

# Gradle - Tâches

script de compilation Gradle décrit un ou plusieurs projets. Chaque projet est composé de différentes tâches. Une tâche est un travail qui a build exécute. La tâche pourrait être la compilation de certaines classes, le stockage de fichiers de classe dans le dossier cible séparé, créant JAR, générant Javadoc, ou de publier des archives à des référentiels.

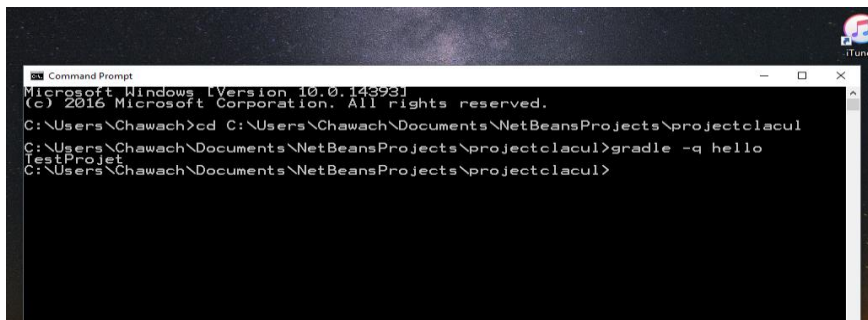
Ce chapitre explique ce qui est la tâche et la façon de générer et d'exécuter une tâche.

## Définition des tâches

Tâche est un mot-clé qui est utilisée pour définir une tâche dans le script de compilation. Jetez un oeil à l'exemple suivant qui représente une tâche **imprime TestProjet**. Copiez et enregistrez le script suivant dans un **build.gradle** fichier nommé **hello**.

```
task hello {  
    doLast {  
        println ' TestProjet '  
    }  
}
```

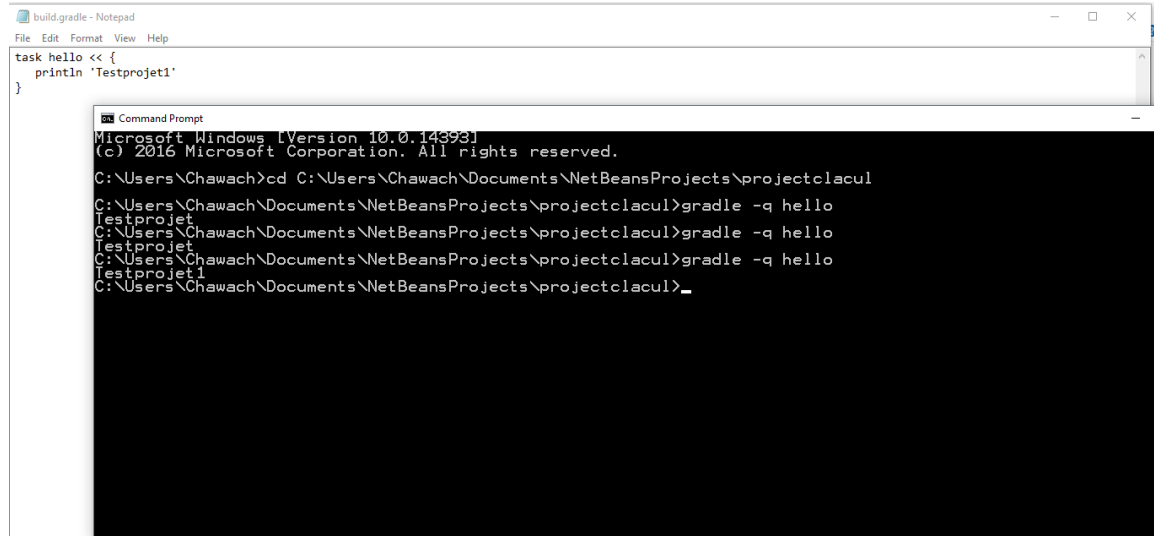
Exécutez la commande suivante dans l'invite de commande. Il exécute le script ci-dessus. Vous devez exécuter ce où le fichier build.gradle est stocké. Si la commande est exécutée avec succès, vous obtiendrez la sortie suivante.



Vous pouvez également utiliser des chaînes pour les noms de tâches. Jetez un oeil à le même exemple de bonjour. Ici, nous allons utiliser cordes comme tâche.

