Docs

[User Manual](http://docs.google.com/userguide/userguide.html)

[Guides and Tutorials](https://guides.gradle.org)

[DSL Reference](http://docs.google.com/dsl/)

[Javadoc](http://docs.google.com/javadoc/)

[Release Notes](http://docs.google.com/release-notes.html)

[Forums](https://discuss.gradle.org/)

[Training](https://gradle.org/training/)

[Try Gradle Enterprise](https://gradle.com/enterprise)

[PDF](http://docs.google.com/userguide/userguide.pdf)

* [User Manual Home](http://docs.google.com/userguide/userguide.html)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Installing Gradle](http://docs.google.com/userguide/installation.html)
* [Tutorials](https://guides.gradle.org/)

### Reference

* [Groovy DSL Reference](http://docs.google.com/dsl/)
* [Gradle API Javadoc](http://docs.google.com/javadoc/)
* [Core Plugins](http://docs.google.com/userguide/plugin_reference.html)
* [Gradle & Third-party Tools](http://docs.google.com/userguide/third_party_integration.html)

### Getting Started

* [Creating New Gradle Builds](https://guides.gradle.org/creating-new-gradle-builds/)
* [Creating Build Scans](https://guides.gradle.org/creating-build-scans/)
* [Migrating From Maven](https://guides.gradle.org/migrating-from-maven/)

### Running Gradle Builds

* [Command-Line Interface](http://docs.google.com/userguide/command_line_interface.html)
* [Customizing Execution](#gjdgxs)
  + [Configuring the Build Environment](http://docs.google.com/userguide/build_environment.html)
  + [Configuring the Gradle Daemon](http://docs.google.com/userguide/gradle_daemon.html)
  + [Initialization Scripts](http://docs.google.com/userguide/init_scripts.html)
* [Directory Layout](http://docs.google.com/userguide/directory_layout.html)
* [Executing Multi-Project Builds](http://docs.google.com/userguide/intro_multi_project_builds.html)
* [Gradle Wrapper](http://docs.google.com/userguide/gradle_wrapper.html)
* [Troubleshooting](http://docs.google.com/userguide/troubleshooting.html)
* [Using Build Scans](https://docs.gradle.com/build-scan-plugin)
* [Enabling and Configuring the Build Cache](http://docs.google.com/userguide/build_cache.html)
* [Integrating Separate Gradle Builds (Composite Builds)](http://docs.google.com/userguide/composite_builds.html)

