





PARTE II – UNIDADE 01 FUNDAMENTOS DE CONSTRUÇÃO DE SISTEMAS COM GRADLE

PROFESSOR: FABRÍCIO G. M. DE CARVALHO

São José dos Campos - 2019

1



OBJETIVOS

- Enfatizar as dificuldades associadas ao processo de construção (build) de software;
- Apresentar as características básicas do Gradle associadas ao processo de automação da construção de software.
- Mostrar como executar casos de testes utilizando-se o JUNIT em conjunto com o Gradle.

2

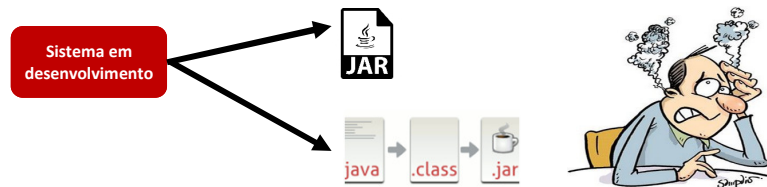
TÓPICOS ABORDADOS

1. MOTIVAÇÃO
2. INSTALAÇÃO E CONFIGURAÇÃO DA FERRAMENTA
3. FUNCIONAMENTO GERAL
4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

3

1. MOTIVAÇÃO

Projetos de software tipicamente fazem uso de componentes já desenvolvidos por terceiros ou reaproveitados de outros projetos. A reutilização desses "pedaços" de software apesar de agilizar o ciclo de desenvolvimento acarreta problemas associados a dependências.



4

1. MOTIVAÇÃO

Vários processos do ciclo de desenvolvimento possuem natureza repetitiva e que são candidatas à automação, incluindo testes, compilação, ligação, geração de arquivos de distribuição, etc.



5

1. MOTIVAÇÃO

A utilização de ferramentas de construção de software serve para automatizar várias atividades do ciclo de desenvolvimento de software, com ênfase na construção do sistema executável, incluindo tarefas tais como a obtenção de dependências, execução de comandos de compilação, execução de comandos de ligação, execução do sistema e até mesmo a implantação (deployment).

6

Fatec
São José dos Campos
Prof. Jessen Vidal

SÃO PAULO
GOVERNO DO ESTADO

1. MOTIVAÇÃO

Exemplos de ferramentas de construção:



7

Fatec
São José dos Campos
Prof. Jessen Vidal

SÃO PAULO
GOVERNO DO ESTADO

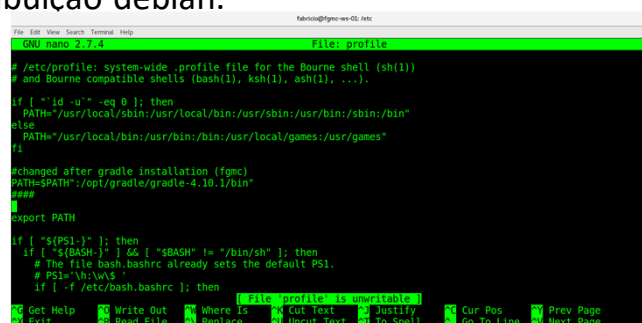
2. GRADLE: INSTALAÇÃO E CONFIGURAÇÃO

1. EFETUAR O DOWNLOAD DA ÚLTIMA VERSÃO (BINÁRIA) DO GRADLE
EM <https://gradle.org/releases/>
- 2a. No linux, descompactar em /opt/gradle
- 2b. No Windows, descompactar em C:/gradle
- 3a. Adicionar a pasta binária da aplicação ao path dos usuários (arquivo etc/profile para o caso da distribuição debian.
- 3b. Adicionar à variável Path do Windows.