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```
task('hello') << {  
    println "TestProjet1"  
}
```

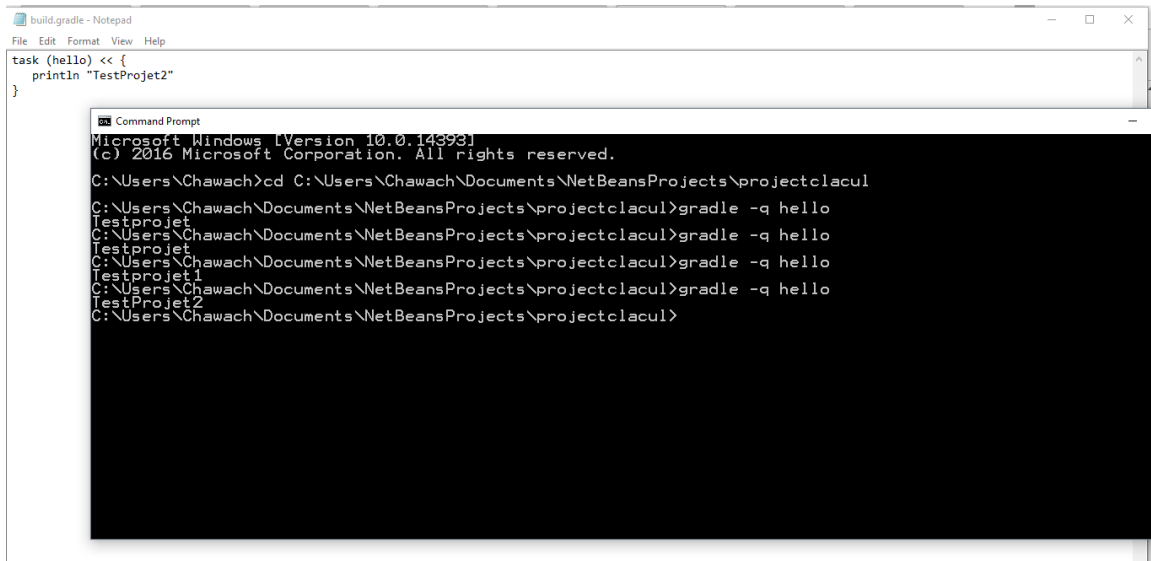


Vous pouvez exécuter le script ci - dessus en utilisant **-q hello commande gradle**.

Voici quelques variations dans la définition d'une tâche, jetez un coup d'oeil.  
L'exemple suivant définit une tâche

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```
task (hello) << {  
    println "TestProjet2"  
}
```



```
build.gradle - Notepad
File Edit Format View Help
task (hello) << {
    println "TestProjet2"
}

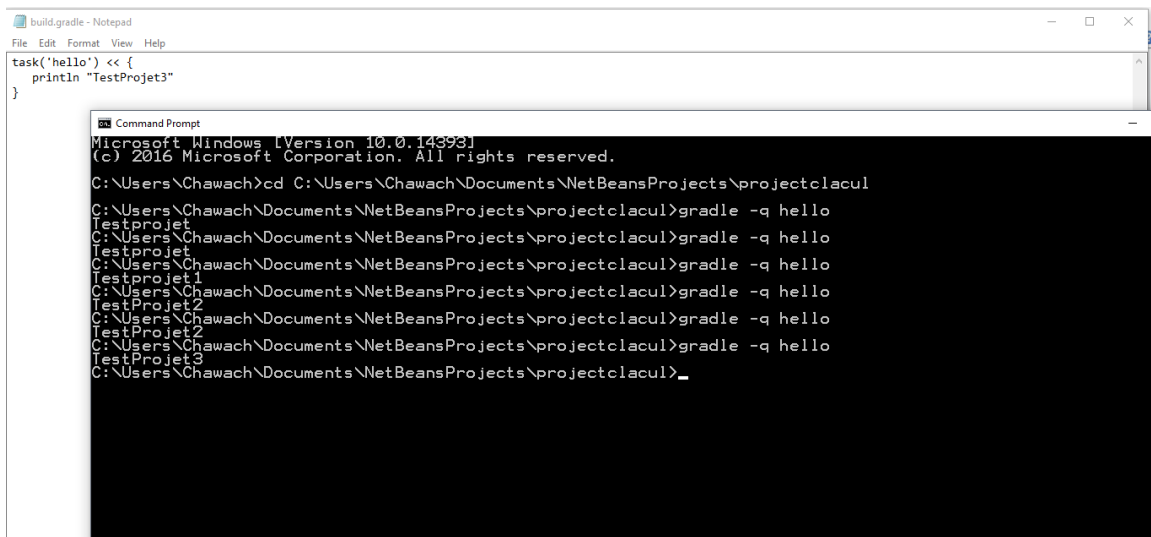
Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Chawach>cd C:\Users\Chawach\Documents\NetBeansProjects\projectclacul
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet1
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet2
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>
```

Vous pouvez également utiliser des chaînes pour les noms de tâches. Jetez un oeil à le même exemple de bonjour. Ici, nous allons utiliser cordes comme tâche.

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```
task('hello') << {
    println "TestProjet3"
}
```



```
build.gradle - Notepad
File Edit Format View Help
task('hello') << {
    println "TestProjet3"
}

Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Chawach>cd C:\Users\Chawach\Documents\NetBeansProjects\projectclacul
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet1
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet2
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet2
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
TestProjet3
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>
```

# Localisation Tâches

Si vous souhaitez localiser les tâches que vous avez définies dans le fichier de construction, alors vous devez utiliser les propriétés du projet de norme respectifs. Cela signifie que chaque tâche est disponible en tant que propriété du projet, en utilisant le nom de la tâche que le nom de la propriété.

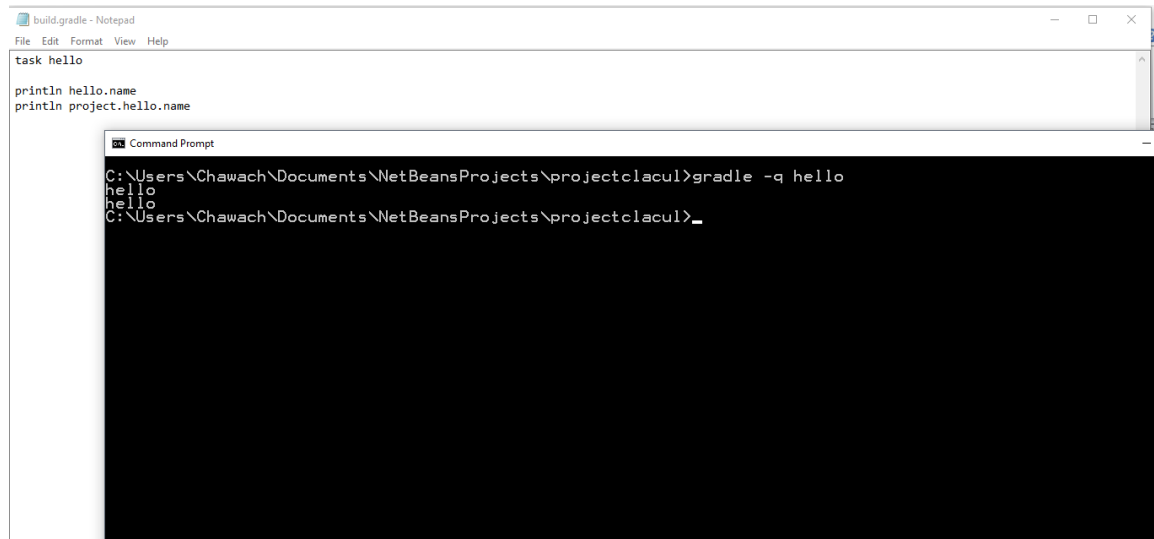
Jetez un oeil sur le code suivant qui accède à des tâches en tant que propriétés.

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```
task hello

println hello.name
println project.hello.name
```

Exécutez la commande suivante dans l'invite de commande. Vous devez exécuter ce où le fichier build.gradle est stocké.



