Duvidas

1. [Duvida sobre repositórios em Java](#_toc363)
2. [Duvida sobre o testes unitários](#_toc433)

Fases do ciclo de vida do gradle

Descrição

Inicialização

1. Verificar se é apenas um projecto ou mais que um
2. Encontrar os ficheiros de “*build*” do gradle

Configuração

1. Executa todas as acções entre chavetas ({) sem (<<)
   1. Estes blocos não são acções de “*tasks”*
   2. *São blocos de configuração*
2. Criação do “*directed acyclic graph*” (DAG)
3. Identificação das dependências das “*tasks*” para serem executadas na próxima fase

Execução

1. Execução de acordo com as dependências identificadas na fase anterior
2. Executa métodos das acções
3. Executa as acções que são identificadas pelo sinal (<<)

Exemplo

task initializeConnections

initializeConnections << { println 'connecting' }

initializeConnections {

description="Connections"

group="GI"

println 'configuring connections'}

Resultado

1. Parametro -q não mostra informação da execução do gradle
   1. Como
      1. <nome da Task>:
      2. BUILD SUCCESSFUL

%gradle -q initializeConnections

configuring connections

connecting

Propriedades

Descrição

1. As “*tasks*” tem propriedades próprias
   1. Referências ás propriedades das “*tasks”*
      1. <https://docs.gradle.org/current/dsl/org.gradle.api.Task.html>
2. No exemplo anterior usamos as propriedades “*group*” e “*description*”
   1. Como estão dentro de chavetas ficam associadas á task
3. Ver aula teórica
   1. É possível usar propriedades não nativas que ficam logo declaradas
   2. É possível usar usar propriedades globais
      1. São associadas ao objecto: “*project*”

Métodos

Descrição

doFirst, doLast, leftShift

1. Executa no inicio e no fim cumulativamente

Exemplo

1. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A2\_Do\_First\_Last](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/odsoft-2016-nmb-g101/odsoft/PL2/A2_Do_First_Last)

task myTask1

myTask1.doLast{println "Last with doLast"}

myTask1.doLast({println "Last with doLast again"})

myTask1.leftShift {println "Last with leftShift"}

myTask1 << {println "Last with <<"}

myTask1.doFirst {println "First with doFirst"}

myTask1.doFirst({println "doFirst again"})

Resultado

%gradle myTask1

:myTask1

doFirst again

First with doFirst

Last with doLast

Last with doLast again

Last with leftShift

Last with <<

BUILD SUCCESSFUL

Relações entre tarefas

Descrição

DependsOn

1. Se a task1 depende da task2 o gradle vai garantir que a task2 é executada antes da task1
2. Pode não ser imediatamente antes

finalizedBy

1. Especifica que a execuçã de uma tarefa tem que ser seguida da execução de outra

mustRunAfter

1. Especifica que uma tarefa tem que ser executada a seguir a outra e que as duas estão marcadas para serem executadas
2. Não implica dependências entre tarefas visto que uma pode ser executada sem a outra

Exemplo 1

1. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A1\_Dependencias/A1\_DependsOn](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/odsoft-2016-nmb-g101/odsoft/PL2/A1_Dependencias/A1_DependsOn)

task myTask1 << {println "This is myTask1"}

task myTask2 << {println "This is myTask2"}

myTask1.dependsOn myTask2

myTask2.dependsOn myTask1

Resultado

1. Dá erro devido a dependências cíclicas

Starting a Gradle Daemon (subsequent builds will be faster)

FAILURE: Build failed with an exception.

\* What went wrong:

Circular dependency between the following tasks:

:myTask1

\--- :myTask2

\--- :myTask1 (\*)

(\*) - details omitted (listed previously)

\* Try:

Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.

