- можно вынести определение задачи в отдельный groovy-класс:

```
class ReleaseVersionTask extends DefaultTask {
    @Input Boolean release
    @OutputFile File destFile

    ReleaseVersionTask() {
        group = 'versioning'
        description = 'Makes project a release version.'
    }

    @TaskAction
    void start() {
        project.version.release = true
        ant.propertyfile(file: destFile) {
            entry(key: 'release', type: 'string', operation: '=', value: 'true')
        }
    }
}
```

Declaring custom task's inputs/ outputs through annotations

Setting task's group and description properties in the constructor

Annotation declares method to be executed

Writing a custom task that extends Gradle's default task implementation

- используем в build.gradle нашу задачу так:

```
task makeReleaseVersion(type: ReleaseVersionTask) {
    release = version.release
    destFile = versionFile
}
```

Setting custom task properties

Defining an enhanced task of type ReleaseVersionTask

- When you run Gradle, it checks for the existence of a directory called buildSrc. Gradle then automatically compiles and tests this code and puts it in the classpath of your build script. You don't need to provide any further instruction. This can be a good place to add your custom tasks and plugins. For multi-project builds there can be only one buildSrc directory, which has to be in the root project directory.
- По умолчанию градл применяет следующий build.gradle к buildSrc:

```
apply plugin: 'groovy'
dependencies {
    compile gradleApi()
    compile localGroovy()
}
```