Vous pouvez également utiliser toutes les propriétés grâce à la collecte des tâches.

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```
task hello

println tasks.hello.name
println tasks['hello'].name
```



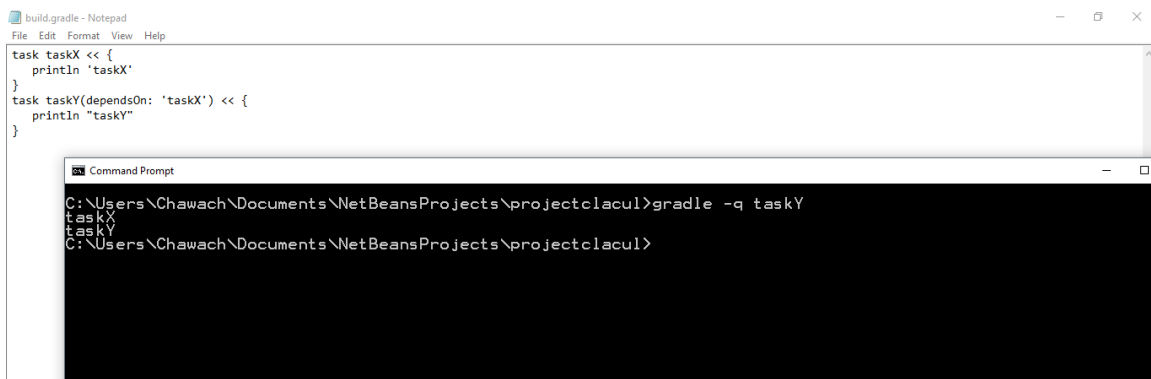
## Ajout d'une dépendance aux tâches

Vous pouvez faire une tâche dépendante d'une autre tâche, ce qui signifie que quand une tâche est effectuée seulement alors l'autre tâche va commencer. Chaque tâche est différenciée avec un nom de tâche. Collection des noms de tâches est appelé par sa collection de tâches. Pour faire référence à une tâche dans un autre projet, vous devez utiliser chemin du projet en tant que préfixe au nom de la tâche respective.

L'exemple suivant ajoute une dépendance de taskX à Tasky.

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```
task taskX << {
    println 'taskX'
}
task taskY(dependsOn: 'taskX') << {
    println "taskY"
}
```



L'exemple ci-dessus est l'ajout d'une dépendance à la tâche à l'aide de ses noms. Il y a une autre façon d'atteindre dépendance de tâche qui est de définir la dépendance à l'aide d'un objet Tâche.

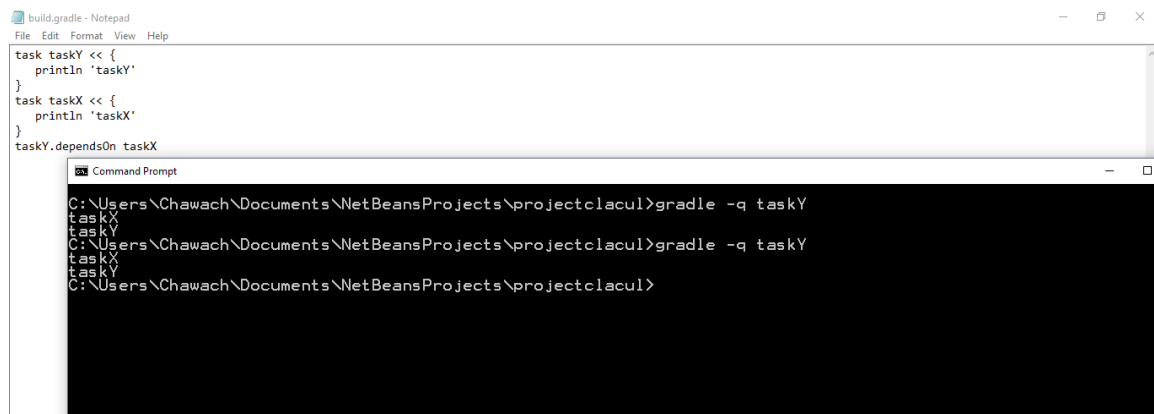
Prenons le même exemple de TaskY dépendant de taskX mais nous utilisons la tâche des objets au lieu des noms de référence de la tâche.

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```
task taskY << {
    println 'taskY'
}

task taskX << {
    println 'taskX'
}

taskY.dependsOn taskX
```



The screenshot displays two windows. The top window, titled 'build.gradle - Notepad', shows the following code:

```
task taskY << {
    println 'taskY'
}
task taskX << {
    println 'taskX'
}
taskY.dependsOn taskX
```

The bottom window, titled 'Command Prompt', shows the execution of the Gradle tasks. It displays the following output:

