

Gerenciamento de Dependências e Build com Maven

Willyan Guimarães Caetano
Desenvolvedor

Mais sobre mim



- Atuação na área desde 2012
- Grande experiência com projetos em Java
- Certificação em Java versão 8 (OCP)
- Atualmente trabalha em uma empresa setor financeiro
- <https://www.linkedin.com/in/willyancaetanodev>

Objetivo do curso

Você será capaz

- Criar um projeto utilizando a ferramenta
- Entender os principais conceitos por trás do Maven
- Gerenciar dependências do seu projeto
- Configurar plugins e projetos com necessidades específicas

Percurso

Aula 1

Definição e Instalação

Aula 2

Primeiro projeto e conceitos

Aula 3

POM, Dependências e repositórios

Percurso

Aula 4

Gerenciando Dependências

Aula 5

Maven Build Lifecycle

Aula 6

Multi-módulos



Percurso

Aula 7 Plugins



Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Aula 1: Definição e instalação

Gerenciamento de
Dependências e Build com
Maven

Objetivos

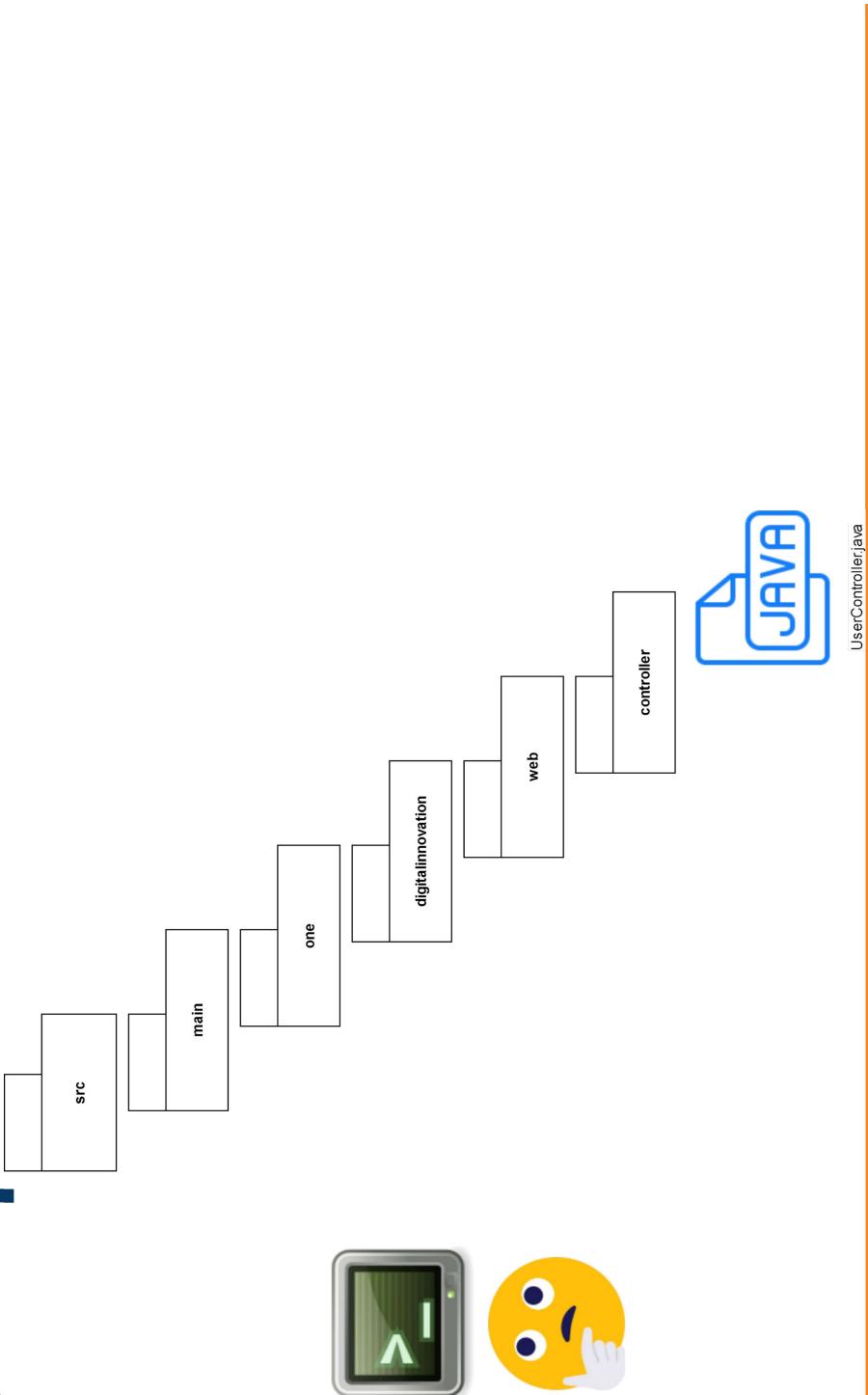
- 1.** Entender o que é o Apache Maven e sua utilidade
- 2.** Entender como se executa a instalação do Maven

Parte 1: O que é Apache Maven?