BUILD FAILED

Total time: 2.507 secs

Exemplo 2

1. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A1\_Dependencias/A2\_MustRunAfter](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/odsoft-2016-nmb-g101/odsoft/PL2/A1_Dependencias/A2_MustRunAfter)

task myTask1 << {println "This is myTask1"}

task myTask2 << {println "This is myTask2"}

task myTask3 << {println "This is myTask3"}

myTask2.dependsOn myTask1

myTask3.mustRunAfter myTask1

Resultado

gradle myTask2 myTask3

1,2,3

gradle myTask3 myTask2

3,1,2

Buid-in Tasks

Descrição

DefaultTask

1. Tipo predefinido de “*task”* se nada for especificado

Trabalhar com ficheiros

1. <https://docs.gradle.org/current/userguide/working_with_files.html>

Exemplo

1. Copiar conjunto de ficheiros de uma pasta para outra
2. A chaveta tem que ficar na mesma linha que a instrução
3. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A3\_Copia\_Ficheiro](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/odsoft-2016-nmb-g101/odsoft/PL2/A3_Copia_Ficheiro)

task copyTXT(type: Copy){

from 'A1\_Origem'

include '\*.txt'

into 'A2\_Destino'

}

Resultado

gradle CopyTXT

:copyTXT

BUILD SUCCESSFUL

Total time: 1.24 secs

“*Build*” com parâmetros

Descrição

1. Na execução do “*build*” podemos passar parâmetros ao gradle
2. É possível adicionar propriedades externas ao objecto “*project*”
   1. Através de
      1. Argumentos na linha de comando
      2. Criação de um ficheiro de propriedades
      3. Definição de propriedades de ambiente
3. As propriedades definidas na linha de comando sobrepõe-se a todas os outros métodos de definição de propriedades

Ficheiros de propriedades

1. Na pasta do "*script*" de “*build*“ podemos criar um ficheiro onde iremos definir as propriedades
   1. É possível utilizar outras localizações para o ficheiro de propriedades
2. Neste ficheiro tem que existir uma linha para cada propriedade onde é definido o valor da mesma

Exemplo 1

1. Passagem de parametros
2. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A4\_Parametros\_na\_Execucao/](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/odsoft-2016-nmb-g101/odsoft/PL2/A4_Parametros_na_Execucao/)

task printCourseName << {

println courseName

}

Resultado

gradle -PcourseName="ODSOFT" printCourseName

:printCourseName

ODSOFT

Exemplo 2

1. Uso de ficheiro de propriedades
2. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A5\_Ficheiro\_Propriedades](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/odsoft-2016-nmb-g101/odsoft/PL2/A5_Ficheiro_Propriedades)
3. Ficheiro: build.gradle

task printCourseName << {

Properties props = new Properties()

props.load(new FileInputStream("course.properties"))

println props.get('courseName')

}

1. Ficheiro: course.properties

courseName = "EDOM

Resultado

gradle printCourseNam

:printCourseName

"EDOM"

BUILD SUCCESSFUL

Total time: 0.92 secs

Exemplo 3

1. Acesso a variáveis do sistema
2. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A6\_Acesso\_Variaveis\_Sistema](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/odsoft-2016-nmb-g101/odsoft/PL2/A6_Acesso_Variaveis_Sistema)

Resultado

gradle printEnvironmentVariable

:printEnvironmentVariable

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/home/p0/bin:/home/p0/lib:/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/A2\_PL2/A1\_Gradle/bin

BUILD SUCCESSFUL

Total time: 0.669 secs

*“Plugin” do Java*

Descrição

1. “*Plugins*” são componentes reutilizáveis que permitem expandir as funcionalidades do gradle
2. Por exemplo usando o “*plugin*” correcto torna expedito o processo de “*building*” de projectos Java

Ficheiro de *“build”*

*apply plugin: 'java'*

Identificação de *“tasks”*

1. *Esta evocação do gradle com o parâmetro “tasks” faz com que seja apresentada uma listagem de todas as “tasks” que o “plugin” contém por omissão*

*gradle tasks*

*:tasks*

*------------------------------------------------------------*

*All tasks runnable from root project*

*------------------------------------------------------------*

*Build tasks*

*-----------*

*assemble - Assembles the outputs of this project.*

*build - Assembles and tests this project.*

*buildDependents - Assembles and tests this project and all projects that depend on it.*

*buildNeeded - Assembles and tests this project and all projects it depends on.*

*classes - Assembles main classes.*

*clean - Deletes the build directory.*

*jar - Assembles a jar archive containing the main classes.*

*testClasses - Assembles test classes.*

*Build Setup tasks*

*-----------------*

*init - Initializes a new Gradle build. [incubating]*

*wrapper - Generates Gradle wrapper files. [incubating]*

*Documentation tasks*

*-------------------*

*javadoc - Generates Javadoc API documentation for the main source code.*

*Help tasks*

*----------*

*buildEnvironment - Displays all buildscript dependencies declared in root project 'A7\_Java'.*

