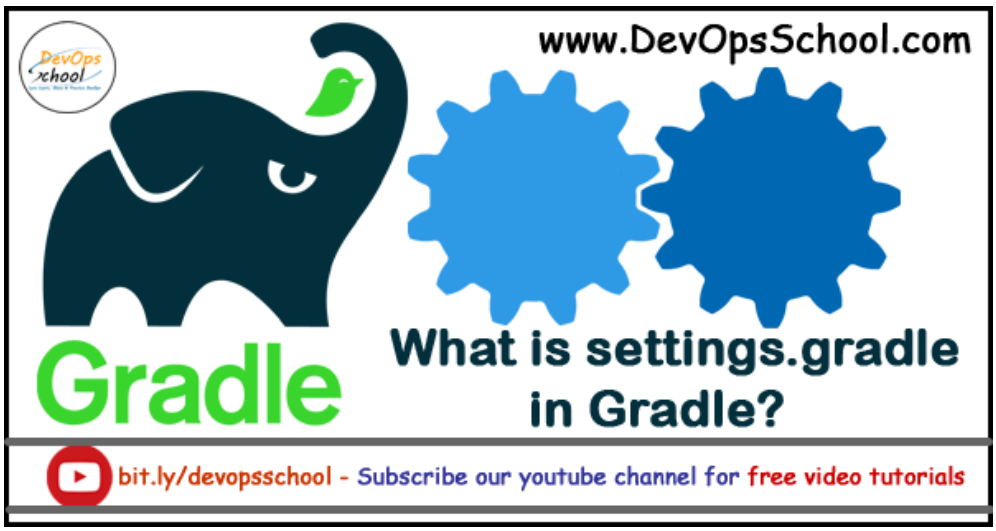
GRADLE

What is settings.gradle in Gradle?

RAJESH KUMAR FEBRUARY 2, 2020 COMMENTS OFF



Spread the Knowledge

A gradle build has three important files. build.gradle, gradle.properties and settings.gradle

Gradle build lifecycle consist of three phases: initialization, configuration, and execution. settings.gradle get evaluated in the initialization phase.

> Gradle looks at the settings.gradle file in order to identify the different projects. At the end of the initialization phase, Gradle creates an instance of org.gradle.api.Project corresponding to each of these projects.
>
> The settings.gradle file is a Groovy script, just like the build.gradle file. Only one settings.gradle script will be executed in each build (in comparison to multiple build.gradle scripts in multi-project builds).
>
> The settings.gradle script will be executed before any build.gradle script and even before the Project instances are created. Therefore, it is evaluated against a Settings object.

settings.gradle is very important in multi-module projects.

A multi-module project has **one main module** and many submodules. It has this layout:

```
(root)
  +- settings.gradle
  +- build.gradle          # optional (commonly present)
  +- gradle.properties     # optional
  +-- buildSrc/            # optional
  |     +- build.gradle
  |     +-- src/...
  +-- my-gradle-stuff/     # optional
  |     +- utils.gradle    # optional
  +-- sub-a/
  |     +- build.gradle
  |     +- src/
  +-- sub-b/
        +- build.gradle
        +- src/
```

submodules can also be located deeper in subfolders, but without modifying code in settings.gradle, their name will include the name of such folders.

settings.gradle

The main role of settings.gradle is to define all included submodules and to mark the directory root of a tree of modules, so you can only have one `settings.gradle` file in a multi-module project.

```
rootProject.name = 'project-x'

include 'sub-a', 'sub-b'
```

settings.gradle may cotains following Properties.

Property	Description
`buildCache`	The build cache configuration.
`extensions`	The container of extensions.
`gradle`	The `Gradle` instance for the current build.
`pluginManager`	The plugin manager for this plugin aware object.
`plugins`	The container of plugins that have been applied to this object.
`rootDir`	The root directory of the build. The root directory is the project directory of the root project.
`rootProject`	The root project of the build.
`settings`	Returns this settings object.
`settingsDir`	The settings directory of the build. The settings directory is the directory containing the settings file.
`startParameter`	The set of parameters used to invoke this instance of Gradle.

Property	Description
`buildCache`	The build cache configuration.
`extensions`	The container of extensions.
`gradle`	The `Gradle` instance for the current build.
`pluginManager`	The plugin manager for this plugin aware object.
`plugins`	The container of plugins that have been applied to this object.
`rootDir`	The root directory of the build. The root directory is the project directory of the root project.
`rootProject`	The root project of the build.

Property	Description
`settings`	Returns this settings object.
`settingsDir`	The settings directory of the build. The settings directory is the directory containing the settings file.
`startParameter`	The set of parameters used to invoke this instance of Gradle.

settings.gradle may cotains following Methods.

Methods

Method	Description
`apply(closure)`	Applies zero or more plugins or scripts.
`apply(options)`	Applies a plugin or script, using the given options provided as a map. Does nothing if the plugin has already been applied.
`apply(action)`	Applies zero or more plugins or scripts.
`buildCache(action)`	Configures build cache.
`findProject(projectDir)`	Returns the project with the given project directory.
`findProject(path)`	Returns the project with the given path.
`include(projectPaths)`	Adds the given projects to the build. Each path in the supplied list is treated as the path of a project to add to the build. Note that these path are not file paths, but instead specify the location of the new project in the project hierarchy. As such, the supplied paths must use the ':' character as separator (and NOT '/').
`includeBuild(rootProject)`	Includes a build at the specified path to the composite build.
`includeBuild(rootProject, configuration)`	Includes a build at the specified path to the composite build, with the supplied configuration.
`includeFlat(projectNames)`	Adds the given projects to the build. Each name in the supplied list is treated as the name of a project to add to the build.
`project(projectDir)`	Returns the project with the given project directory.
`project(path)`	Returns the project with the given path.