cesae digital

Centro para o Desenvolvimento
de Competências Digitais

Engenharia de Software II

Version
V1.0

Introdução a Gradle

Summary | Noções Básicas • Demonstração Prática

Engenharia de Software II

- Introdução ao Gradle
- Configuração do Gradle
- Demonstração
- Exercício

Introdução

- Porque a automatização do processo de construção de software é tão importante?
- Em que consiste um processo de construção?
  - Executar testes;
  - Requerer recursos externos de outras fontes;
  - Gerar documentação;
  - Criar variantes de compilação;
  - Gerir dependências;
  - Publicar.
- Podemos realizar uma analogia com uma fábrica em que o processo produtivo é constituído por várias etapas:

Introdução

- O processo de construção de software (por exemplo em linguagem Java) envolve várias tarefas:
  - Estruturação/organização do projeto
    - Facilita a navegação e leitura do código; torna a arquitetura do software mais evidente e clarividente; facilita a automatização. Sem uma organização uniformizada é difícil a automatização de todo o processo de “build”;
  - Codificação
  - Revisão

Introdução

- Compilação (utilizando os ficheiros com código fonte - .java)
  - No caso de o projeto possuir dependências é necessário incluir essas dependências na classpath (javac -cp *.:jars/jar1:jars/jar2:jars/jar3 source/*.java*)
- Testar o código produzido;
  - Criação do *package* (JAR, WAR, etc);
  - Geração do javadoc;
  - Publicar.
- Atualmente, o processo de construção pode ser visto como um grafo de tarefas.

Introdução

- O Make, Ant e Maven são exemplos de ferramentas que antecederam o Gradle e que influenciaram o seu desenvolvimento;

Make

```
./project/
├── Makefile
└── src
    ├── a
    │   ├── b
    │   │   ├── c
    │   │   │   └── foo.h
    │   │   ├── bar.c
    │   │   └── bar.h
    │   ├── qux.c
    │   └── qux.h
    └── main.c
```

Introdução

- Ferramentas que suportam Gradle:



- O Gradle permite integrar e automatizar várias tarefas relacionadas processo de desenvolvimento software;
- O Gradle determina quais os componentes do projeto que estão atualizados, evitando a recompilação de todo o projeto.

Configuração do Gradle

- O Gradle possui dois conceitos base: projetos e tarefas;
- Um projeto representa o código produzido (por exemplo um ficheiro JAR), consistindo em uma ou mais tarefas;
- Uma tarefa é uma unidade atómica que é executada. Por exemplo: compilação do projeto ou execução de testes;
- Cada compilação do Gradle contém um ou mais projetos;
- O processo de compilação Gradle contempla três fases: inicialização, configuração e execução:
  - Na inicialização, o Gradle localiza os ficheiros de construção (verificando se é uma configuração de um projeto ou multiprojeto - https://docs.gradle.org/current/userguide/multi_project_builds.html);
  - Na configuração, o Gradle executa cada ficheiro de construção como um Groovy script - http://groovy-lang.org/structure.html ;
  - Na execução, o Gradle identifica as tarefas a executar considerando as precedências de execução.

Configuração do Gradle

- Exemplo de ficheiro de configuração Gradle:

```
apply plugin: 'java'
uploadArchives {
    repositories {
       flatDir {
           dirs 'repos'
       }
    }
}

task complete {
      println 'Done.'
}
```

Utilização de *plugin*

Bloco de configuração

Declaração de tarefa

Configuração do Gradle

- Como pré-requisito é necessário possuir pelo menos a versão 7 do Java JDK;

- Considerando a instalação manual:
  - Criar uma pasta Gradle na raiz do Sistema;
  - Realizar o download e extrair para a pasta criada anteriormente
  - Criar a variável de ambiente: **GRADLE_HOME** e incluir o diretório na **PATH**;
  - Abra a linha de comandos e teste o comando: **gradle –v**;

https://docs.gradle.org/current/userguide/installation.html#installing_gradle

Configuração do Gradle

- O Gradle é uma ferramenta de construção e automação de uso geral;
- Todas as instruções são expressas utilizando um build script;
- A maioria dos projetos Java são bastante similares:
  - Compilar os ficheiros fonte (.java);
  - Executar testes para validação do código implementado;
  - Criar o ficheiro JAR contendo todas as classes;
  - (…)
- O Gradle pode ser utilizado para automatizar estas tarefas utilizando *plugins*;
- O *plugin*: Java permite adicionar tarefas a um projeto de acordo com as tarefas estabelecidas no plugin de suporte à linguagem Java;

https://docs.gradle.org/current/userguide/tutorial_java_projects.html#sec:java_tutorial_where_to_nex

Configuração do Gradle

- O *plugin* Java do Gradle encapsula várias convenções, por exemplo, relativamente à localização dos ficheiros fonte;
- Não é obrigatória a utilização do *plugin* e várias das convenções seguidas podem ser personalizadas;
- Inicialmente é necessário criar uma pasta para o projeto e proceder à sua configuração com o Gradle:
  - gradle init para criar uma build