### Authoring Gradle Builds

* [Fundamentals](#30j0zll)
  + [Introducing the Basics of Build Scripts](http://docs.google.com/userguide/tutorial_using_tasks.html)
  + [Working with Tasks](http://docs.google.com/userguide/more_about_tasks.html)
  + [Learning More About Build Scripts](http://docs.google.com/userguide/writing_build_scripts.html)
  + [Working with Files](http://docs.google.com/userguide/working_with_files.html)
  + [Creating Custom Task Types](http://docs.google.com/userguide/custom_tasks.html)
  + [Using Gradle Plugins](http://docs.google.com/userguide/plugins.html)
  + [The Standard Gradle Plugins](http://docs.google.com/userguide/standard_plugins.html)
  + [Understanding the Build Lifecycle](http://docs.google.com/userguide/build_lifecycle.html)
  + [Working with Logging](http://docs.google.com/userguide/logging.html)
  + [Configuring Multi-Project Builds](http://docs.google.com/userguide/multi_project_builds.html)
* [Best Practices](#1fob9te)
  + [Authoring Maintainable Build Scripts](http://docs.google.com/userguide/authoring_maintainable_build_scripts.html)
  + [Organizing Gradle Projects](http://docs.google.com/userguide/organizing_gradle_projects.html)
  + [Optimizing Build Performance](https://guides.gradle.org/performance/)
  + [Using the Build Cache](https://guides.gradle.org/using-build-cache/)
* [Dependency Management](#3znysh7)
  + [Introduction to Dependency Management](http://docs.google.com/userguide/introduction_dependency_management.html)
  + [Dependency Management Terminology](http://docs.google.com/userguide/dependency_management_terminology.html)
  + [Dependency Types](http://docs.google.com/userguide/dependency_types.html)
  + [Repository Types](http://docs.google.com/userguide/repository_types.html)
  + [Declaring Dependencies](http://docs.google.com/userguide/declaring_dependencies.html)
  + [Declaring Repositories](http://docs.google.com/userguide/declaring_repositories.html)
  + [Inspecting Dependencies](http://docs.google.com/userguide/inspecting_dependencies.html)
  + [Managing Dependency Configurations](http://docs.google.com/userguide/managing_dependency_configurations.html)
  + [Managing Transitive Dependencies](http://docs.google.com/userguide/managing_transitive_dependencies.html)
  + [Dependency Locking](http://docs.google.com/userguide/dependency_locking.html)
  + [Troubleshooting Dependency Resolution](http://docs.google.com/userguide/troubleshooting_dependency_resolution.html)
  + [Customizing Dependency Resolution Behavior](http://docs.google.com/userguide/customizing_dependency_resolution_behavior.html)
  + [Dependency Cache Internals](http://docs.google.com/userguide/dependency_cache.html)
  + [Working with Dependencies](http://docs.google.com/userguide/working_with_dependencies.html)
* [Publishing Artifacts](http://docs.google.com/userguide/artifact_management.html)
* [C++ Projects](#2et92p0)
  + [Building Native Software](http://docs.google.com/userguide/native_software.html)
  + [Software Model Concepts](http://docs.google.com/userguide/software_model_concepts.html)
  + [Rule-based Model Configuration](http://docs.google.com/userguide/software_model.html)
  + [Implementing Model Rules in a Plugin](http://docs.google.com/userguide/rule_source.html)
  + [Extending the Software Model](http://docs.google.com/userguide/software_model_extend.html)
* [Java Projects](#tyjcwt)
  + [Building Java & JVM projects](http://docs.google.com/userguide/building_java_projects.html)
  + [Testing Java & JVM projects](http://docs.google.com/userguide/java_testing.html)
* [Advanced Techniques](#3dy6vkm)
  + [Configuring Tasks Lazily](http://docs.google.com/userguide/lazy_configuration.html)
  + [Developing Parallel Tasks](https://guides.gradle.org/using-the-worker-api/)
  + [Testing Your Build with TestKit](http://docs.google.com/userguide/test_kit.html)
  + [Using Ant from Gradle](http://docs.google.com/userguide/ant.html)
* [Sample Gradle builds](#1t3h5sf)
  + [Groovy DSL Samples](https://github.com/gradle/gradle/tree/master/subprojects/docs/src/samples)
  + [Kotlin DSL Samples](https://github.com/gradle/kotlin-dsl/tree/master/samples)

### Extending Gradle

* [Writing Custom Plugins](http://docs.google.com/userguide/custom_plugins.html)
* [Plugin Development Guides](https://gradle.org/guides/?q=Plugin+Development)

[Edit this page](https://github.com/gradle/gradle/edit/master/subprojects/docs/src/docs/userguide/)

# Lazy Configuration

Contents

[Lazy properties](#4d34og8)

[Creating a Property or Provider](#2s8eyo1)

[Working with files and Providers](#17dp8vu)

[Working with task dependencies and Providers](#3rdcrjn)

[Working with collection Providers](#26in1rg)

[Guidelines](#lnxbz9)

[Future development](#35nkun2)

[Provider Files API Reference](#1ksv4uv)

[Property Files API Reference](#44sinio)

[Lazy Collections API Reference](#2jxsxqh)

[Lazy Objects API Reference](#z337ya)

As a build grows in complexity, knowing when and where a particular value is configured can become difficult to reason about. Gradle provides several ways to manage this complexity using *lazy configuration*.

[Lazy properties](#4d34og8)

| **✨** | The Provider API is currently [incubating](http://docs.google.com/feature_lifecycle.html#feature_lifecycle). Please be aware that the DSL and other configuration may change in later Gradle versions. |
| --- | --- |

Gradle provides lazy properties, which delay the calculation of a property’s value until it’s absolutely required. Lazy types are faster, more understandable and better instrumented than the internal convention mapping mechanisms. This provides two main benefits to build script and plugin authors:

1. Build authors can wire together Gradle models without worrying when a particular property’s value will be known. For example, when you want to map properties in an extension to task properties but the values aren’t known until the build script configures them.
2. Build authors can avoid resource intensive work during the configuration phase, which can have a direct impact on maximum build performance. For example, when a property value comes from parsing a file.