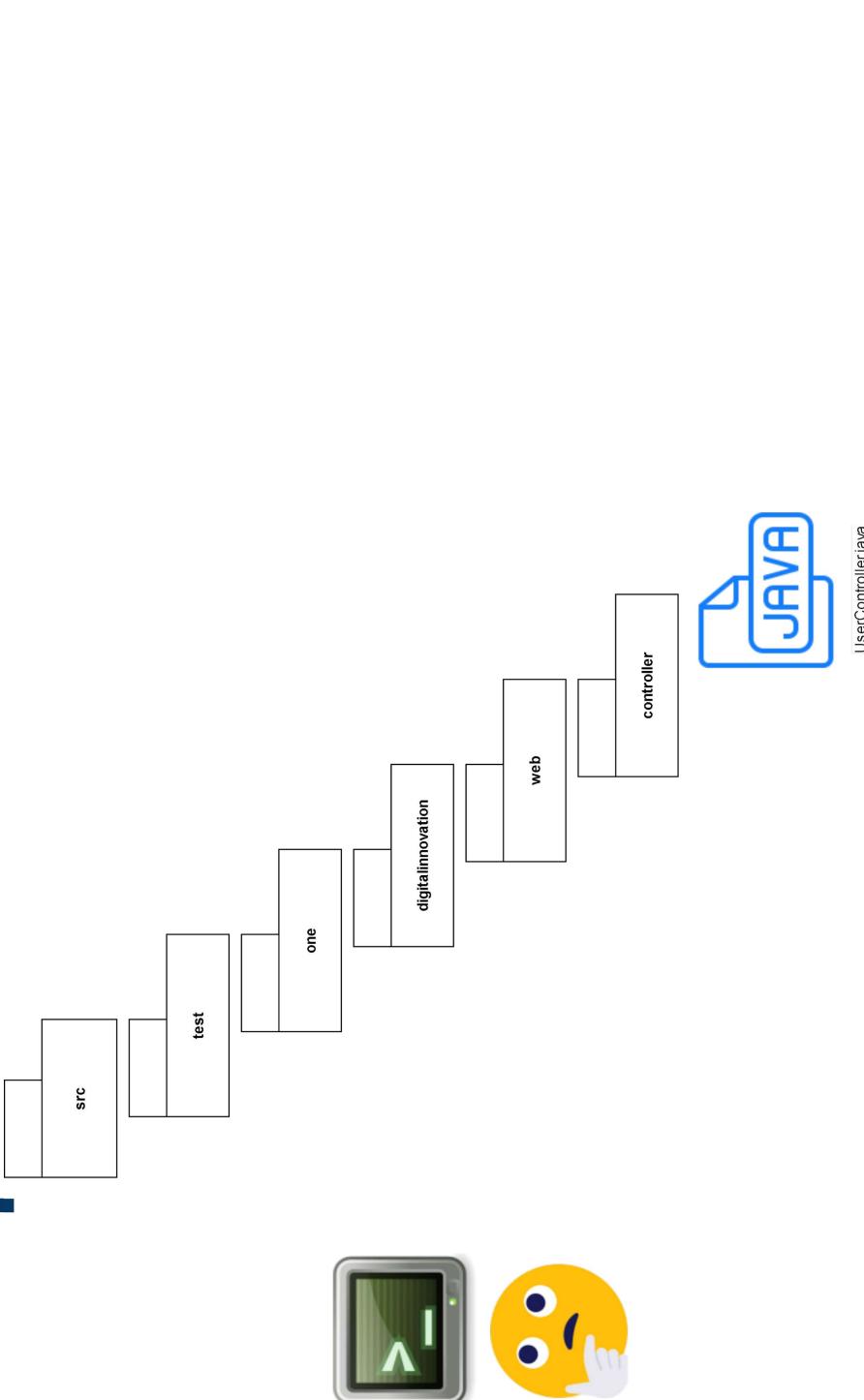
O que é o Maven?

- Ferramenta para gerenciar build e dependências de um projeto
- Primeira versão em julho de 2004, mantido pela Apache Software Foundation

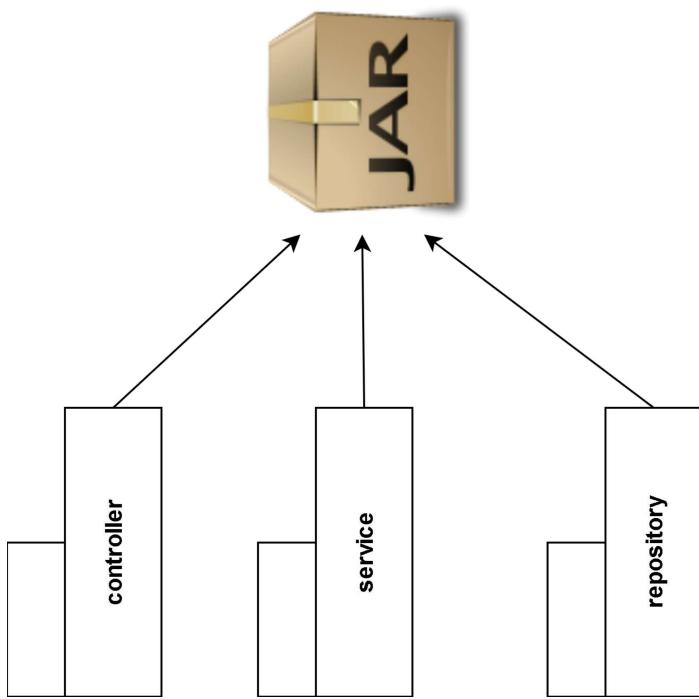
Qual problema ele resolve?



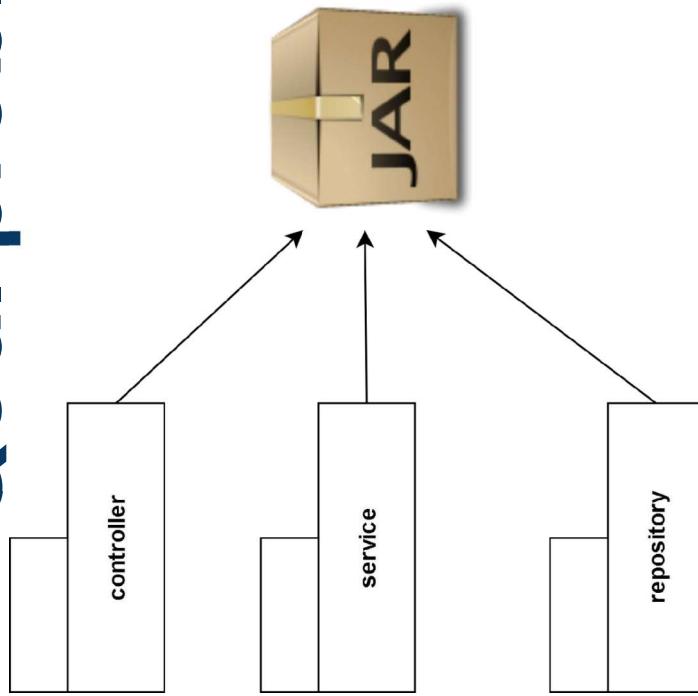
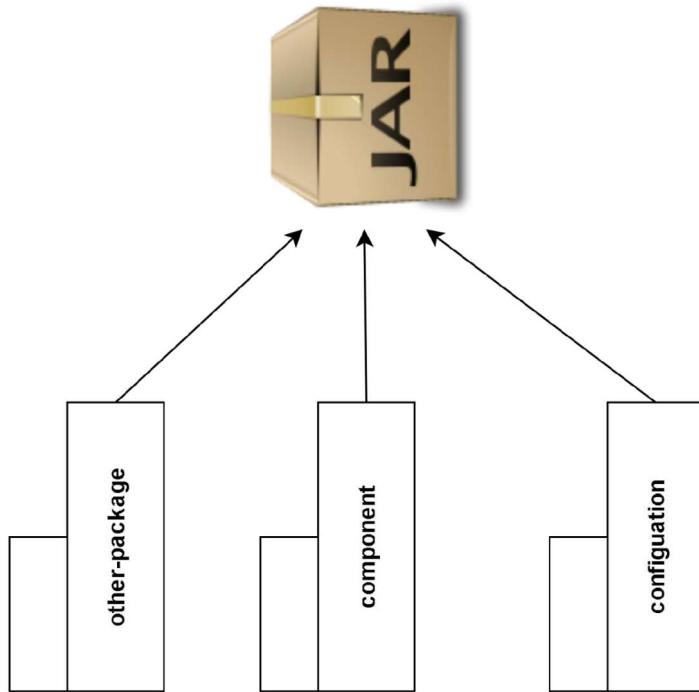
Qual problema ele resolve?