8

2. GRADLE: INSTALAÇÃO E CONFIGURAÇÃO

3a. Adicionar a pasta binária da aplicação ao path dos usuários (arquivo etc/profile para o caso da distribuição debian).



```
File: profile
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

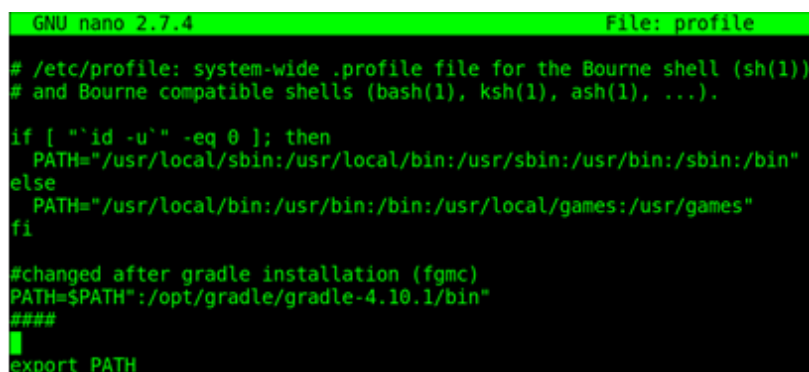
if [ "`id -u`" -eq 0 ]; then
    PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
    PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi

#changed after gradle installation (fgmc)
PATH=$PATH:/opt/gradle/gradle-4.10.1/bin"
###
export PATH

if [ "${PS1:-}" ]; then
    if [ "${BASH:-}" ] && [ "${BASH}" != "/bin/sh" ]; then
        # The file bash.bashrc already sets the default PS1.
        # PS1="\h:\V\$ "
        if [ -f /etc/bash.bashrc ]; then
            # file 'profile' is unwritable
        fi
    fi
fi
```

9

2. GRADLE: INSTALAÇÃO E CONFIGURAÇÃO




```
GNU nano 2.7.4 File: profile
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).


if [ "`id -u`" -eq 0 ]; then
    PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
    PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi

#changed after gradle installation (fgmc)
PATH=$PATH:/opt/gradle/gradle-4.10.1/bin"
###
export PATH
```

10



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO

2. GRADLE: INSTALAÇÃO E CONFIGURAÇÃO

4. Testar digitando gradle -v no prompt de comando /shell


```
E:\PRJ-OPEN\3.pub\3.1.tut\3.1.6\docs>gradle -v

-----
Gradle 4.10.2
-----


Build time:   2018-09-19 18:10:15 UTC
Revision:    b4d8d5d170bb4ba516e88d7fe5647e2323d791dd

Kotlin DSL:   1.0-rc-6
Kotlin:       1.2.61
Groovy:       2.4.15
Ant:          Apache Ant(TM) version 1.9.11 compiled on March 23 2016
JVM:          1.8.0_151 (Oracle Corporation 25.151-b12)
OS:           Windows 7 6.1 amd64
```

11



Fatec
São José dos Campos
Prof. Jessen Vidal




SÃO PAULO
GOVERNO DO ESTADO

3. GRADLE: FUNCIONAMENTO GERAL


3.1. CARACTERÍSTICAS GERAIS

- Essa ferramenta opera através da execução de vários objetos que são invocados a partir de um script.
- Esses objetos efetuam várias tarefas, incluindo a inicialização de um novo projeto, busca de dependências **diretas** ou **transitivas** em um repositório, compilação do código-fonte, geração do executável, empacotamento do executável, etc.

12



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO

3.GRADLE: FUNCIONAMENTO GERAL


3.1. CARACTERÍSTICAS GERAIS

Exemplos de objetos de alto nível em um script do Gradle (são ditos objetos delegados):


- Project – objeto delegado de um build script.
- Gradle – objeto delegado de um init script.
- Settings – objeto delegado de um settings script.

OBS: Propriedades e métodos dos objetos delegados são acessíveis através dos scripts.

13



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO

3.GRADLE: FUNCIONAMENTO GERAL

3.2. SINTAXE DOS SCRIPTS

O componente essencial para utilização do Gradle como *build tool* é o chamado build script (arquivo ***build.gradle***).