```
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q taskY
taskX
taskY
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q taskX
taskX
taskY
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>
```

## Ajout d'une description à une tâche

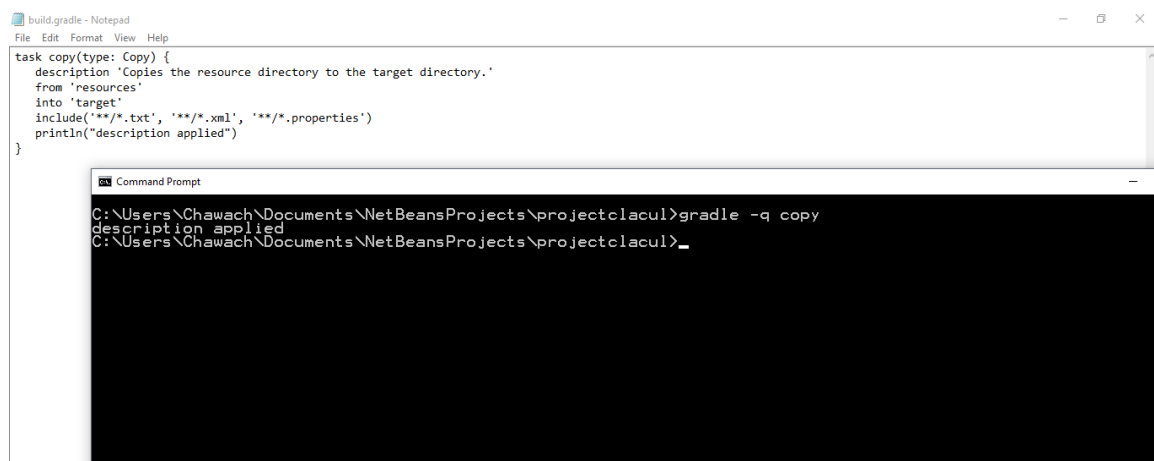
Vous pouvez ajouter une description à votre tâche. Cette description est affichée lors de l'exécution des **tâches Gradle**. Ceci est rendu possible à l'aide de la description des mots clés.

Copiez et enregistrez le code suivant dans le fichier **build.gradle**.

```

task copy(type: Copy) {
    description 'Copies the resource directory to the target di
rectory.'
    from 'resources'
    into 'target'
    include('**/*.txt', '**/*.xml', '**/*.properties')
    println("description applied")
}

```



## Importations par défaut pour les scripts Gradle

Gradle ajoute automatiquement un ensemble de déclarations d'importation aux scripts Gradle. La liste suivante vous montre les packages d'importation par défaut du script Gradle.

Ci - dessous sont les packages d'importation par défaut du script Gradle

```

import org.gradle.*
import org.gradle.api.*
import org.gradle.api.artifacts.*
import org.gradle.api.artifacts.cache.*
import org.gradle.api.artifacts.component.*

```

```
import org.gradle.api.artifacts.dsl.*
import org.gradle.api.artifacts.ivy.*
import org.gradle.api.artifacts.maven.*
import org.gradle.api.artifacts.query.*
import org.gradle.api.artifacts.repositories.*
import org.gradle.api.artifacts.result.*
import org.gradle.api.component.*
import org.gradle.api.credentials.*
import org.gradle.api.distribution.*
import org.gradle.api.distribution.plugins.*
import org.gradle.api.dsl.*
import org.gradle.api.execution.*
import org.gradle.api.file.*
import org.gradle.api.initialization.*
import org.gradle.api.initialization.dsl.*
import org.gradle.api.invocation.*
import org.gradle.api.java.archives.*
import org.gradle.api.logging.*
import org.gradle.api.plugins.*
import org.gradle.api.plugins.announce.*
import org.gradle.api.plugins.antlr.*
import org.gradle.api.plugins.buildcomparison.gradle.*
import org.gradle.api.plugins.jetty.*
import org.gradle.api.plugins.osgi.*
import org.gradle.api.plugins.quality.*
import org.gradle.api.plugins.scala.*
import org.gradle.api.plugins.sonar.*
import org.gradle.api.plugins.sonar.model.*
import org.gradle.api.publish.*
import org.gradle.api.publish.ivy.*
```



```
import org.gradle.api.publish.ivy.plugins.*
import org.gradle.api.publish.ivy.tasks.*
import org.gradle.api.publish.maven.*
import org.gradle.api.publish.maven.plugins.*
import org.gradle.api.publish.maven.tasks.*
import org.gradle.api.publish.plugins.*
import org.gradle.api.reporting.*
import org.gradle.api.reporting.components.*
import org.gradle.api.reporting.dependencies.*
import org.gradle.api.reporting.model.*
import org.gradle.api.reporting.plugins.*
import org.gradle.api.resources.*
import org.gradle.api.specs.*
import org.gradle.api.tasks.*
import org.gradle.api.tasks.ant.*
import org.gradle.api.tasks.application.*
import org.gradle.api.tasks.bundling.*
import org.gradle.api.tasks.compile.*
import org.gradle.api.tasks.diagnostics.*
import org.gradle.api.tasks.incremental.*
import org.gradle.api.tasks.javadoc.*
import org.gradle.api.tasks.scala.*
import org.gradle.api.tasks.testing.*
import org.gradle.api.tasks.testing.junit.*
import org.gradle.api.tasks.testing.testng.*
import org.gradle.api.tasks.util.*
import org.gradle.api.tasks.wrapper.*
import org.gradle.authentication.*
import org.gradle.authentication.http.*
import org.gradle.buildinit.plugins.*
```

```
import org.gradle.buildinit.tasks.*
import org.gradle.external.javadoc.*
import org.gradle.ide.cdt.*
import org.gradle.ide.cdt.tasks.*
import org.gradle.ide.visualstudio.*
import org.gradle.ide.visualstudio.plugins.*
import org.gradle.ide.visualstudio.tasks.*
import org.gradle.ivy.*
import org.gradle.jvm.*
import org.gradle.jvm.application.scripts.*
import org.gradle.jvm.application.tasks.*
import org.gradle.jvm.platform.*
import org.gradle.jvm.plugins.*
import org.gradle.jvm.tasks.*
import org.gradle.jvm.tasks.api.*
import org.gradle.jvm.test.*
import org.gradle.jvm.toolchain.*
import org.gradle.language.assembler.*
import org.gradle.language.assembler.plugins.*
import org.gradle.language.assembler.tasks.*
import org.gradle.language.base.*
import org.gradle.language.base.artifact.*
import org.gradle.language.base.plugins.*
import org.gradle.language.base.sources.*
import org.gradle.language.c.*
import org.gradle.language.c.plugins.*
import org.gradle.language.c.tasks.*
import org.gradle.language.coffeescript.*
import org.gradle.language.cpp.*
import org.gradle.language.cpp.plugins.*
```

