

# Gradle

---

Fernando Camargo

11 de julho de 2018

ZG Soluções

**Por que uma ferramenta de Build?**

---

## Por que uma ferramenta de Build?

- Projeto independente de IDE

## Por que uma ferramenta de Build?

- Projeto independente de IDE
- Automação de build

# **Ant, Maven e Gradle**

---

- Primeira build tool para Java

- Primeira build tool para Java
- Extrema flexibilidade

- Primeira build tool para Java
- Extrema flexibilidade
- Não impõe convenções em projetos Java



# Exemplo de Ant

```
<project>

  <target name="clean">
    <delete dir="build"/>
  </target>

  <target name="compile">
    <mkdir dir="build/classes"/>
    <javac srcdir="src" destdir="build/classes"/>
  </target>

  <target name="jar">
    <mkdir dir="build/jar"/>
    <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
      <manifest>
        <attribute name="Main-Class" value="oata.HelloWorld"/>
      </manifest>
    </jar>
  </target>

  <target name="run">
    <java jar="build/jar/HelloWorld.jar" fork="true"/>
  </target>

</project>
```

- Flexível demais → projetos não possuem estrutura padrão

## Problemas do Ant

- Flexível demais → projetos não possuem estrutura padrão
- Muito verboso → escreve-se muito para uma build simples

## Problemas do Ant

- Flexível demais → projetos não possuem estrutura padrão
- Muito verboso → escreve-se muito para uma build simples
- Não possui gerenciamento de dependências

- Convenção sobre Configuração → escreve-se pouco para uma build simples

- Convenção sobre Configuração → escreve-se pouco para uma build simples
- Gerenciamento de dependências com resolução de dependências transitivas

## Estrutura de diretórios

| Diretório          | Função                            |
|--------------------|-----------------------------------|
| src/main/java      | Código fonte                      |
| src/main/resources | Recursos não compilados           |
| src/test/java      | Código de testes                  |
| src/test/resources | Recursos não compilados de testes |
| src/main/webapp    | Recursos WEB                      |
| target             | Resultados de build               |

# Exemplo de POM

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

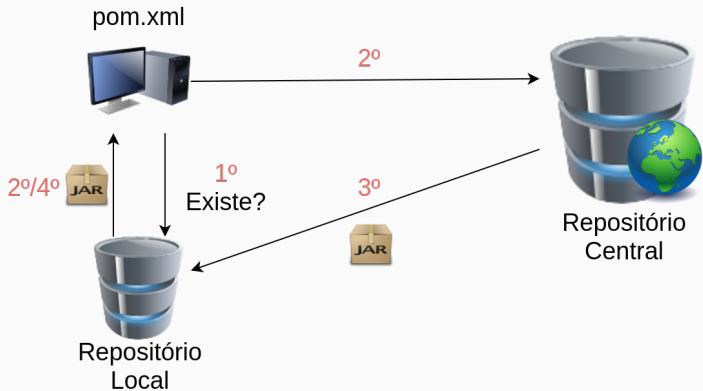
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```



# Repositórios Maven



- Rígido demais

- Rígido demais
- Uso de XML na configuração

- Rígido demais
- Uso de XML na configuração
- Ótimo para 90% dos casos, complicado nos 10% mais específicos

# Gradle

---

## Por que Gradle?

- Combina as partes boas de Ant e Maven
  - Poder e Flexibilidade do Ant
  - Ciclo de vida e facilidade de uso do Maven

## Por que Gradle?

- Combina as partes boas de Ant e Maven
  - Poder e Flexibilidade do Ant
  - Ciclo de vida e facilidade de uso do Maven
- Adiciona uma DSL e outras melhorias

- Flexibilidade



## Características do Gradle

- Flexibilidade
- Completo controle

## Características do Gradle

- Flexibilidade
- Completo controle
- Encadeamento de tarefas

## Características do Gradle

- Flexibilidade
- Completo controle
- Encadeamento de tarefas
- Gerenciamento de dependências

## Características do Gradle

- Flexibilidade
- Completo controle
- Encadeamento de tarefas
- Gerenciamento de dependências
- Convenção sobre configuração