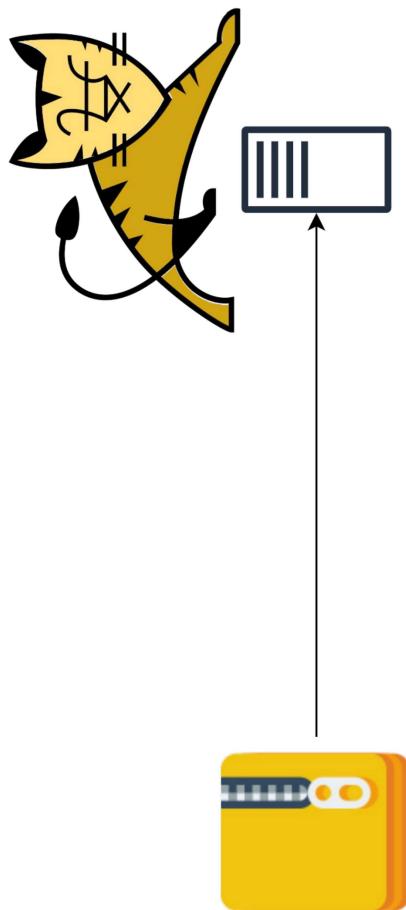
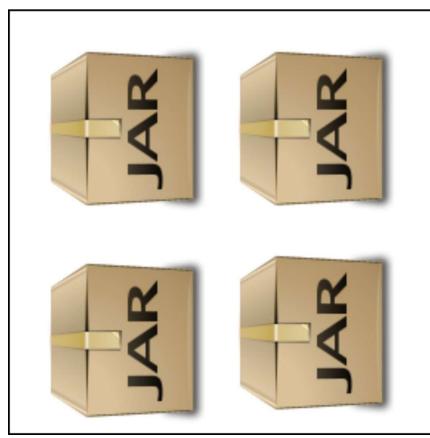
Qual problema ele resolve?



Qual problema ele resolve?



Qual problema ele resolve?



—



O que é Maven?

- Endereça como o software foi construído e suas dependências através do POM(Project Object Model)
- Facilita a compreensão do desenvolvedor
- Fornecer informações de qualidade

Para saber mais

- https://en.wikipedia.org/wiki/Apache_Maven
- <https://maven.apache.org/what-is-maven.html>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 2: Instalação e Configuração

Pré-requisitos

- JDK Instalado, nos exemplos iremos utilizar versão 11 no exemplo, verifique rodando o comando:

```
javac -version
```

Instalação

- Baixar pacote do site oficial do Apache Maven:
<https://maven.apache.org/>
- Descompactar em um diretório

Configuração

- No Windows
 - Adicionar no Path pelo Painel Controle > Sistema e Segurança > Sistema > Configurações avançadas do sistema > Avançado > Variáveis de ambiente
- No Linux
 - Adicionar no PATH

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Aula 2: Primeiro projeto e conceitos

Gerenciamento de Dependências e Build com Maven

Objetivos

- 1.** Entender como criar um projeto usando o Maven
- 2.** Aprender alguns comandos que auxiliam no dia a dia
- 3.** Criando diferentes tipos de projeto

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 1: Criando um projeto via linha de comando

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 2: Comandos que auxiliam o dia a dia

Comandos

- 1. Compilar: compile**
- 2. Testar: test**
- 3. Empacotar: package**
- 4. Limpar diretório de trabalho: clean**

Para saber más

- <https://www.baeldung.com/maven-compiler-plugin>
- <https://mkyong.com/maven/how-to-run-unit-test-with-maven/>
- <http://tutorials.jenkov.com/maven/maven-commands.html>

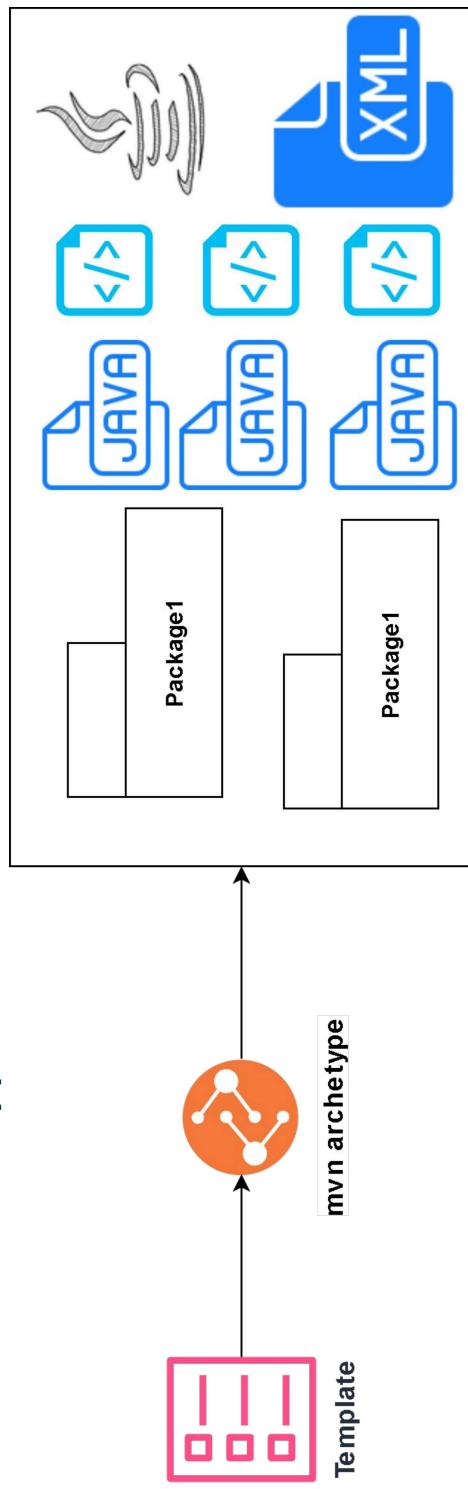
Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 3: Criando diferentes tipos de projeto