```
import org.gradle.language.cpp.tasks.*
import org.gradle.language.java.*
import org.gradle.language.java.artifact.*
import org.gradle.language.java.plugins.*
import org.gradle.language.java.tasks.*
import org.gradle.language.javascript.*
import org.gradle.language.jvm.*
import org.gradle.language.jvm.plugins.*
import org.gradle.language.jvm.tasks.*
import org.gradle.language.nativeplatform.*
import org.gradle.language.nativeplatform.tasks.*
import org.gradle.language.objectivec.*
import org.gradle.language.objectivec.plugins.*
import org.gradle.language.objectivec.tasks.*
import org.gradle.language.objectivec.cpp.*
import org.gradle.language.objectivec.cpp.plugins.*
import org.gradle.language.objectivec.cpp.tasks.*
import org.gradle.language.rc.*
import org.gradle.language.rc.plugins.*
import org.gradle.language.rc.tasks.*
import org.gradle.language.routes.*
import org.gradle.language.scala.*
import org.gradle.language.scala.plugins.*
import org.gradle.language.scala.tasks.*
import org.gradle.language.scala.toolchain.*
import org.gradle.language.twirl.*
import org.gradle.maven.*
import org.gradle.model.*
import org.gradle.nativeplatform.*
import org.gradle.nativeplatform.platform.*
```

```
import org.gradle.nativeplatform.plugins.*
import org.gradle.nativeplatform.tasks.*
import org.gradle.nativeplatform.test.*
import org.gradle.nativeplatform.test.cunit.*
import org.gradle.nativeplatform.test.cunit.plugins.*
import org.gradle.nativeplatform.test.cunit.tasks.*
import org.gradle.nativeplatform.test.googletest.*
import org.gradle.nativeplatform.test.googletest.plugins.*
import org.gradle.nativeplatform.test.plugins.*
import org.gradle.nativeplatform.test.tasks.*
import org.gradle.nativeplatform.toolchain.*
import org.gradle.nativeplatform.toolchain.plugins.*
import org.gradle.platform.base.*
import org.gradle.platform.base.binary
import org.gradle.platform.base.component.*
import org.gradle.platform.base.plugins.*
import org.gradle.platform.base.test.*
import org.gradle.play.*
import org.gradle.play.distribution.*
import org.gradle.play.platform.*
import org.gradle.play.plugins.*
import org.gradle.play.tasks.*
import org.gradle.play.toolchain.*
import org.gradle.plugin.use.*
import org.gradle.plugins.ear.*
import org.gradle.plugins.ear.descriptor.*
import org.gradle.plugins.ide.api.*
import org.gradle.plugins.ide.eclipse.*
import org.gradle.plugins.ide.idea.*
import org.gradle.plugins.javascript.base.*
```

```
import org.gradle.plugins.javascript.coffeescript.*
import org.gradle.plugins.javascript.envjs.*
import org.gradle.plugins.javascript.envjs.browser.*
import org.gradle.plugins.javascript.envjs.http.*
import org.gradle.plugins.javascript.envjs.http.simple.*
import org.gradle.plugins.javascript.jshint.*
import org.gradle.plugins.javascript.rhino.*
import org.gradle.plugins.javascript.rhino.worker.*
import org.gradle.plugins.signing.*
import org.gradle.plugins.signing.signatory.*
import org.gradle.plugins.signing.signatory.pgp.*
import org.gradle.plugins.signing.type.*
import org.gradle.plugins.signing.type.pgp.*
import org.gradle.process.*
import org.gradle.sonar.runner.*
import org.gradle.sonar.runner.plugins.*
import org.gradle.sonar.runner.tasks.*
import org.gradle.testing.jacoco.plugins.*
import org.gradle.testing.jacoco.tasks.*
import org.gradle.testkit.runner.*
import org.gradle.util.*
```

## Rédaction personnalisée Plugins

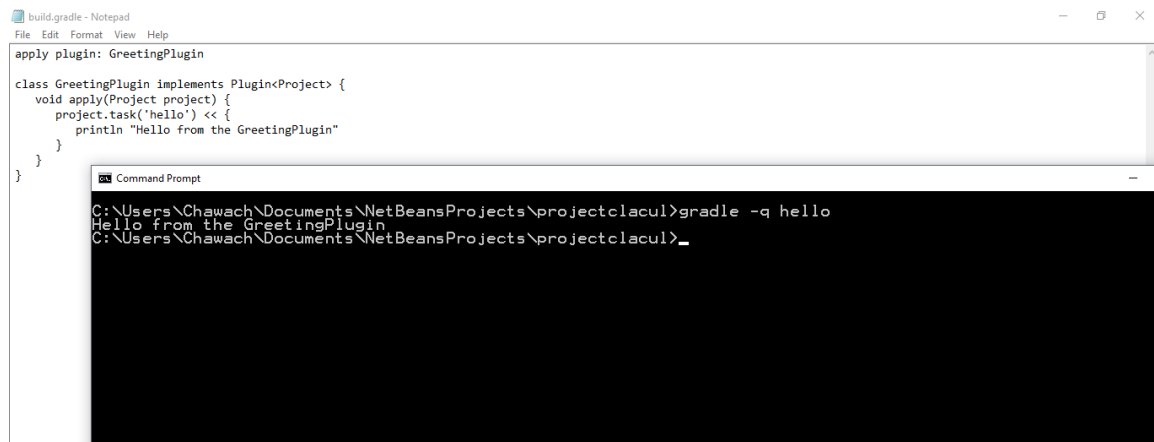
Lors de la création d'un plugin personnalisé, vous devez écrire une implémentation de plug-in. Gradle instancie le plugin et appelle l'instance plug-in en utilisant la méthode `Plugin.apply()`. L'exemple suivant contient un plugin d'accueil, ce qui ajoute une tâche `bonjour` au projet. Jetez un oeil sur le code suivant. Utilisez ce code dans le fichier **build.gradle**.