## Características do Gradle

- Flexibilidade
- Completo controle
- Encadeamento de tarefas
- Gerenciamento de dependências
- Convenção sobre configuração
- Projetos multimódulo

## Características do Gradle

- Flexibilidade
- Completo controle
- Encadeamento de tarefas
- Gerenciamento de dependências
- Convenção sobre configuração
- Projetos multimódulo
- Extensível via plugins

## Características do Gradle

- Flexibilidade
- Completo controle
- Encadeamento de tarefas
- Gerenciamento de dependências
- Convenção sobre configuração
- Projetos multimódulo
- Extensível via plugins
- Groovy DSL

## Exemplo de build.gradle

```
apply plugin: 'java'

version = '1.0'

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.11'
}
```



## DSL = Domain Specific Language

- Pequena linguagem para solução de problemas específicos

## DSL = Domain Specific Language

- Pequena linguagem para solução de problemas específicos

Exemplo (SQL): `SELECT * FROM Produtos WHERE ID = 5;`

- Scripts Gradle são **scripts de configuração**

- Scripts Gradle são **scripts de configuração**
  - Execução do script → configuração de um **objeto**

- Scripts Gradle são **scripts de configuração**
  - Execução do script → configuração de um **objeto**
    - Build script → **Project**

- Scripts Gradle são **scripts de configuração**
  - Execução do script → configuração de um **objeto**
    - Build script → **Project**
    - Init script → **Gradle**

- Scripts Gradle são **scripts de configuração**
  - Execução do script → configuração de um **objeto**
    - Build script → **Project**
    - Init script → **Gradle**
    - Settings script → **Settings**

- Scripts Gradle são **scripts de configuração**
  - Execução do script → configuração de um **objeto**
    - Build script → **Project**
    - Init script → **Gradle**
    - Settings script → **Settings**
  - Propriedades e métodos do **objeto** estão disponíveis no script



- Scripts Gradle são **scripts de configuração**
  - Execução do script → configuração de um **objeto**
    - Build script → **Project**
    - Init script → **Gradle**
    - Settings script → **Settings**
  - Propriedades e métodos do **objeto** estão disponíveis no script
  - Propriedades e métodos da interface **Script** também disponíveis

- Composto **instruções** e **blocos** de script

- Composto **instruções** e **blocos** de script
- **Instruções**
  - Invocação de métodos
  - Atribuição de propriedades
  - Definição de variáveis locais
  - Definição de métodos e classes
  - **Qualquer elemento de script Groovy**

- Composto **instruções** e **blocos** de script
- **Instruções**
  - Invocação de métodos
  - Atribuição de propriedades
  - Definição de variáveis locais
  - Definição de métodos e classes
  - **Qualquer elemento de script Groovy**
- **Blocos** → invocação de um método com uma **closure de configuração** como parâmetro

## Blocos de Script de base

| Bloco            | O que configura                               |
|------------------|---|
| repositories { } | Os repositórios de dependências deste projeto |

## Blocos de Script de base

| Bloco            | O que configura                               |
|------------------|---|
| repositories { } | Os repositórios de dependências deste projeto |
| dependencies { } | As dependências deste projeto                 |

## Blocos de Script de base

| Bloco              | O que configura                                |
|--------------------|--|
| repositories { }   | Os repositórios de dependências deste projeto  |
| dependencies { }   | As dependências deste projeto                  |
| configurations { } | As configurações de dependências deste projeto |

## Blocos de Script de base

| Bloco              | O que configura                                |
|--------------------|--|
| repositories { }   | Os repositórios de dependências deste projeto  |
| dependencies { }   | As dependências deste projeto                  |
| configurations { } | As configurações de dependências deste projeto |
| sourceSets { }     | Os grupos de código e recursos deste projeto   |



## Blocos de Script de base

| Bloco              | O que configura                                |
|--------------------|--|
| repositories { }   | Os repositórios de dependências deste projeto  |
| dependencies { }   | As dependências deste projeto                  |
| configurations { } | As configurações de dependências deste projeto |
| sourceSets { }     | Os grupos de código e recursos deste projeto   |
| artifacts { }      | Os artefatos publicados deste projeto          |

