**Java Platform, Enterprise Edition (JEE)**

A Java EE (Java Platform, Enterprise Edition) é uma plataforma padrão para desenvolver **aplicações Java de grande porte e/ou para a internet**, que **inclui bibliotecas e funcionalidades** para implementar software Java distribuído, baseado em **componentes modulares** que executam em **servidores de aplicações**.

A plataforma Java EE possui uma série de especificações (tecnologias) com objetivos distintos. Entre as especificações da Java EE, as mais conhecidas são:

* **JSF (JavaServer Faces):** é um framework web baseado em Java que tem como objetivo simplificar o desenvolvimento de interfaces (telas) de sistemas para a web, através de um modelo de componentes reutilizáveis.
* **JPA** **(Java Persistence API**): é uma API padrão do Java para persistência de dados, que usa um conceito de mapeamento objeto-relacional.
* **EJB** (**Enterprise Java Beans**): são componentes que executam em servidores de aplicação.

**Maven**

O Maven é uma ferramenta de gerenciamento, construção e implantação de projetos, que ajuda no processo de gerenciamento de dependências e no de build, geração de relatórios e de documentação.

* **Gerenciamento**
* Gerenciamento de Dependências
* Gerenciamento de Builds
* Gerenciamento de Relatórios
* Gerenciamento de Documentação
* **Construção**
* **Implantação**

**Configuração, Instalação e Teste**

**Maven**: <http://maven.apache.org/download.html>

**Java 8**: <http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html>

**Configurar variáveis de ambiente**

**M2\_HOME**="Diretório/de/instalação"

**JAVA\_HOME**="Diretório/de/instalação"

Adicionar as variáveis acima ao "Path" do sistema acrescidos do diretório "bin".

**Ex.:**

**WINDOWS**: ...;%M2\_HOME%\bin

**LINUX**: PATH=$PATH:$M2\_HOME\bin

**Verificar funcionamento**

No terminal (CMD) executar o seguinte comando:

**java -version**

**mvn -version**

**Estrutura**

A unidade básica de configuração do Maven é um arquivo chamado **pom.xml**, que deve ficar na raiz do projeto. Ele é um arquivo conhecido como **Project Object Model**. Nesse é arquivo é declarado a estrutura, dependências e características do projeto.

**<project>**

**<modelVersion>**4.0.0**</modelVersion>**

**<groupId>**br.ufpi**</groupId>**

**<artifactId>**teste-app**</artifactId>**

**<version>**0.0.1-SNAPSHOT**</version>**

**<packaging>**war**</packaging>**

**</project>**

O arquivo contém inicialmente apenas a identificação do projeto, e uma informação a mais**: modelVersion**, que é a identificação da versão do arquivo **pom.xml**. A identificação do projeto consiste em três informações:

* **groupId:** Um identificador da empresa/grupo ao qual o projeto pertence. Geralmente o nome do site da empresa/grupo ao contrário. Ex: br.ufpi.
* **artifactId**: O nome do projeto. Ex: teste-app.
* **version**: A versão atual do projeto. Ex: 0.0.1-SNAPSHOT.
* **packaging**: A forma de empacotamento do projeto. Ex.: war, para projetos web.

**Criação de um projeto web utilizando Maven**

**Eclipse:**

**File** => **New** => **Maven Project** => Selecionar opção para "**simple project**" para não usar nenhum ArcheType (planta) de projeto => Adicionar informações (**ArtifactId**, **GroupId**, **Version**, **Package**)

**Linha de Comando:**

mvn archetype:generate

-DarchetypeGroupId=org.apache.maven.archetypes

-DarchetypeArtifactId=maven-archetype-webapp

-DgroupId=com.mycompany.app

-DartifactId=my-webapp

**Ciclo de vida**

Uma das principais funcionalidades do Maven é o **build**. O build 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 WAR.

**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

**Obs.:** O ciclo *default* acima não se limita apenas as estas etapas, vide referência de documentação**:** <http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference>

**Escopos**

A tag "**scope**" de uma dependência explicita o escopo em que o artefato definido será usado. Alguns dos principais escopos utilizados, são:

**compile**: escopo padrão do maven para o momento em que o código é compilado e vai junto com o artefato;

**provided**: adicionado no momento da compilação, mas não vai junto com o artefato. Dessa forma você está dizendo que a dependência virá de maneira transitiva de alguma outra dependência;

**runtime**: não inclui no artefato, pois estará disponível em tempo de execução;

**test**: inclui apenas no escopo de testes;

**system**: não inclui no artefato, pois estará disponível no ambiente.

**Plugins**

**maven-compile-plugin**

* **Documentação**: <http://maven.apache.org/plugins/maven-compiler-plugin/>
* **Referência de uso**: <http://maven.apache.org/plugins/maven-compiler-plugin/examples/set-compiler-source-and-target.html>

**jboss-as-maven-plugin**

* **Documentação**: <https://docs.jboss.org/jbossas/7/plugins/maven/latest/>
* **Referência de uso**: <https://docs.jboss.org/jbossas/7/plugins/maven/latest/examples/deployment-example.html>

**maven-glassfish-plugin**

* Documentação: <https://maven-glassfish-plugin.java.net/>
* Referência de uso: <https://maven-glassfish-plugin.java.net/examples/complete.html>

**Referências Extras**

**Plug-ins Maven**: <http://maven.apache.org/plugins/>

**Livro de apoio Java EE**: <http://cafe.algaworks.com/livro-java-ee-7-com-jsf-primefaces-e-cdi/>