**MAVEN**

* Sempre na primeira vez, o Maven faz download das dependências para cada comando que estamos utilizando (compile, test, etc)
* Maven é utilizado para gerenciar as dependências, executar testes, gerar relatório, fazer o build da aplicação
* O diretório /m2/repository é o repositório local que será utilizado por outros projetos que utilizam as mesmas bibliotecas para que não precise fazer download novamente
* O PMD é um analisador de código-fonte que consegue encontrar algumas falhas no nosso código, como por exemplo, variáveis que não estão sendo utilizadas.
* Utilize o plugin PMD do Maven para gerar um relatório da análise do código-fonte do seu projeto. Após executar o comando, você encontrará o relatório dentro do diretório *target/site*, no arquivo pmd.html. (**https://maven.apache.org/plugins/maven-pmd-plugin/**)
* Jacoco - (https://www.eclemma.org/jacoco/trunk/doc/maven.html) plugin para gerar um relatório mais robusto do código
* Relatórios sobre a cobertura de testes do projeto. Esse tipo de relatório é interessante pois é possível ver de uma forma fácil o que não testamos em nossa aplicação.

**SCOPE**

* O Scope padrão de uma dependência é o **COMPILE** que indica que a dependência estará disponível em todos os classpaths: de compilação de teste e execução. Esse é o escopo padrão utilizado quando não declaramos um escopo na dependência.
* O Scope de **TEST** significa que a dependência não será colocada no pacote final pois ela é utilizada somente pra test
* O Scope **PROVIDE** significa que a dependência é utilizada pra compilar e testar o projeto, mas não será utilizada no momento de gerar o pacote final do projeto
* O Scope **RUNTIME** significa que a dependência será utilizada somente no momento de execução - A dependência estará no classpath de runtime e de tests mas não de compilação.

\*\* *É uma boa pratica utilizar o clean antes de realizar o package*

**COMMANDS**

* **mvn compile** -- Gera os arquivos .class no folder target/classes
* **mvn -o compile** -- Compila o projeto em modo offline, ou seja, não irá no repositório online verificar se tem atualização (utilizado se o projeto já tem todas as dependências e não precisa ser atualizado), é mais rápido
* **mvn test** -- Executa os testes e gera um relatório(xml e txt) na pasta target
* **mvn clean** -- Remove o diretório target com os .class do projeto e dependências
* **mvn surefire-report:report** -- Gera um relatório dos tests em formato html (precisa colocar o plugin no pom.xml)
* **mvn package** -- Executa os tests e gera o arquivo .jar ou .war dependendo do que está configurado no pom.xml
* **mvn pmd:pmd** -- Gera um relatório de análise de código, mostrando o que não precisa ser utilizado etc
* **mvn pmd:check** -- Vai executar a validação de código e se estiver algo com problema nao irá fazer o build
* **mvn install** -- Irá atualizar o repositório local com uma lib que está na dependência, mas não está em um repositório local e nem em um remoto

**Ciclo de vida/fases de um projeto Maven (deploy executa todas as fases anteriores):**

* **validate**: validate the project is correct and all necessary information is available
* compile: compile the source code of the Project;
* **test**: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed;
* **package**: take the compiled code and package it in its distributable format, such as a JAR;
* **integration-test**: process and deploy the package if necessary into an environment where integration tests can be run;
* **verify**: run any checks to verify the package is valid and meets quality criteria;
* **install**: install the package into the local repository, for use as a dependency in other projects locally;
* **deploy**: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects;
* **clean**: cleans up artifacts created by prior builds;
* **site**: generates site documentation for this Project;

**MAVEN POJETO WEB**

* Criar um projeto maven como archetype webapp (https://maven.apache.org/guides/introduction/introduction-to-archetypes.html)
* Colocar a dependência do servlet 3.1(+) para que o eclipse não apresente erro. Precisa colocar o scope como provide para que a dependência não seja colocada no artefato final (Na pratica os servidores já possuem no classpath o servlet)
* Atualizar o web.xml apontando para a versão do servlet correta (maven cria o web.xml apontando para uma versão antiga)