```
apply plugin: GreetingPlugin
```

```

class GreetingPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.task('hello') << {
            println "Hello from the GreetingPlugin"
        }
    }
}

```



## Obtenir l'entrée de la construction

La plupart des plugins ont besoin du soutien de configuration à partir du script de compilation. Le projet Gradle a un objet associé «ExtensionContainer» qui permet de suivre tous les réglages et les propriétés étant transmises à des plugins.

Ajoutons un objet simple extension du projet. Ici, nous ajoutons un objet d'extension de voeux pour le projet, ce qui vous permet de configurer le message d'accueil.

Utilisez ce code dans le fichier **build.gradle**.

```

apply plugin: GreetingPlugin

greeting.message = 'Hi from Gradle'

class GreetingPlugin implements Plugin<Project> {
    void apply(Project project) {
        // Add the 'greeting' extension object
    }
}

```

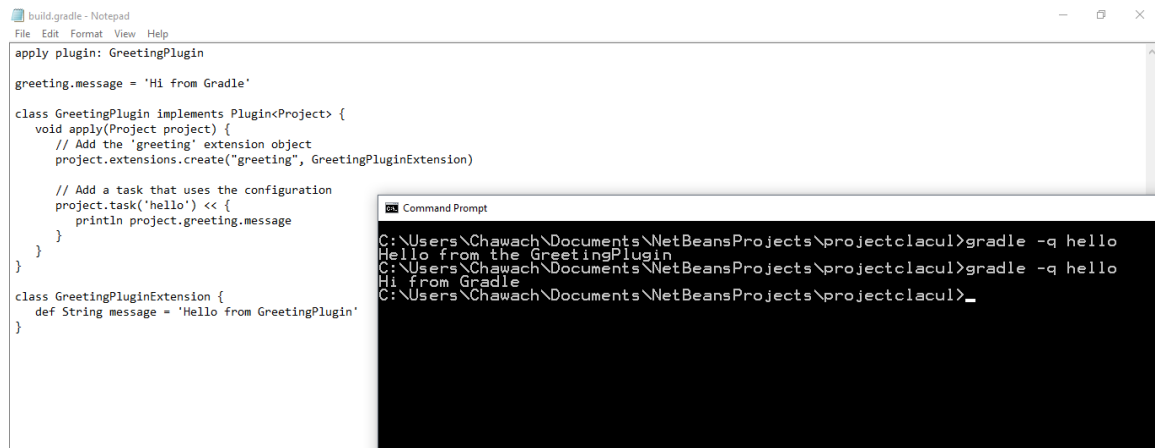
```

        project.extensions.create("greeting", GreetingPluginExtension)

        // Add a task that uses the configuration
        project.task('hello') << {
            println project.greeting.message
        }
    }
}

class GreetingPluginExtension {
    def String message = 'Hello from GreetingPlugin'
}

```



The screenshot shows a Notepad window titled 'build.gradle - Notepad' with the following content:

```

apply plugin: GreetingPlugin

greeting.message = 'Hi from Gradle'

class GreetingPlugin implements Plugin<Project> {
    void apply(Project project) {
        // Add the 'greeting' extension object
        project.extensions.create("greeting", GreetingPluginExtension)

        // Add a task that uses the configuration
        project.task('hello') << {
            println project.greeting.message
        }
    }
}

class GreetingPluginExtension {
    def String message = 'Hello from GreetingPlugin'
}

```

Overlaid on the bottom right is a Command Prompt window showing the execution of the 'gradle hello' command twice. The first execution outputs 'Hello from the GreetingPlugin', and the second execution outputs 'Hi from Gradle'.

Dans cet exemple, GreetingPlugin est un ancien objet Groovy simple avec un champ appelé «message». L'objet d'extension est ajouté à la liste des plug-in avec le nom salutation. Cet objet devient alors disponible en tant que propriété de projet avec le même nom que l'objet d'extension.

Gradle ajoute une fermeture de configuration pour chaque objet d'extension, de sorte que vous pouvez regrouper les paramètres ensemble. Jetez un oeil sur le code suivant. Utilisez ce code dans le fichier **build.gradle**.

```

apply plugin: GreetingPlugin

```

```

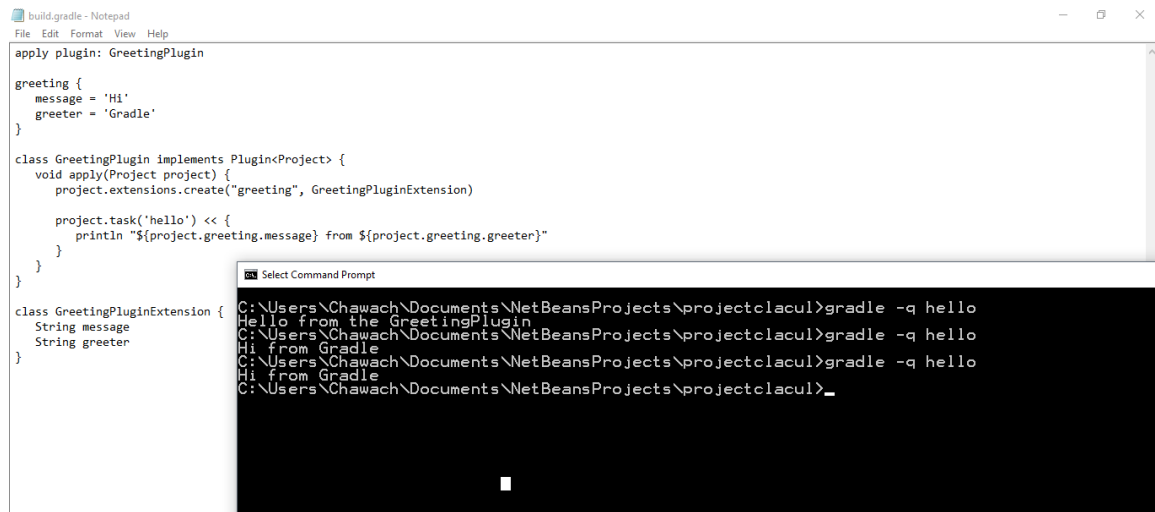
greeting {
    message = 'Hi'
    greeter = 'Gradle'
}

class GreetingPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.extensions.create("greeting", GreetingPluginExtension)

        project.task('hello') << {
            println "${project.greeting.message} from ${project.greeting.greeter}"
        }
    }
}

class GreetingPluginExtension {
    String message
    String greeter
}

```



The screenshot shows a Notepad window titled 'build.gradle - Notepad' with the following content:

```

apply plugin: GreetingPlugin

greeting {
    message = 'Hi'
    greeter = 'Gradle'
}

class GreetingPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.extensions.create("greeting", GreetingPluginExtension)

        project.task('hello') << {
            println "${project.greeting.message} from ${project.greeting.greeter}"
        }
    }
}

class GreetingPluginExtension {
    String message
    String greeter
}

```

Below the Notepad window is a Command Prompt window titled 'Select Command Prompt' with the following output:

```

C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
Hello from the GreetingPlugin
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
Hi from Gradle
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q hello
Hi from Gradle
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>_

```

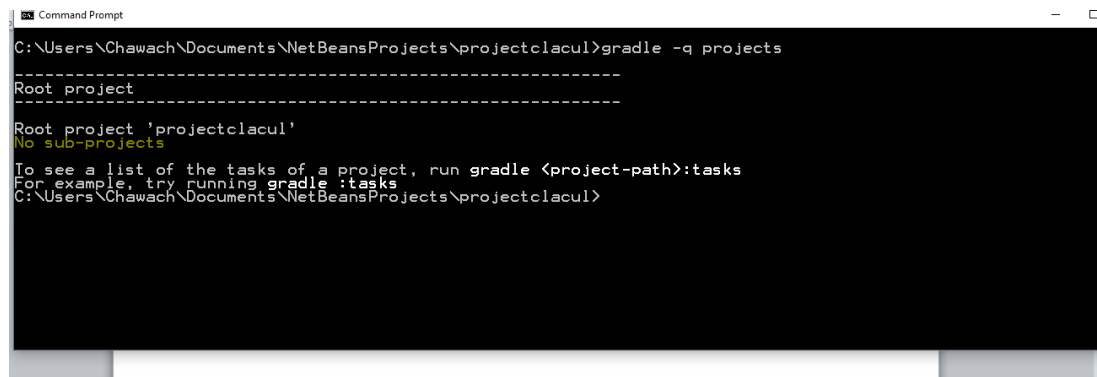


# Obtenir l'information Construire

Gradle fournit plusieurs tâches intégrées pour récupérer les détails de l'information concernant la tâche et le projet. Cela peut être utile pour comprendre la structure et les dépendances de votre construction et pour les problèmes de débogage. Vous pouvez utiliser le plugin projet rapport pour ajouter des tâches à votre projet, qui va générer ces rapports.

## Liste des projets

Vous pouvez lister la hiérarchie du projet du projet sélectionné et leurs sous - projets utilisant **gradle commande projets -q**. Voici l'exemple, utilisez la commande suivante pour répertorier tous les projets dans le fichier de construction.



```
Command Prompt
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle -q projects

-----
Root project
-----

Root project 'projectclacul'
No sub-projects

To see a list of the tasks of a project, run gradle <project-path>:tasks
For example, try running gradle :tasks
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>
```

## Gradle - Construire un projet Groovy

Le dossier complet de script de compilation est la suivante. Copiez le code suivant dans le fichier **build.gradle**.

```
apply plugin: 'groovy'