Como?

- Maven archetype



Como ?

- Pesquise na internet por “maven archetype list” para encontrar a opção que faz mais sentido pro seu cenário
- Procure pelas instruções de execução do comando mvn archetype

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Aula 3: POM, Dependências e Repositórios

Gerenciamento de
Dependências e Build
com Maven

Objetivos

- 1.** Entender a estrutura do POM e sua função para o Maven
- 2.** Entender sobre repositórios e seus tipos
- 3.** Entender como configurar uma nova dependência no projeto

Parte 1: O POM

Entendendo o POM

- POM – Project Object Model
- Unidade fundamental de trabalho
- Formato XML
- Detalha o projeto
- Detalha como construir o projeto
- Maven sempre procura pelo pom.xml para realizar sua execução

Mais detalhes pom.xml

- Nome do projeto
- Dependências
- Módulos
- Configurações de build
 - Detalhes do projeto (nome, descrição, licença, url)
 - Configurações de ambiente (repositórios, tracking, profiles)
- Exemplo

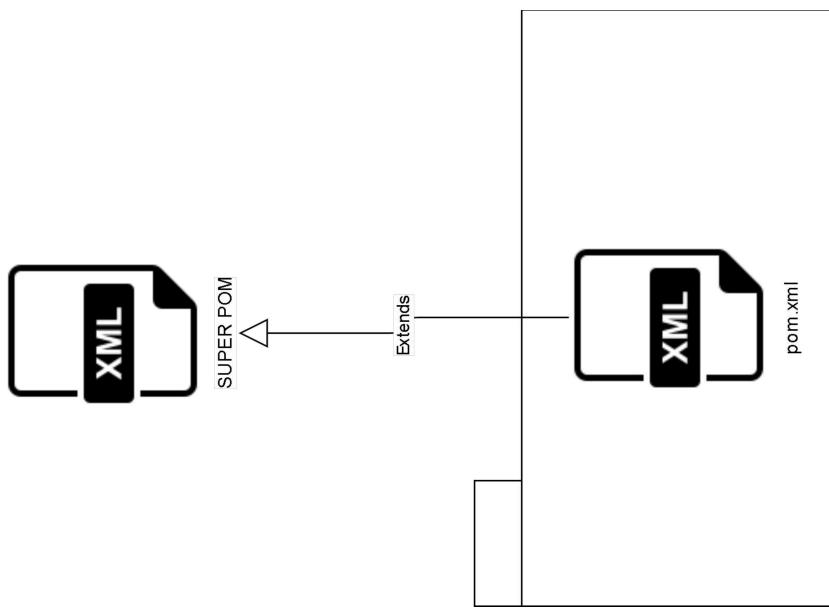
Pom.xml básico

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1</version>
</project>
```

E as outras configurações ?



O Super POM



Para saber mais

- <https://maven.apache.org/pom.html>
- <https://maven.apache.org/ref/3.8.2/maven-model-builder/super-pom.html>
- <https://maven.apache.org/guides/introduction/introduction-to-the-pom.html>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 2: Repositórios

O que são repositórios?

- São locais onde podemos encontrar plugins e bibliotecas que o Maven provê
- Dois tipos: Local e Remoto

Repositório remoto

- É o local central utilizado pelo Maven para buscar os artefatos.
- Configurado automaticamente pelo Super POM para utilizar o Maven Central

Repositório remoto

```
<repositories>
  <repository>
    <id>central</id>
    <name>Central Repository</name>
    <url>http://repo.maven.apache.org/maven2</url>
    <layout>default</layout>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
```

Repositório remoto

```
<pluginRepositories>
  <pluginRepository>
    <id>central</id>
    <name>Central Repository</name>
    <url>http://repo.maven.apache.org/maven2</url>
    <layout>default</layout>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
    <releases>
      <updatePolicy>never</updatePolicy>
    </releases>
    </pluginRepository>
  </pluginRepositories>
```

Configuração

- Via pom.xml do projeto