Um build script possui zero ou vários blocos ou declarações.

Blocos: delimitados por { }. Correspondem a uma chamada de um método de um dos objetos delegados .

Declarações: definições de variáveis, atribuição de propriedades, etc.

14

3.GRADLE: FUNCIONAMENTO GERAL

3.3. EXEMPLO DE BUILD DE PROJETO JAVA

Criar um arquivo chamado build.gradle com o seguinte conteúdo:

```
plugins{  
    id 'java'  
    id 'application'  
}
```

15



3.GRADLE: FUNCIONAMENTO GERAL

3.3. EXEMPLO DE BUILD DE PROJETO JAVA

Criar um arquivo chamado build.gradle com o seguinte conteúdo (cont):

```
mainClassName = 'Aplicacao'  
sourceSets{  
    main{  
        java{  
            srcDirs 'src/main'  
        }  
    }  
}
```

16



3.GRADLE: FUNCIONAMENTO GERAL



3.3. EXEMPLO DE BUILD DE PROJETO JAVA

Criar um arquivo chamado *settings.gradle* e acrescentar a seguinte linha:

```
rootProject.name = 'primeiro_projeto'
```

Feito isso, criar, uma pasta *src/main* e, dentro dela, criar uma classe chamada *Aplicacao*, que contenha o método estático *main* que imprima uma mensagem de texto.

17



3.GRADLE: FUNCIONAMENTO GERAL


3.3. EXEMPLO DE BUILD DE PROJETO JAVA

Depois, no prompt de comando digitar os seguintes comandos:


```
gradle wrapper  
./gradlew build (se Linux)  
gradlew.bat (se Windows)
```

Verificar que, caso seja bem sucedido, o build criará uma pasta *build* que conterá, entre outras coisas, um arquivo de distribuição *.tar* e *.zip*, contendo o *.jar* e scripts de inicialização da aplicação.

18



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO


3.GRADLE: FUNCIONAMENTO GERAL

3.4. EXEMPLO DE EXECUÇÃO DE PROJETO JAVA


Os plugins 'java' e 'application' permitem que as aplicações java sejam executadas a partir do Gradle, para tanto, basta invocar uma tarefa chamada 'run':

```
./gradlew run
```