## Blocos de Script de base

| Bloco              | O que configura                                |
|--------------------|--|
| repositories { }   | Os repositórios de dependências deste projeto  |
| dependencies { }   | As dependências deste projeto                  |
| configurations { } | As configurações de dependências deste projeto |
| sourceSets { }     | Os grupos de código e recursos deste projeto   |
| artifacts { }      | Os artefatos publicados deste projeto          |
| buildscript { }    | O classpath do build script deste projeto      |

## Blocos de Script de base

| Bloco              | O que configura                                |
|--------------------|--|
| repositories { }   | Os repositórios de dependências deste projeto  |
| dependencies { }   | As dependências deste projeto                  |
| configurations { } | As configurações de dependências deste projeto |
| sourceSets { }     | Os grupos de código e recursos deste projeto   |
| artifacts { }      | Os artefatos publicados deste projeto          |
| buildscript { }    | O classpath do build script deste projeto      |
| allprojects { }    | Este projeto e cada um dos sub projetos        |

## Blocos de Script de base

| Bloco              | O que configura                                |
|--------------------|--|
| repositories { }   | Os repositórios de dependências deste projeto  |
| dependencies { }   | As dependências deste projeto                  |
| configurations { } | As configurações de dependências deste projeto |
| sourceSets { }     | Os grupos de código e recursos deste projeto   |
| artifacts { }      | Os artefatos publicados deste projeto          |
| buildscript { }    | O classpath do build script deste projeto      |
| allprojects { }    | Este projeto e cada um dos sub projetos        |
| subprojects { }    | Apenas os sub projetos deste projeto           |

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle
- Task
  - Unidade atômica de trabalho da build

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle
- Task
  - Unidade atômica de trabalho da build
  - Entidade central à lógica do Gradle



# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle
- Task
  - Unidade atômica de trabalho da build
  - Entidade central à lógica do Gradle
  - Exemplos: compilar classes, gerar javadoc, etc.

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle
- Task
  - Unidade atômica de trabalho da build
  - Entidade central à lógica do Gradle
  - Exemplos: compilar classes, gerar javadoc, etc.
- Script

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle
- Task
  - Unidade atômica de trabalho da build
  - Entidade central à lógica do Gradle
  - Exemplos: compilar classes, gerar javadoc, etc.
- Script
  - Interface com métodos específicos do Gradle

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle
- Task
  - Unidade atômica de trabalho da build
  - Entidade central à lógica do Gradle
  - Exemplos: compilar classes, gerar javadoc, etc.
- Script
  - Interface com métodos específicos do Gradle
  - Implementado por todos scripts

# Principais Tipos de Objetos

- Project
  - Objeto alvo da configuração da build
  - Através dele, pode-se acessar todas funcionalidades do Gradle
- Task
  - Unidade atômica de trabalho da build
  - Entidade central à lógica do Gradle
  - Exemplos: compilar classes, gerar javadoc, etc.
- Script
  - Interface com métodos específicos do Gradle
  - Implementado por todos scripts
  - Automaticamente implementado pelo build.gradle

## Exemplo de Task e sua invocação

build.gradle:

```
task hello << {  
    println "Hello world!"  
}
```

Invocação no terminal:

```
> gradle hello
```

# Tasks padrões e dependências entre tasks

build.gradle:

```
defaultTasks 'clean', 'compile'

task clean << {
    println "Executando clean"
}

task compile << {
    println "Executando compile"
}

task other(dependsOn: 'compile') << {
    println "Executando other"
}
```

Invocação no terminal:

```
> gradle
Executando clean
Executando compile
> gradle other
Executando compile
Executando other
```

## Voltamos ao Ant?

- Gradle suporta plugins, como o Maven



## Voltamos ao Ant?

- Gradle suporta plugins, como o Maven
- Esses plugins registram Tasks e SourceSets