*components - Displays the components produced by root project 'A7\_Java'. [incubating]*

*dependencies - Displays all dependencies declared in root project 'A7\_Java'.*

*dependencyInsight - Displays the insight into a specific dependency in root project 'A7\_Java'.*

*help - Displays a help message.*

*model - Displays the configuration model of root project 'A7\_Java'. [incubating]*

*projects - Displays the sub-projects of root project 'A7\_Java'.*

*properties - Displays the properties of root project 'A7\_Java'.*

*tasks - Displays the tasks runnable from root project 'A7\_Java'.*

*Verification tasks*

*------------------*

*check - Runs all checks.*

*test - Runs the unit tests.*

*Rules*

*-----*

*Pattern: clean<TaskName>: Cleans the output files of a task.*

*Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.*

*Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to a configuration.*

*To see all tasks and more detail, run gradle tasks --all*

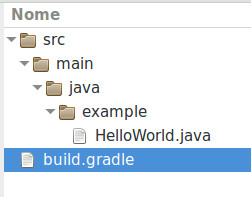
*To see more detail about a task, run gradle help --task <task>*

*BUILD SUCCESSFUL*

*Total time: 0.693 secs*

*Criação da estrutura de pastas para projectos*

1. *Neste exemplo apenas tenho uma classe chamada HelloWorld*



Código Java

package example;

public class HelloWorld

{

public static void main(String args[])

{

System.out.println("hello, world");

}

}

Repositórios

1. Dependências externas de um projecto podem ser descarregadas automaticamente a partir de repositórios
2. As especificações dos repositórios devem ser colocadas num bloco chamado “*repositories”*

Exemplo 1

1. Executa a compilação do projecto Java
2. [/home/p0/A3\_Corrente/A4\_Universidade/2017/A4\_1\_Semestre/A1\_ODSOFT/odsoft-2016-nmb-g101/odsoft/PL2/A7\_Java](file:///home/p0/A3_Corrente/A4_Universidade/2017/A4_1_Semestre/A1_ODSOFT/odsoft-2016-nmb-g101/odsoft/odsoft-2016-nmb-g101/odsoft/PL2/A7_Java)

gradle classes

Resultado

:compileJava

:processResources UP-TO-DATE

:classes

BUILD SUCCESSFUL

Total time: 0.685 secs

1. Executar o projecto compilado

java -cp ./build/classes/main/ example.HelloWorld

Resultado

hello, world

Exemplo 2

1. Descarregar repositórios para satisfazer dependências do projecto Java

apply plugin: "java"

repositories {

mavenCentral()

mavenLocal()

jcenter()

}

Resultado

1. Apagar ultima compilação

gradle clean

:clean UP-TO-DATE

BUILD SUCCESSFUL

Total time: 0.707 secs

1. Realizar nova compilação

gradle classes

:compileJava

:processResources UP-TO-DATE

:classes

BUILD SUCCESSFUL

Total time: 0.685 secs

1. Executar projecto Java compilado

java -cp ./build/classes/main/ example.HelloWorld

hello, world

Duvidas de repositórios

1. Não fez download de nada
2. Perguntar ao professor o que deveria acontecer

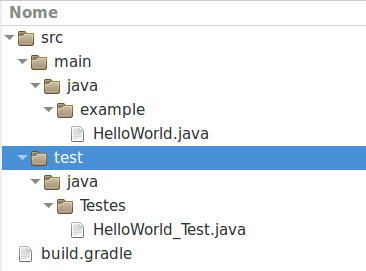
Testes de unidade

Descrição

1. O gradle suporta o uso da “*framework*” de testes JUnit
2. Os testes devem ficar na pasta src/test/java
3. O Gradle compila os código Java e corre os testes
4. É necessário configurar as dependências de teste
5. Para executar o teste é necessário corre a “*task*” chamada “*test*”
6. Os relatórios dos testes são colocados na pasta build/reports/tests.