```
<project>
  ...
    <repositories>
      <repository>
        <id>my-repo1</id>
        <name>your custom repo</name>
        <url>http://jarsm2.dyndns.dk</url>
      </repository>
      <repository>
        <id>my-repo2</id>
        <name>your custom repo</name>
        <url>http://jarsm2.dyndns.dk</url>
      </repository>
    </repositories>
  ...
</project>
```

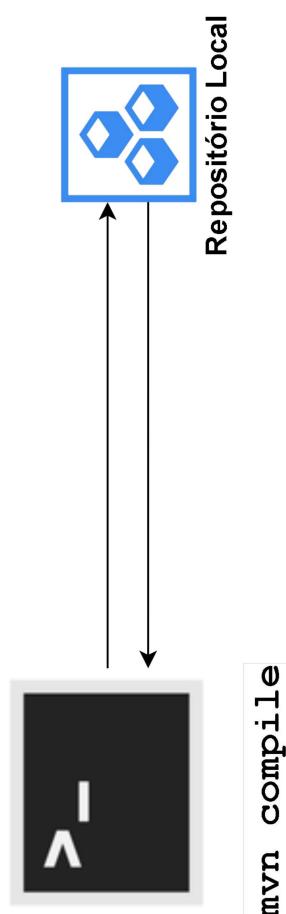
Configuração

- Via `settings.xml`
- Localização: `pasta_apache_maven/conf/settings.xml`

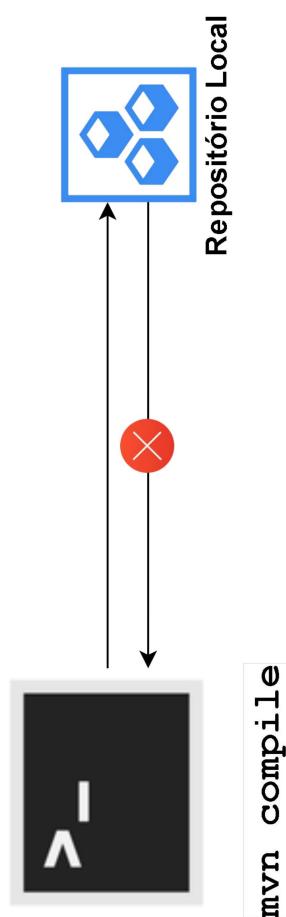
Repositório local

- É o repositório na máquina utilizado pelo Maven para buscar os artefatos.
- Estratégia de caching
- Localizações
 - Windows: %USERPROFILE%\.m2\repository
 - Linux: \$HOME/.m2/repository

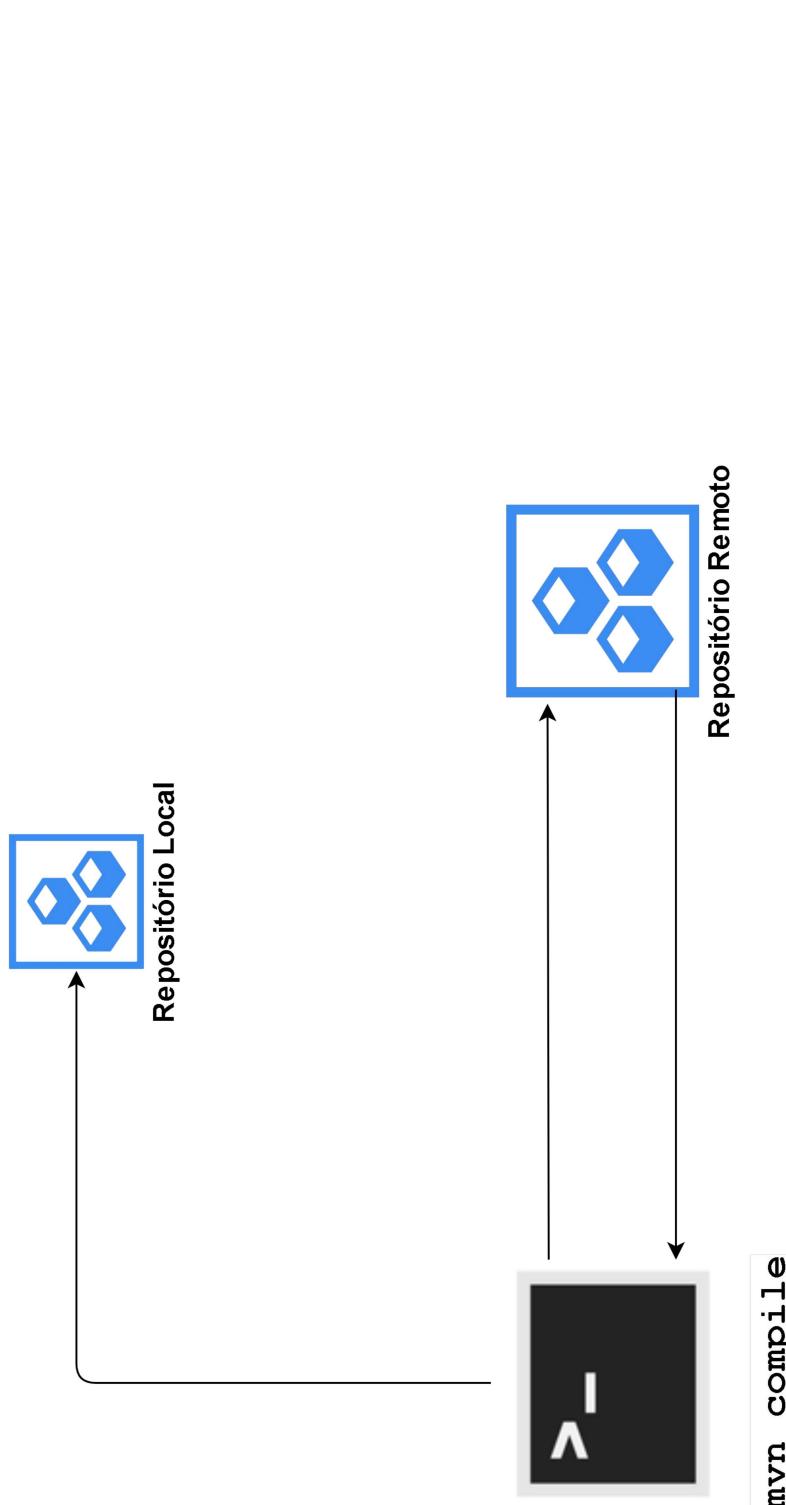
Como isso funciona?



Como isso funciona?



Como isso funciona?



Para saber más

- <https://maven.apache.org/guides/introduction/introduction-to-repositories.html>
- <https://maven.apache.org/ref/3.0.4/maven-model-builder/separator-pom.html>
- <https://repo.maven.apache.org/maven2/>
- <https://mvnrepository.com/>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 3: Como adicionar dependências

Configurando

```
...<dependencies>
<dependency>
<groupId>one.digitalinnovation</groupId>
<artifactId>app-spring</artifactId>
<version>1.0.0</version>
</dependency>
...
...
```

Propriedades

- groupId: É como se fosse o id da organização. Segue as regras de nomes de pacote Java
- artifactId: Nome do projeto em si
- Version: Número da versão que será utilizada

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-search-orm</artifactId>
    <version>5.11.9.Final</version>
</dependency>
```



hibernate.org/search/

Na prática

- Adicionando uma dependência ao nosso projeto
- Acompanhando o download na pasta .m2

Para saber más

- <https://docs.oracle.com/javase/specs/jls/se6/html/packages.html#7.7>
- <https://maven.apache.org/guides/mini/guide-naming-conventions.html>
- <https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html>
- <https://mvnrepository.com/>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Aula 4: Gerenciando dependências

Gerenciamento de Dependências e Build com Maven

Objetivos

- 1.** Entender os tipos de dependência que existem
- 2.** Compreender transitividade e escopos

Parte 1: Tipos de dependências

Um cenário

Quando baixamos uma dependência

Tipos de Dependências

- Direta: dependências declaradas no pom.xml
- Transitiva: dependências obrigatórias das dependências declaradas no pom.xml

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 2: Transitiività e Escopos

Um cenário

O problema de dependências transitivas

Escopos

Para lidar com esse problema, o Maven provê escopos para limitar a transitividade das dependências. Existem 6 tipos de escopos que podemos utilizar.

