Writing Custom Gradle Tasks

Table of Contents

Tasks are the cornerstone of getting things done in Gradle. They represent single atomic pieces of work within a build such as creating a JAR or linking an executable. This guide will walk you through the process of customising your build using small, bespoke tasks.

What you'll create

You'll start by creating an ad-hoc Gradle task that prints *Hello, World* to the console. You'll then move to making it configurable to print any message. In this process you'll learn about ad-hoc tasks and custom task types.

What you'll need

- About 20 minutes
- A text editor or IDE
- A Java Development Kit (JDK), version 1.7 or better
- A Gradle distribution (https://docs.gradle.org/3.4/userguide//installation.html#sec:download), version 3.4 or better

Create an ad-hoc task

In a new folder create a `build.gradle` and enter the following

build.gradle

GROOVY

```
task hello {  1
  doLast {  2
      println 'Hello, World!'
  }
}
```

1. Create a new ad-hoc task called `hello`.
2. Add a task action to print to the console.

Save the file and on the command-line and type:

```
gradle tasks
```

Save the file and on the command-line type `gradle tasks --all`. Your new task will appear under `Other tasks`.

Verifying that your task has been created

```
Other tasks
-----------
hello
```

Run your task

```
gradle hello
```

It will produce a simple output.

Output from executing your ad-hoc task

```
:hello  1
Hello, World  2
```

1. This indicates that your `hello` task has executed.
2. This is the output from your ad-hoc task.

Congratulations! You have you added your first ad-hoc task.

Do not use << as an alternative to doLast

GROOVY

```
task hello << {
  println 'Hello, World!'
}
```

You may comes across an alternative form for creating ad-hoc tasks in some older documentation and blog posts whereby the left-shift operator (<<) is used.

This form has caused too much confusion in the past especially among people that are new to Gradle. It also reduces readability. We have deprecated this behaviour (https://docs.gradle.org/3.2/release-notes#the-left-shift-operator-on-the-task-interface) and the functionality will be removed in Gradle 5.0.

Add a task description

Although you have tested your new ad-hoc task and know that it works, it is good practice to tell others who will use your build script what the purpose of your new task is. It is also useful to categorise your task.

When you ran `gradle tasks --all` earlier you would have seen other tasks in the listing which contain descriptions and are grouped according to function. Also (and since Gradle 3.3) if a task does not have a group it will not be listed unless `--all` is specified.

Well-documented tasks

```
Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating]
wrapper - Generates Gradle wrapper files. [incubating]
```

This is achieved by setting the `group` and `description` properties on the task. Edit your ad-hoc task and add the following:

build.gradle

GROOVY

```
task hello {
  group 'Welcome'
  description 'Produces a greeting'

  doLast {
      println 'Hello, World'
  }
}
```

Run `gradle tasks` again.

Output from 'tasks' task

```
Welcome tasks
-------------
hello - Produces a greeting
```

Well done!

Make the message configurable

This is the point where you need to convert your ad-hoc task into a custom task type, and this is achieved by creating a class within the build script.

Return to your build script and add the following class and refactor your `hello` task.

build.gradle

GROOVY

```
class Greeting extends DefaultTask {   1  2
    String message  3
    String recipient

    @TaskAction  4
    void sayGreeting() {
        println "${message}, ${recipient}!"  5
    }
}

task hello ( type : Greeting ) {  6
    group 'Welcome'
    description 'Produces a world greeting'
    message 'Hello'  7
    recipient 'World'
}
```

1 As the build DSL in a `build.gradle` file is a Groovy-base DSL, the class will be a Groovy class.

2 Although other task classes from the Gradle API can be used in specific circumstances, extending {javadoc}/org/gradle/api/DefaultTask.html[DefaultTask] is the most common scenario.

3 Adding `message` and `recipient` properties allow instances of this custom task type to be configurable

4 Annotate the default task action.

5 Print the message using a standard Groovy interpolated string.

6 Specify the task type by referencing the class type `Greeting` you have added above.

7 Configure the message and the recipient.

Test your modification.

```
gradle hello
```

You should see the same output

Output after conversion to a custom task type

```
:hello
Hello, World!
```

Now that you have the custom task type, you can add additional tasks. Add a German version of the greeting by just creating an additional task.

Adding a second task

GROOVY

```
task gutenTag( type : Greeting ) {
    group 'Welcome'
    description 'Produces a German greeting'
    message 'Guten Tag'
    recipient 'Welt'
}
```

No assignment is required during configuration as Gradle decorates the properties to allow for a more declarative DSL.

Run `gradle tasks` again to verify that the new task has been added.

Output of 'gradle tasks' after adding second task.

```
Welcome tasks
-------------
hello - Produces a greeting
gutenTag - Produces a German greeting
```

Finally, run the new task by doing `gradle gutenTag`

Output of your second task.

```
:gutenTag
Guten Tag, Welt!
```

Summary

That's it! You've worked through the steps necessary to create a custom Gradle Task. You should now have learned how to

- Create an ad-hoc task and add an action using `doLast`.
- Document a task.
- Convert an ad-hoc task to a custom Gradle task type and creating task instances.
- Using `@TaskAction` to set a default action for a task type.

Next steps

- Having classes in a build script will soon lead to a messy and potentially unmaintainable build script. Learn how to organize your build logic (https://docs.gradle.org/3.4/userguide/organizing_build_logic.html).
- Read more about using tasks (https://docs.gradle.org/current/userguide/tutorial_using_tasks.html), and predefined tasks and task types (https://docs.gradle.org/3.4/dsl/org.gradle.api.Task.html)

Help improve this guide

Have feedback or a question? Found a typo? Like all Gradle guides, help is just a GitHub Issue away. Please add an issue or pull request to the gradle-guides/writing-gradle-tasks (https://github.com/gradle-guides/writing-gradle-tasks/) and we'll get back to you.

Last updated 2017-02-21 12:18:34 UTC