Criação da estrutura de pastas para projectos

1. *Neste exemplo tenho uma classe chamada HelloWorld que irá ser testada pela classe HolloWorld\_Test.java*



Código Java

1. Estou a usar o código Java do ponto anterior ([consultar para mais detalhes](#_toc232))
2. HelloWorld\_Test.java

//Pachage

package Testes;

//Imports

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class HelloWorld\_Test

{

@Test

public void mainShouldPrintHelloWorld()

{

//Variaveis locais

example.HelloWorld clsTeste = new example.HelloWorld();

String[] teParametroL = {"",""};

//Teste

assertEquals("A execução do metodo main deve imprimir no ecrã: Helo Word", 0, clsTeste.main(teParametroL));

}

}

Exemplo

1. build.gradle

apply plugin: 'java'

repositories {

mavenCentral()

}

dependencies {

testCompile 'junit:junit:4.11'

}

Resultado

1. Executar o teste através do gradle

gradle test

1. Resultado da execução do teste do gradle

:compileJava

:processResources UP-TO-DATE

:classes

:compileTestJava

:processTestResources UP-TO-DATE

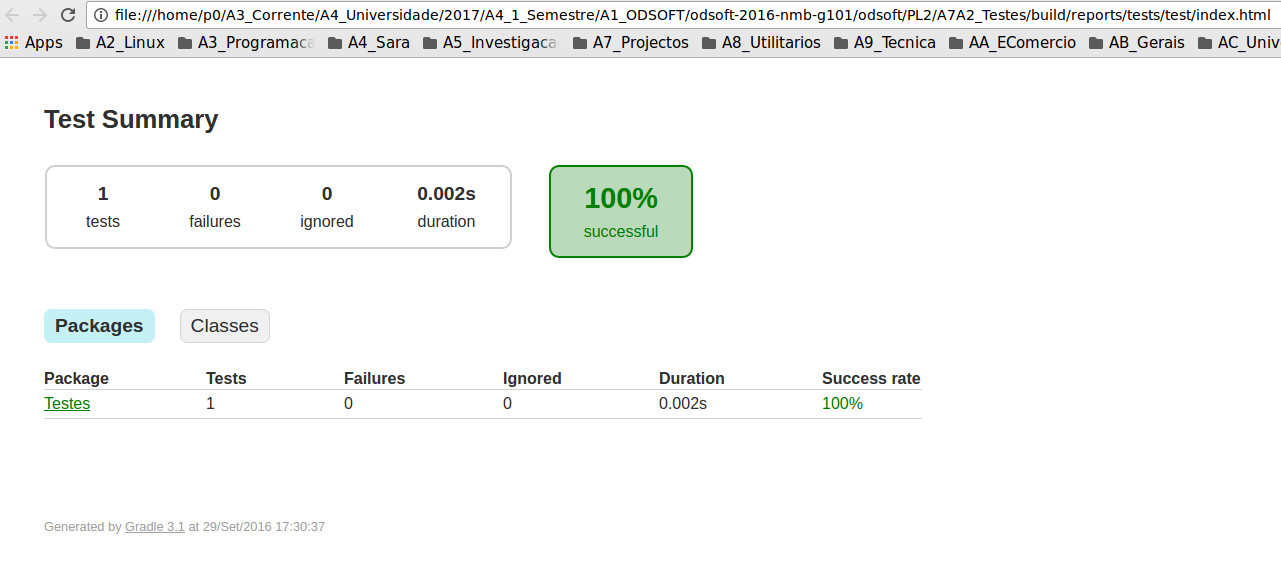
:testClasses

:test

BUILD SUCCESSFUL

Total time: 1.153 secs

1. Relatório de testes



Duvidas testes unitários

1. Na realização do teste não mostra o “*assert*” nem a execução do teste
   1. Como fazer para ver
2. Explicação sobre o funcionamento do Junit
   1. [http://www.vogella.com/tutorials/JUnit/article.html#testintroduction](http://www.vogella.com/tutorials/JUnit/article.html" \l "testintroduction)
   2. O Junit faz
      1. Integration tests
      2. Performance tests
      3. Behavior vs. state testing
   3. Como executava o teste na linha de comando
3. Como é que a classe exemplo.HelloWorl é conhecida na classe Testes.HelloWorld\_Test
4. Porque não tenho que criar um método “*main”* na classe de teste