Configuração do Gradle

- A seguinte estrutura de diretórios é criada:
  - **build.gradle** – build script para o projeto atual
  - **Gradle/wrapper/gradle-wrapper.jar** – executável gradle
  - **Gradle/wrapper/gradle-wrapper.properties** – propriedades de configuração
  - **Gradlew** – wrapper script para sistemas Unix
  - **Gradlew.bat** – wrapper script para sistemas Windows
  - **settings.gradle** – script para configuração da build do Gradle
- Editar o ficheiro build.grade e especificar a utilização do plugin java:
  - **apply plugin: 'java'**
- O comando: gradle tasks apresenta as tarefas possíveis que podem ser aplicadas ao projeto Java;

Configuração do Gradle

- Por defeito o *plugin* java assume as seguintes propriedades:
  - O código fonte é armazenado na diretoria: **src/main/java**
  - O código fonte relativo aos testes é armazenado na diretoria: **src/test/java**
  - Na diretórica: **src/test/resources** encontram-se os recursos necessários para o projeto que serão posteriormente acrescentados na **classpath**;
  - Os ficheiros de output são armazenados no diretório: build, enquanto que o ficheiro JAR encontra-se no diretório: **build/libs**;
- Para realizar a construção do projeto basta executar: **gradle build**. Note que:
  - Terá de colocar os ficheiros .java na diretoria por defeito: src/main/java
  - Terá de colocar os restantes ficheiros de acordo com a sua localização por defeito;
  - Terá de configurar a variável de ambiente: JAVA_HOME.

Demonstração

- Considere os seguintes ficheiros Java de exemplo:

HelloWorld.java

```
package gradleExample;

public class HelloWorld {
  public static void main(String[] args) {
    Greeter greeter = new Greeter();
    System.out.println(greeter.sayHello());
  }
}
```

Greeter.java

```
package gradleExample;

public class Greeter {
  public String sayHello() {
    return "Hello world!";
  }
}
```

Exemplos retirados de: https://spring.io/guides/gs/gradle/

Demonstração

- Realize o processo de construção utilizando o procedimento descrito anteriormente;
- Execute o ficheiro JAR criado;
- Irá receber um problema relacionado com a configuração da classe **main** no ficheiro **Manifest**. Para resolver o problema basta acrescentar (para indicar qual a classe main do projeto) no ficheiro **build.gradle**:

```
jar {
    manifest {
        attributes 'Main-Class': ‘gradleExample.HelloWorld'
    }
}
```

Demonstração

- O exemplo do slide anterior apresenta um bloco de configuração, permitindo definir variáveis e estruturas de dados necessárias para as ações posteriores do processo de compilação;
- O bloco de configuração será executado durante a fase de configuração do Gradle;
- Considerando o exemplo do slide anterior, poderíamos enriquecer o ficheiro Manifest com meta informação sobre o projeto:

```
jar {
    manifest {
        attributes 'Specification-Title' : 'Hello World - Eng.Software'
        attributes 'Specification-Version' : '1.0'
        attributes 'Specification-Vendor' : 'ESTG'
        attributes 'Implementation-Title' : 'hello.HelloWorld'
        attributes 'Implementation-Version' : 'build01'
        attributes 'Implementation-Vendor' : 'ESTG'
                attributes 'Main-Class': 'hello.HelloWorld'
    }
}
```

Nota: Ler sobre o ficheiro Manifest: https://docs.oracle.com/javase/tutorial/deployment/jar/manifestindex.html

Demonstração

- O bloco dependencies é utilizado para referenciar dependências externas (ficheiros JAR) com o projeto;
- Podemos referenciar os ficheiros JAR localmente:

```
dependencies {
   compile name: 'joda-time-2.10'
}
```

- Indicando o repositório local:

```
repositories {
   flatDir {
       dirs 'libs'
   }
}
```

Demonstração

- Podemos utilizar um repositório para importar as dependências do projeto ou até para publicar dependências do projeto;
  - Exemplo utilizando o repositório Maven (https://mvnrepository.com/):

```
repositories {
    mavenCentral()
}
```

- Indicando as dependências:

```
dependencies {
    compile "joda-time:joda-time:2.10"
    }
```

Consultar o repositório Maven para identificar a declaração da dependência: https://mvnrepository.com/artifact/joda-time/joda-time/2.10

Demonstração

- É também possível indicar como o Gradle realiza a publicação do projeto utilizando a tarefa: **uploadArchives**:
  - Localmente:

```
uploadArchives {
    repositories {
        flatDir {
            dirs 'repos'
        }
    }
}
```

  - Utilizando um repositório Maven:

```
uploadArchives {
    repositories {
        mavenDeployer {
            repository(url: "…")
        }
    }
}
```

Referências Web

- https://guides.gradle.org/
- https://guides.gradle.org/creating-new-gradle-builds/
- https://docs.gradle.org/current/userguide/tutorial_java_projects.html#sec:java_tutorial_where_to_next
- Leitura recomendada:
  - https://guides.gradle.org/building-java-applications/

Exercício

- Criar uma classe WorldTime com um método getTimeByCountry que com base num país retorna a hora atual no formato: hh:mm;
- Deve utilizar a biblioteca Joda-Time: http://www.joda.org/joda-time/
- Crie os ficheiros .java necessários para responder ao problema e utilizando o Gradle, crie um ficheiro de configuração que:
  - Permita a execução do projeto;
  - Contemple a meta-informação base no ficheiro Manifest;
  - Após a compilação do projeto, utilize um repositório local para a publicação do ficheiro JAR;
  - Realize a geração do javadoc associado ao projeto (para isso deverá documentar o projeto);
  - Gere um ficheiro ZIP com o ficheiro JAR gerado e o javadoc associado.