Classpath

- Runtime
- Test
- Compile

Escopo compile

- Escopo default
- Disponível em todos os classpaths
- É transitivo

Escopo provided

- Indica que a dependência será fornecida em tempo de execução por uma implementação na JDK ou via container
- Exemplos: Servlet API, Java EE APIs
 - A dependência com esse escopo é adicionado no classpath usado para compilação(compile) e teste(test) mas não em runtime;
 - Não é transitiva

Escopo provided

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
```

Escopo runtime

- Indica que a dependência é necessária para execução e não para compilação
- Maven inclui no classpath dos escopos de runtime e test

Escopo runtime

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>6.0.6</version>
    <scope>runtime</scope>
</dependency>
```

Escopo test

- Disponível somente para compilação e execução de testes
- Não é transitivo

Escopo test

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>${junit.jupiter.version}</version>
    <scope>test</scope>
</dependency>
```

Escopo system

- Similar ao escopo provided exceto por ser necessário prover o JAR explicitamente
- A dependência com esse escopo é adicionado no classpath usado para compilação(compile) e teste(test) mas não em runtime;
- Não é transitiva

Escopo system

```
<dependency>
    <groupId>com.baeldung</groupId>
    <artifactId>custom-dependency</artifactId>
    <version>1.3.2</version>
    <scope>system</scope>
    <systemPath>${project.basedir}/libs/custom-dependency-1.3.2.jar</systemPath>
</dependency>
```

Escopo import

- Este escopo é disponível apenas com uma dependência do tipo pom e com tag <dependencyManagement>
- Indica um processo de reutilizar dependências de um projeto

Escopo import

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.programmergirl</groupId>
      <artifactId>my-project</artifactId>
      <version>1.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Para saber mais

- <https://www.baeldung.com/maven-dependency-scopes>
- [https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency Scope](https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency_Scope)
- <https://www.baeldung.com/maven-optional-dependency>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Parte 3: Dica sobre escopos, dependências, opcionais e exclusões

Ver o classpath

```
mvn dependency:build-classpath -DincludeScope=compile  
mvn dependency:build-classpath -DincludeScope=test  
mvn dependency:build-classpath -DincludeScope=runtime
```

Dependências Opcionais

- Utilizado quando uma dependência não é necessária para os projetos que irão reutilizar seu componente

```
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.8</version>
    <optional>true</optional>
</dependency>
```

Exclusions

- Utilizado quando o componente que você usa compartilha uma biblioteca que você já tem ou não quer ter disponível

Exclusions