## Voltamos ao Ant?

- Gradle suporta plugins, como o Maven
- Esses plugins registram Tasks e SourceSets
- Convenção sobre Configuração → mesma estrutura de diretórios do Maven

# Exemplos de plugins

```
buildscript {  
    dependencies {  
        classpath 'com.android.tools.build:gradle:2.3.2'  
    }  
}  
  
apply plugin: 'java'  
apply plugin: 'groovy'  
apply plugin: 'com.android.application'
```

- Compatível com Maven e Ivy
  - Suporta repositórios Maven

- Compatível com Maven e Ivy
  - Suporta repositórios Maven
- Gerenciamento de dependências transitivas

## Gerenciamento de dependências

- Compatível com Maven e Ivy
  - Suporta repositórios Maven
- Gerenciamento de dependências transitivas
- Dependências/projetos são identificados por  
`"groupId:artifactId:version"`

## Definindo groupId, artifactId e version

- As propriedades **group** e **version** podem ser definidas em **gradle.build** ou **gradle.properties**

## Definindo groupId, artifactId e version

- As propriedades **group** e **version** podem ser definidas em **gradle.build** ou **gradle.properties**
  - build.gradle:
    - `group = "br.com.zeroglosa"`  
`version = "1.0.0"`



## Definindo groupId, artifactId e version

- As propriedades **group** e **version** podem ser definidas em **gradle.build** ou **gradle.properties**
  - build.gradle:
    - `group = "br.com.zeroglosa"`  
`version = "1.0.0"`
  - gradle.properties:
    - `group=br.com.zeroglosa`  
`version=1.0.0`

## Definindo groupId, artifactId e version

- As propriedades **group** e **version** podem ser definidas em **gradle.build** ou **gradle.properties**
  - build.gradle:
    - `group = "br.com.zeroglosa"`  
`version = "1.0.0"`
  - gradle.properties:
    - `group=br.com.zeroglosa`  
`version=1.0.0`
- A propriedade **name** pode ser definida em settings.gradle:
  - `rootProject.name = "artifactId"`

## Definindo groupId, artifactId e version

- As propriedades **group** e **version** podem ser definidas em **gradle.build** ou **gradle.properties**
  - build.gradle:
    - `group = "br.com.zeroglosa"`  
`version = "1.0.0"`
  - gradle.properties:
    - `group=br.com.zeroglosa`  
`version=1.0.0`
- A propriedade **name** pode ser definida em settings.gradle:
  - `rootProject.name = "artifactId"`
  - Se não definida, considera-se o nome da pasta do projeto

# Exemplos de dependências

```
apply plugin: 'java'

repositories {
    mavenCentral()
    mavenLocal()
    jcenter()
}

dependencies {
    compile 'com.google.guava:guava:20.0'

    testCompile 'junit:junit:4.12'
}
```

## Gerenciamento de dependências transitivas

- Dependências podem possuir dependências transitivas

## Gerenciamento de dependências transitivas

- Dependências podem possuir dependências transitivas
  - Que podem possuir outras dependências transitivas

## Gerenciamento de dependências transitivas

- Dependências podem possuir dependências transitivas
  - Que podem possuir outras dependências transitivas
- Diferentes versões podem ser especificadas
  - **A:1.0  $\rightarrow$  B:1.2**
  - **C:1.5  $\rightarrow$  B:1.3**
  - **D:2.1  $\rightarrow$  C:1.6  $\rightarrow$  B:1.5**

## Gerenciamento de dependências transitivas

- Dependências podem possuir dependências transitivas
  - Que podem possuir outras dependências transitivas
- Diferentes versões podem ser especificadas
  - **A:1.0** → **B:1.2**
  - **C:1.5** → **B:1.3**
  - **D:2.1** → **C:1.6** → **B:1.5**
  - Qual versão de **A**, **B**, **C** e **D**?