19



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO

3.GRADLE: FUNCIONAMENTO GERAL

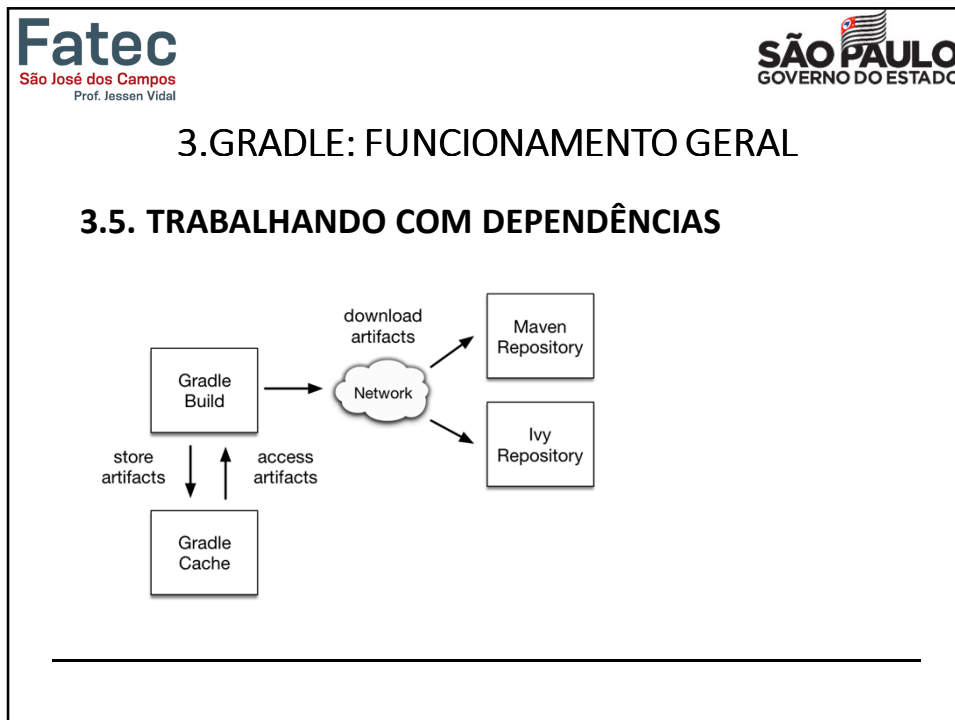
3.5. TRABALHANDO COM DEPENDÊNCIAS

Uma das vantagens de se utilizar o Gradle é a resolução dos problemas de dependência transitiva (i.e. Dependência da dependência direta).

Os módulos ou componentes que compõem a dependência são armazenados em um cache local do gradle após terem sido obtidas do repositório na primeira vez em que forem utilizadas.

Cada plugin pode adicionar configurações e dependências específicas.

20



21

Fatec
São José dos Campos
Prof. Jessen Vidal

SÃO PAULO
GOVERNO DO ESTADO

3. GRADLE: FUNCIONAMENTO GERAL



3.5. TRABALHANDO COM DEPENDÊNCIAS

Para se trabalhar com dependências, primeiro especifica-se um repositório, através do bloco ***repositories***, e, a seguir, as dependências requeridas, através do bloco ***dependencies***:

```

repositories{
    mavenCentral()
    jcenter()
    google()
}
  
```

22





3.GRADLE: FUNCIONAMENTO GERAL

3.5. TRABALHANDO COM DEPENDÊNCIAS

Para se customizar um repositório supondo-se um repositório do tipo Maven ou Ivy, utiliza-se a seguinte sintaxe (exemplo):

```
repositories{  
    maven{  
        url  
        "http://seudominio/seurepositorio"  
    }  
}
```

23



3.GRADLE: FUNCIONAMENTO GERAL


3.5. TRABALHANDO COM DEPENDÊNCIAS

No bloco ***dependencies***, efetivamente se informa ao Gradle a configuração da dependência, qual tipo de dependência deve ser utilizada e qual o módulo ou componente é requerido pela dependência.


A configuração corresponde ao escopo de aplicação da dependência.

O tipo de dependência refere-se a um módulo ou arquivo específico.

24



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO

3. GRADLE: FUNCIONAMENTO GERAL


3.5. TRABALHANDO COM DEPENDÊNCIAS

De um modo geral a sintaxe é a seguinte:


```
dependencies{
    <configuracao><tipo_de_dependencia>:<modulo_
    ou_componente>:<versão>
}
```

**Obs: Algumas vezes ao invés da separação com :
pode haver a sintaxe do tipo chamada a método.**

25



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO

3. GRADLE: FUNCIONAMENTO GERAL

3.5. TRABALHANDO COM DEPENDÊNCIAS

Exemplos:

```
dependencies{
    compile '<grupo>:<nome>:<versão>' //módulo usado
        na compilação
    runtime '<grupo>:<nome>:<versão>' //módulo usado
        na execução
    //notação de mapa - módulo
    compile group: 'grupo', name: 'nome', version:
        'versão'
}
```

26

3. GRADLE: FUNCIONAMENTO GERAL

3.5. TRABALHANDO COM DEPENDÊNCIAS

Exemplos:

```
dependencies{
    runtime files('libs/a.jar',
        'libs/b.jar')
    compile fileTree(dir: 'libs', include:
        '*.jar')
}
```


27

3. GRADLE: FUNCIONAMENTO GERAL


3.6. CONFIGURAÇÕES DE DEPENDÊNCIA - JAVA PLUGIN

```
dependencies{
    implementation //substitui compile
    compileOnly
    runtimeOnly
    testImplementation // testCompile
    e implementation
    testCompileOnly //apenas para compilar testes
    testRuntimeOnly //apenas para executar
    testes
}
```

28



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO


3. GRADLE: FUNCIONAMENTO GERAL

3.7. OBSERVAÇÕES GERAIS


1. Para se criar um projeto, seguindo-se as opções de diretório default para o caso do java, pode-se digitar:


```
gradle init --type java-application
```
2. Nesse caso, deve-se configurar o seu IDE preferido para ficar compatível com tal estrutura de diretórios.

