## 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:

**Felipe Ferreira;**

**Marcus Daniel**

**Pedro Henrique;**

**Raif André.**

# Gerência de Configuração de Software

## AULA 10

**Debate sobre o Maven**

**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.

# Repositório

## - 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.

## - 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).

## - 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:

# 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

## 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.