# Gerenciamento de dependências transitivas

- Regras para definição de versões:
  - Primeiro mais próximo
    - Dependência de nível mais alto tem prioridade
    - Dependência direta tem prioridade sobre transitiva

# Gerenciamento de dependências transitivas

- Regras para definição de versões:
  - Primeiro mais próximo
    - Dependência de nível mais alto tem prioridade
    - Dependência direta tem prioridade sobre transitiva
  - Primeiro encontrado (desempate)
    - Primeira encontrada é usada

# Gerenciamento de dependências transitivas

- Regras para definição de versões:
  - Primeiro mais próximo
    - Dependência de nível mais alto tem prioridade
    - Dependência direta tem prioridade sobre transitiva
  - Primeiro encontrado (desempate)
    - Primeira encontrada é usada
- Problema:
  - **A:1.0 → B:1.2**
  - **C:1.5 → B:1.3**
  - **D:2.1 → C:1.6 → B:1.5**

# Gerenciamento de dependências transitivas

- Regras para definição de versões:
  - Primeiro mais próximo
    - Dependência de nível mais alto tem prioridade
    - Dependência direta tem prioridade sobre transitiva
  - Primeiro encontrado (desempate)
    - Primeira encontrada é usada
- Problema:
  - **A:1.0** → **B:1.2**
  - **C:1.5** → **B:1.3**
  - **D:2.1** → **C:1.6** → **B:1.5**
- Solução:
  - A:1.0
  - B:1.2
  - C:1.5
  - D:2.1

# Exclusão de dependências

```
apply plugin: 'java'

repositories {
    mavenCentral()
    mavenLocal()
    jcenter()
}

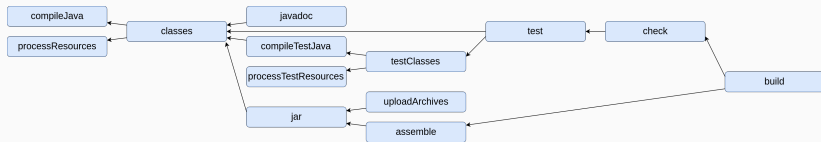
dependencies {
    compile('org.hibernate:hibernate-core:5.2.10.Final'){
        exclude module: 'dom4j'
        exclude group: 'org.javassist', module: 'javassist'
    }

    testCompile 'junit:junit:4.12'
}
```

# Java Plugin

---

# Tasks

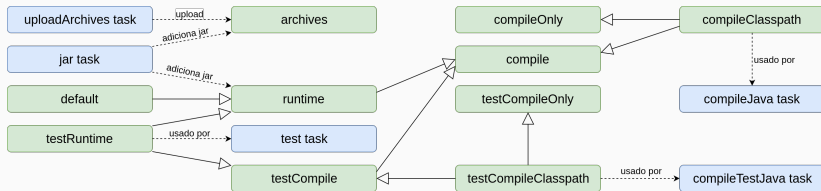


## Estrutura do projeto

| Diretório                        | Função                             |
|----------------------------------|------------------------------------|
| src/main/java                    | Código fonte de produção           |
| src/main/resources               | Recursos de produção               |
| src/test/java                    | Código de testes                   |
| src/test/resources               | Recursos de testes                 |
| src/ <b>sourceSet</b> /java      | Código de dado <b>dataSource</b>   |
| src/ <b>sourceSet</b> /resources | Recursos de dado <b>dataSource</b> |



# Configurações de dependências/Escopos



## Conclusões

---

- Existem  **muitos plugins**, inclusive para diversas linguagens

- Existem  **muitos plugins**, inclusive para diversas linguagens
- Grande **flexibilidade**, inclusive para desenvolver plugins

- Existem  **muitos plugins**, inclusive para diversas linguagens
- Grande  **flexibilidade**, inclusive para desenvolver plugins
- Permite fácil  **automatização** da build

- Existem  **muitos plugins**, inclusive para diversas linguagens
- Grande  **flexibilidade**, inclusive para desenvolver plugins
- Permite fácil  **automatização** da build
- **Convenção** sobre configuração

- Existem  **muitos plugins**, inclusive para diversas linguagens
- Grande  **flexibilidade**, inclusive para desenvolver plugins
- Permite fácil  **automatização** da build
- **Convenção** sobre configuração
- **DSL** compacta e poderosa