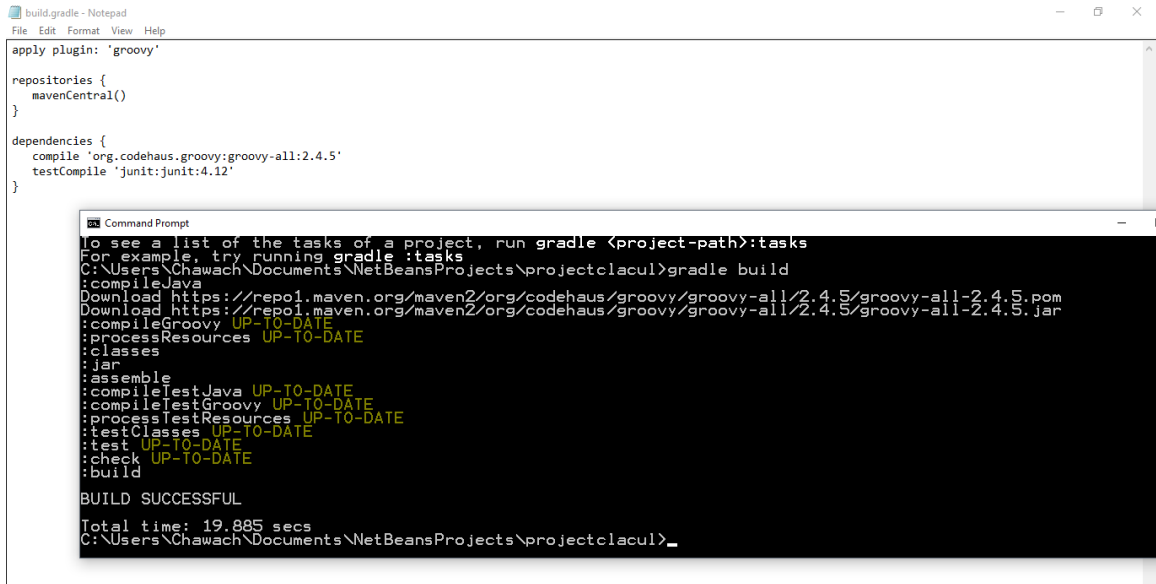
repositories {
    mavenCentral()
}

dependencies {
```

```
compile 'org.codehaus.groovy:groovy-all:2.4.5'

testCompile 'junit:junit:4.12'

}
```



The screenshot shows a Notepad window titled 'build.gradle - Notepad' with the following content:

```
apply plugin: 'groovy'

repositories {
    mavenCentral()
}

dependencies {
    compile 'org.codehaus.groovy:groovy-all:2.4.5'
    testCompile 'junit:junit:4.12'
}
```

Below the Notepad window is a Command Prompt window showing the output of a gradle build:

```
to see a list of the tasks of a project, run gradle <project-path>:tasks
For example, try running gradle :tasks
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>gradle build
:compileJava
Download https://repo1.maven.org/maven2/org/codehaus/groovy/groovy-all/2.4.5/groovy-all-2.4.5.pom
Download https://repo1.maven.org/maven2/org/codehaus/groovy/groovy-all/2.4.5/groovy-all-2.4.5.jar
:compileGroovy UP-TO-DATE
:processResources UP-TO-DATE
:classes
:jar
:assemble
:compileTestJava UP-TO-DATE
:compileTestGroovy UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE
:check UP-TO-DATE
:build
BUILD SUCCESSFUL
Total time: 19.885 secs
C:\Users\Chawach\Documents\NetBeansProjects\projectclacul>
```

Par défaut Projet Mise en page des projets Groovy

Le plugin Groovy suppose une certaine configuration de votre projet Groovy.

- src / main / groovy contient le code source Groovy
- src / test / groovy contient les essais Groovy
- src / main / java contient le code source Java
- src / test / java contient les tests de Java

## Déclarer vos dépendances

Gradle suit une syntaxe particulière pour définir des dépendances. Le script suivant définit deux dépendances, on est Hibernate noyau 3.6.7 et second est Junit avec la version 4.0 et versions ultérieures. Jetez un oeil sur le code suivant. Utilisez ce code dans le fichier **build.gradle**.

```
apply plugin: 'java'

repositories {
    mavenCentral()
}
```

```
dependencies {  
    compile group: 'org.hibernate', name: 'hibernate-core', version: '3.6.7.Final'  
    testCompile group: 'junit', name: 'junit', version: '4.+'  
}
```

## Configurations de dépendance

configuration de dépendance est rien, mais définit un ensemble de dépendances. Vous pouvez utiliser cette fonction pour déclarer des dépendances externes, que vous souhaitez télécharger à partir du Web. Ceci définit les configurations standard différentes suivantes.

- **Compiler - Les dépendances nécessaires pour compiler les sources du projet de production.**
- **Runtime - Les dépendances requises par les classes de production lors de l' exécution.** Par défaut, comprend également les dépendances de compilation.
- **Test de Compile - Les dépendances nécessaires pour compiler la source du projet de test.** Par défaut, il inclut des classes de production compilées et les dépendances de temps de compilation.
- **Test de Runtime - Les dépendances requises pour exécuter les tests.** Par défaut, il comprend exécution et de compilation de test dépendances.

## Référentiels

Tout en ajoutant des dépendances externes. Gradle regarde pour eux dans un référentiel. Un référentiel est juste une collection de fichiers, organisés par le groupe, le nom et la version. Par défaut, Gradle ne définit pas de dépôts. Nous devons définir au moins un référentiel explicitement. Le code suivant définit comment définir référentiel maven. Utilisez ce code dans le fichier **build.gradle**.

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

Le code suivant est de définir maven à distance. Utilisez ce code dans le fichier **build.gradle**.

```

repositories {
    maven {
        url "http://repo.mycompany.com/maven2"
    }
}

```

**Vous pouvez taper gradle -h dans command line pour avoir l'aide.**

```

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Chawach>gradle -h

USAGE: gradle [option...] [task...]

-?, -h, --help           Shows this help message.
-a, --no-rebuild         Do not rebuild project dependencies.
-b, --build-file         Specifies the build file.
-c, --settings-file     Specifies the settings file.
--configure-on-demand    Only relevant projects are configured in this build run. This means faster build for large multi-project builds. [incubating]
--console               Specifies which type of console output to generate. Values are 'plain', 'auto' (default) or 'rich'.
--continue              Continues task execution after a task failure.
-D, --system-prop       Set system property of the JVM (e.g. -Dmyprop=myvalue).
-d, --debug             Log in debug mode (includes normal stacktrace).
--daemon               Uses the Gradle Daemon to run the build. Starts the Daemon if not running.
--foreground            Starts the Gradle Daemon in the foreground. [incubating]
-g, --gradle-user-home  Specifies the gradle user home directory.
--gui                  Launches the Gradle GUI.
-I, --init-script       Specifies an initialization script.
-i, --info              Set log level to info.
--include-build         Includes the specified build in the composite. [incubating]
-m, --dry-run           Runs the builds with all task actions disabled.
--max-workers           Configure the number of concurrent workers Gradle is allowed to use. [incubating]
--no-daemon             Do not use the Gradle Daemon to run the build.
--offline              The build should operate without accessing network resources.

```