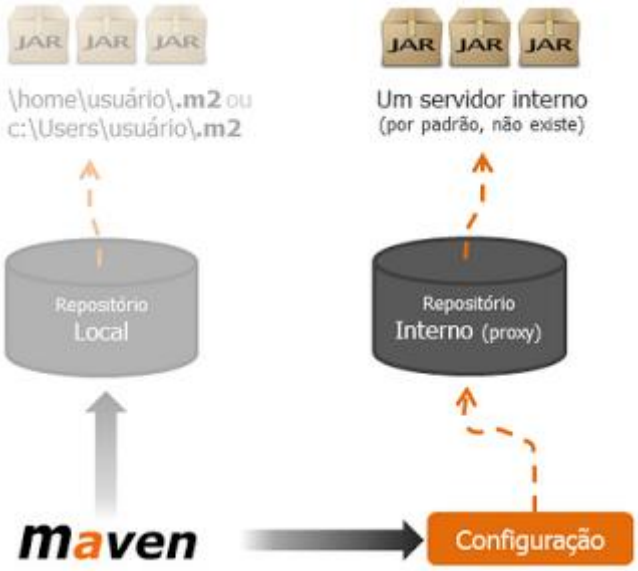
Existem 2 tipos de repositórios Maven: Interno e Externo.

2 - Repositório Interno

É um repositório criado dentro da rede da empresa. Este repositório não será detalhado neste post, mas é importante saber que ele existe e que o seu uso tem algumas vantagens.

A ideia aqui é não depender da internet para baixar as dependencias, bem como ter um lugar para guardar os artefatos (bibliotecas) que a empresa produz (que não estão na internet).

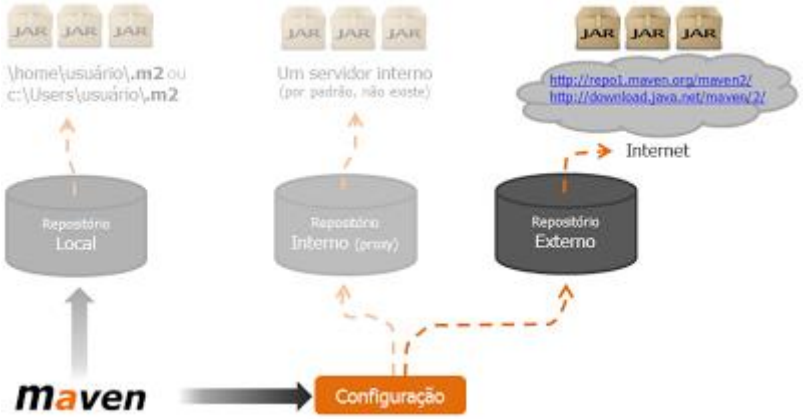
Um exemplo de repositório interno é o Nexus



3 - Repositório Externo

Basicamente é buscar JARs na internet em repositórios Maven públicos.

Obs: Algumas vezes, é necessário adicionar mais repositórios para encontrar os JARs do seu projeto. Um único repositório, pode não conter todas as dependências necessárias.



Exemplos de repositórios Maven na internet:

http://repo1.maven.org/maven2/

http://download.java.net/maven/2/

http://google-maven-repository.googlecode.com/svn/repository/

http://repository.jboss.org/nexus/content/groups/public-jboss/

http://nexus.codehaus.org/snapshots/

http://repository.apache.org/snapshots/

http://www.ibiblio.org/maven/

O ciclo de vida do processo build

Agora com um projeto Maven já preparado, vamos para a principal funcionalidade: o build. O build do Maven é baseado no conceito de ciclo de vida: o processo de construção e distribuição da sua aplicação é dividido em partes bem definidas chamadas fases, seguindo um ciclo. O ciclo padrão é o seguinte:

compile – compila o código fonte do projeto

test – executa os testes unitários do código compilado, usando uma ferramenta de testes unitários, como o junit.

package – empacota o código compilado de acordo com o empacotamento escolhido, por exemplo, em JAR.

integration-test – processa e faz o deploy do pacote em um ambiente onde os testes de integração podem ser rodados.

install – instala o pacote no repositório local, para ser usado como dependência de outros projetos locais

deploy – feito em ambiente de integração ou de release, copia o pacote final para um repositório remoto para ser compartilhado entre desenvolvedores e projetos

Você pode invocar qualquer dessas fases na linha de comando, digitando:

mvn [fase]

Por exemplo se você digitar mvn package o Maven vai executar todas as fases anteriores do ciclo até a fase package. Uma lista completa das fases do ciclo de vida possíveis pode ser encontrada aqui.

Algumas das fases do ciclo possuem plugins associadas a elas, e esses plugins são executados assim que a fase é chamada para ser executada. Você pode também registrar plugins para rodarem em qualquer fase do ciclo, conseguindo, assim, personalizar o build do seu projeto facilmente.

Principais Conceitos Básicos

Project Object Model (POM)

Cada projeto Maven fornece um arquivo pom.xml que captura dependências, propriedades de estrutura do projeto, tarefas da fase de construção e comportamento. A maioria das propriedades POM possui padrões que resultam em um mecanismo compacto, porém poderoso, para descrever o comportamento da construção do projeto.

Fases de construção e o ciclo de vida de construção

Um ciclo de vida de construção consiste em várias fases. Quando um comando de fase é fornecido, o Maven executa cada fase na sequência até e incluindo a fase definida. Após o arquivo pom.xml ser definido, as ferramentas Maven priorizam as fases de construção específicas e reagem às fases de validação, geração de códigos, conjunto de recursos e compilação.

Um ciclo de vida de construção consiste nas seguintes fases:

validar

compilar

testar

empacotar

teste de integração

verificar

instalar

implementar

Mais informações sobre o ciclo de vida de construção podem ser localizadas em http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

Objetivo

Um objetivo representa uma tarefa específica que é melhor do que uma fase de construção e que contribui para a criação e o gerenciamento de um projeto.

Pacote

A configuração do empacotamento designa um conjunto de objetivos padrão. Exemplos de valores de empacotamento válidos incluem jar, war, ear e pom.

Plug-in Maven

Um plug-in descreve um conjunto de objetivos que estão ligados a um esquema ou processo de empacotamento específico.

Mojo

Uma tarefa específica que é implementada dentro de um plug-in. Por exemplo, uma implementação de classe Java™ para implementar em seu ambiente de tempo de execução preferencial.

Archetype

Archetypes são utilizados como modelos de projeto para configurar novos projetos. Esses modelos tornam mais fácil ativar padrões dentro de sua organização, definindo objetivos de empacotamento, configurações de plug-in e dependências predefinidas para as bibliotecas padrão.

Repositórios Maven

Repositórios são utilizados para armazenar artefatos de construção e dependências de tipos variados. Por exemplo, archetypes, plug-ins, e arquivos JAR, entre outros. Repositórios locais são preenchidos vagarosamente, conforme necessário, a partir de repositórios remotos para propósitos de construção.