29



Fatec
São José dos Campos
Prof. Jessen Vidal



SÃO PAULO
GOVERNO DO ESTADO

3. GRADLE: FUNCIONAMENTO GERAL



3.7. OBSERVAÇÕES GERAIS

3. Para se copiar uma versão local das dependências a partir do cache do Gradle, pode-se utilizar uma regra tal como:


```
task copyCompileDependencies(type: Copy){
    from configurations.compile,
    configurations.runtime
    into 'lib/main'
}
```

OBS: Examinar dependências com gradle -q dependencies

30





4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

4.1. FONTES DE TESTE E DA APLICAÇÃO (build script)

```
sourceSets{  
    main{  
        java{  
            srcDirs 'src/main'  
        }  
    }  
    test{  
        java{  
            srcDirs 'src/test'  
        }  
    }  
}
```

31





4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

4.1. FONTES DE TESTE E DA APLICAÇÃO (Somador.java)

```
public class Somador{  
    public int somar(int x, int y){  
        return (x+y);  
    }  
}
```

32





4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

4.1. FONTES DE TESTE E DA APLICAÇÃO (SomadorTest.java)

```
//Importação das classes da API:  
import static  
    org.junit.jupiter.api.Assertions.assertEquals;  
import org.junit.jupiter.api.Test;
```

33





4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

4.1. FONTES DE TESTE E DA APLICAÇÃO (SomadorTest.java)

```
// Código da classe de teste  
public class SomadorTest{  
    private Somador somador;  
    public SomadorTest(){  
        somador = new Somador();  
    }  
    @Test  
    public void somarTest(){  
        assertEquals(2,somador.somar(1,1));  
    }  
}
```

34





4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

4.2. REPOSITÓRIOS E DEPENDÊNCIAS (build script)

```
repositories{  
    mavenCentral()  
}  
  
dependencies{  
    testCompileOnly 'org.junit.jupiter:junit-jupiter-api:5.3.1'  
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.3.1'  
}
```

35





4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

4.2. REPOSITÓRIOS E DEPENDÊNCIAS (build script)

```
test{  
    useJUnitPlatform()  
}
```

36



4. EXECUÇÃO DE CASOS DE TESTE COM JUNIT

4.2. EXECUÇÃO DE UMA TAREFA DE TESTE



No prompt de comando digitar:

```
./gradlew test
```

Observar que os relatórios de teste estão dentro da pasta

```
build/reports/tests/test/index.html
```

37



EXERCÍCIO

Desenvolva uma classe chamada Calculadora, em um pacote chamado modelo, que possui os métodos somar, subtrair, multiplicar e dividir. Em todos os casos, os parâmetros de entrada são do tipo float e a saída é do tipo float. Em um pacote chamado aplicacao, crie uma aplicação que faça uso da calculadora, exercitando todos os métodos da classe e, adicionalmente:

- A) Faça o build e a execução da aplicação com o Gradle;
- B) Execute casos de teste para todos os métodos e mostre seu relatório gerado a partir do JUnit. Insira, propositadamente, um bug em cada método de modo que falhe para um caso de teste e seja aprovado em outro (ou seja, crie pelo menos dois casos de teste para cada método).

38

Fatec
São José dos Campos
Prof. Jessen Vidal

**SÃO PAULO**
GOVERNO DO ESTADO

FIM DA UNIDADE