```
<dependency>
  <groupId>one.digitalinnovation</groupId>
  <artifactId>component</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <exclusions>
    <exclusion>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
```

Para saber mais

- <https://maven.apache.org/guides/introduction/introduction-to-optional-and-excludes-dependencies.html>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

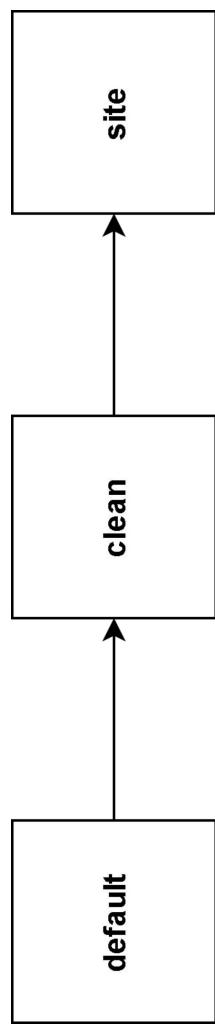
Aula 5: Maven Build Lifecycle

Gerenciamento de Dependências e Build com Maven

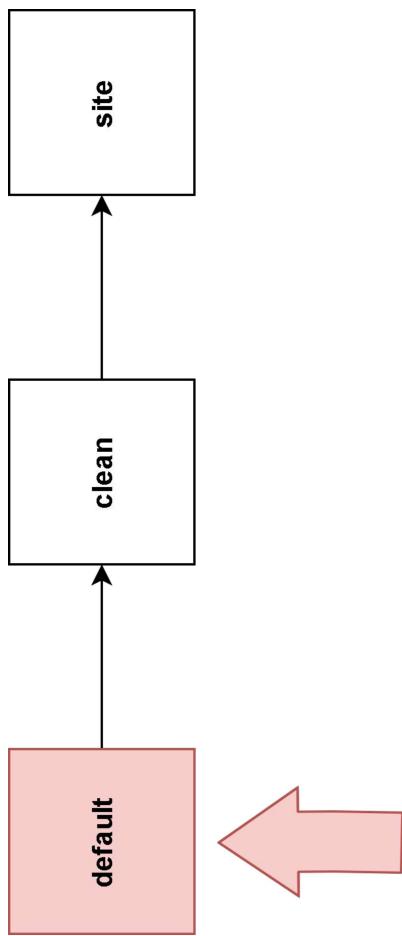
O que é

- Conceito de ciclo de vida de construção
- Conceito e os comandos da ferramenta
- Composto por 3 ciclos de vida
- Cada ciclo possui fases (Maven Phases)
- Cada fase possui objetivos (Maven Goals)

Como é



Default Lifecycle



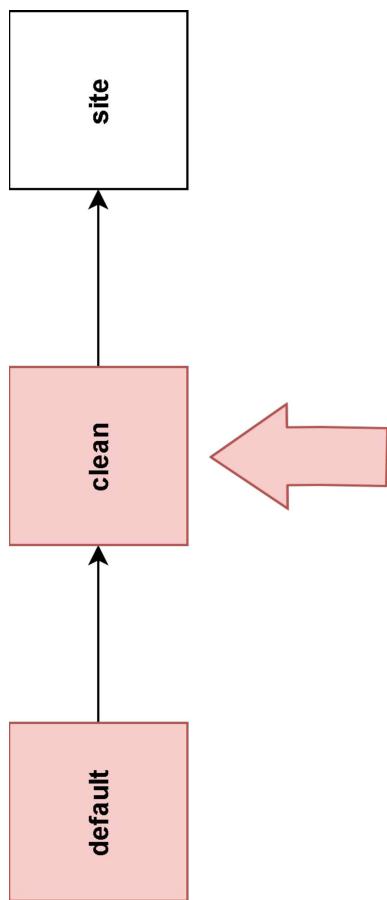
Default Lifecycle

- Principal ciclo
- Responsável pelo deploy local
- Composto por 23 fases

Principais fases

- validate
- compile
- test-compile
- test
- integration-test
- package
- install
- deploy

Clean Lifecycle



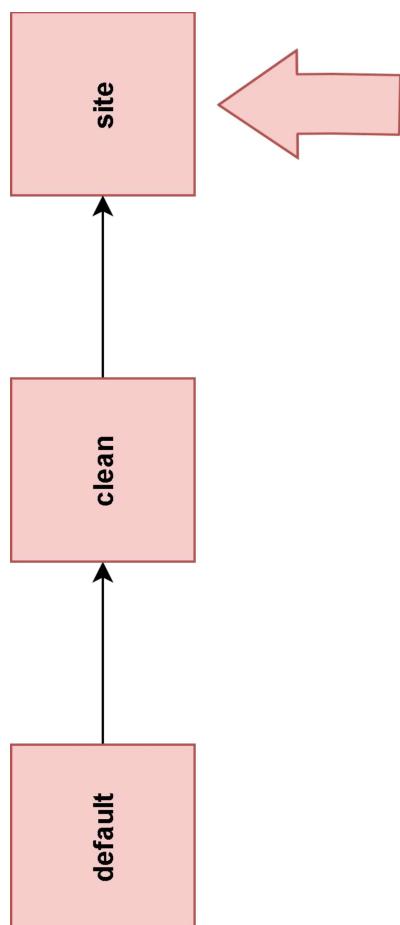
Clean Lifecycle

- Ciclo intermediário
- Responsável pela limpeza do projeto
- Composto por 3 fases

Fases

- pre-clean
- clean
- post-clean

Site Lifecycle



Site Lifecycle

- Ciclo final
- Responsável pela criação do site de documentação do projeto
- Composto por 4 fases

Fases

- pre-site
- site
- post-site
- site-deploy

Para saber más

- <https://medium.com/@andgomes/os-ciclos-de-vida-do-maven-cefc18ba8ff3>
- <https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>
- https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference
- <https://www.baeldung.com/maven-site-plugin>
- <https://www.baeldung.com/maven-goals-phases>

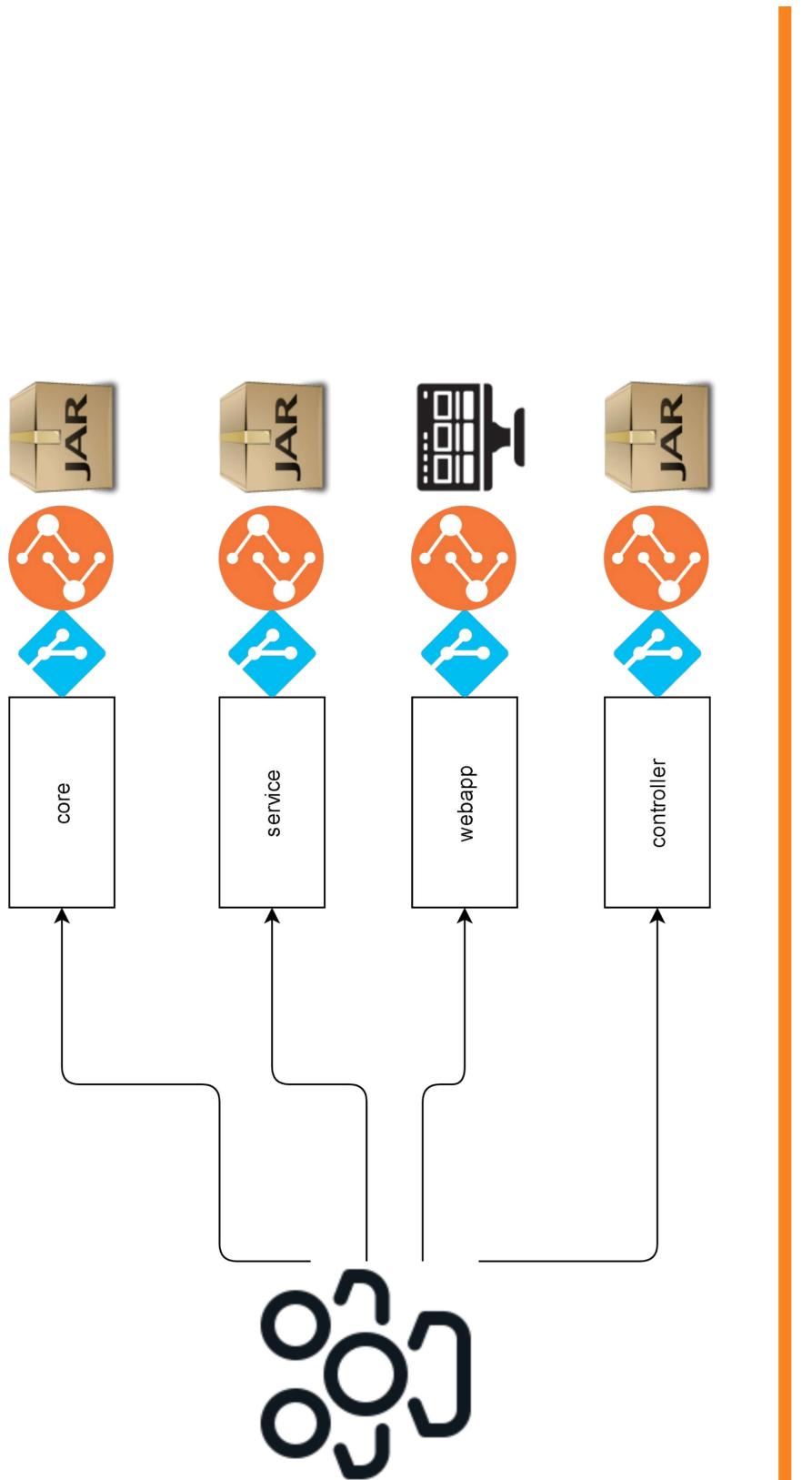
Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

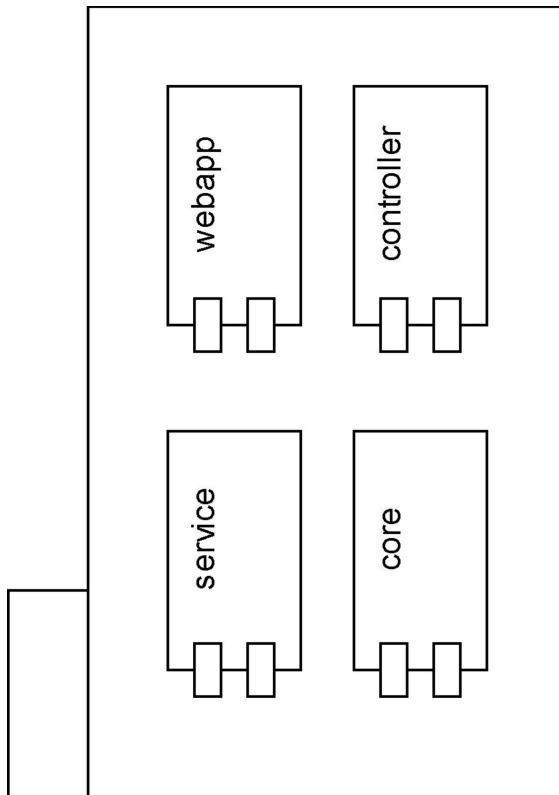
Aula 6: Multi-módulos

Gerenciamento de
Dependências e Build
com Maven

Projetos multi-módulos



Projetos multi-módulos



Para saber mais

- <https://www.baeldung.com/maven-multi-module>
- <https://maven.apache.org/guides/mini/guide-multiple-module-s.html>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

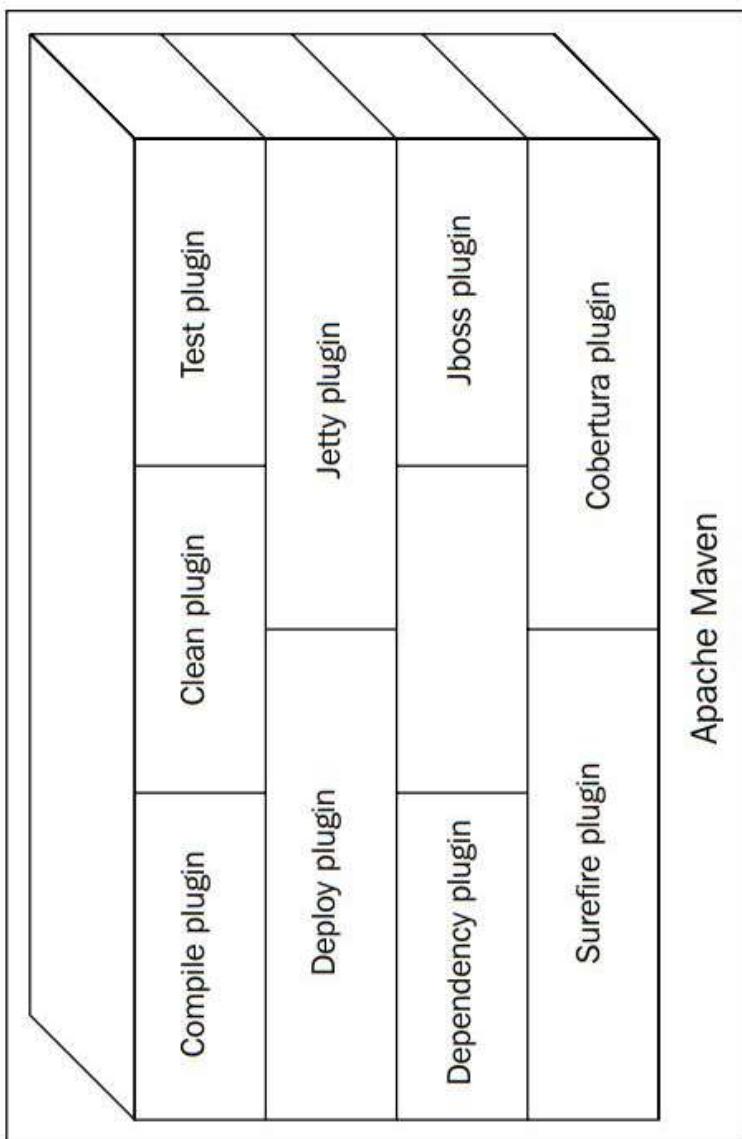
Aula 7: Plugins

Gerenciamento de
Dependências e Build
com Maven

O que são os plugins?

- A maioria das funcionalidades são providas por plugins
- Estilo arquitetural para extensibilidade (criar seu próprio plugin)
- Escrito prioritariamente em Java e disponibilizados comumente como JARs

Plugins Maven



Plugins mais utilizados

- eclipse
- jacoco
- ear
- war
- compile
- clean
- checkstyle
- javadoc

Uso

`mvn [plugin-name]:[goal-name]`

Configuração

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
      <configuration>
        <release>11</release>
      </configuration>
    </plugin>
  </plugins>
```

Na prática

- Gerando Javadoc no projeto

Para saber mais

- <https://maven.apache.org/plugins/>
- <https://maven.apache.org/guides/introduction/introduction-to-plugins.html>

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Gerenciamento de Dependências e Build com Maven

Willyan Guimarães Caetano
Desenvolvedor

Para profundar

- <https://maven.apache.org/guides/>
- Apache Maven 3 Cookbook, Srirangan, 2011, Packt Publishing

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)