Gradle represents lazy properties with two interfaces:

* [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) are properties that can only be queried and cannot be changed.
  + Properties with these types are read-only.
  + The method [Provider.get()](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html#get--) returns the current value of the property.
  + A Provider can be created by the factory method [ProviderFactory.provider(java.util.concurrent.Callable)](http://docs.google.com/javadoc/org/gradle/api/provider/ProviderFactory.html#provider-java.util.concurrent.Callable-).
* [Property](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html) are properties that can be queried and overwritten.
  + Properties with these types are configurable.
  + Property implements the Provider interface.
  + The method [Property.set(T)](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html#set-T-) specifies a value for the property, overwriting whatever value may have been present.
  + The method [Property.set(org.gradle.api.provider.Provider)](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html#set-org.gradle.api.provider.Provider-) specifies a Provider for the value for the property, overwriting whatever value may have been present. This allows you to wire together Provider and Property instances before the values are configured.
  + A Property can be created by the factory method [ObjectFactory.property(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html#property-java.lang.Class-).

Neither of these types nor their subtypes are intended to be implemented by a build script or plugin author. Gradle provides several factory methods to create instances of these types. See the [Quick Reference](#1ksv4uv) for all of the types and factories available.

Lazy properties are intended to be passed around and only evaluated when required (usually, during the execution phase). For more information about the Gradle build phases, please see [Build Lifecycle](http://docs.google.com/build_lifecycle.html#sec:build_phases).

The following demonstrates a task with a read-only property and a configurable property:

[Example: Using a read-only and configurable property](#3j2qqm3)

**build.gradle**

class Greeting extends DefaultTask {  
 // Configurable by the user  
 @Input  
 final Property<String> message = project.objects.property(String)  
  
 // Read-only property calculated from the message  
 @Internal  
 final Provider<String> fullMessage = message.map { it + " from Gradle" }  
  
 @TaskAction  
 void printMessage() {  
 logger.quiet(fullMessage.get())  
 }  
}  
  
task greeting(type: Greeting) {  
 // Note that this is effectively calling Property.set()  
 message = 'Hi'  
}

**Output of** gradle greeting

> gradle greeting  
  
> Task :greeting  
Hi from Gradle  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

The Greeting task has a Property<String> for the mutable part of the message and a Provider<String> for the calculated, read-only, message.

| **✨** | Note that Groovy Gradle DSL will generate setter methods for each Property-typed property in a task implementation. These setter methods allow you to configure the property using the assignment (=) operator as a convenience. |
| --- | --- |

[Creating a Property or Provider](#2s8eyo1)

If provider types are not intended to be implemented directly by build script or plugin authors, how do you create a new one? Gradle provides various factory APIs to create new instances of both [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) and [Property](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html):

* [ProviderFactory.provider(java.util.concurrent.Callable)](http://docs.google.com/javadoc/org/gradle/api/provider/ProviderFactory.html#provider-java.util.concurrent.Callable-) instantiates a new Provider. An instance of the [ProviderFactory](http://docs.google.com/javadoc/org/gradle/api/provider/ProviderFactory.html) can be referenced from [Project.getProviders()](http://docs.google.com/javadoc/org/gradle/api/Project.html#getProviders--) or by injecting ProviderFactory through a constructor or method.
* [ObjectFactory.property(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html#property-java.lang.Class-) instantiates a new Property. An instance of the [ObjectFactory](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html) can be referenced from [Project.getObjects()](http://docs.google.com/javadoc/org/gradle/api/Project.html#getObjects--) or by injecting ObjectFactory through a constructor or method.

| **✨** | [Project](http://docs.google.com/dsl/org.gradle.api.Project.html) does not provide a specific method signature for creating a provider from a groovy.lang.Closure. When writing a plugin with Groovy, you can use the method signature accepting a java.util.concurrent.Callable parameter. Groovy’s [Closure to type coercion](http://docs.groovy-lang.org/next/html/documentation/core-semantics.html#_assigning_a_closure_to_a_sam_type) will take care of the rest. |
| --- | --- |

[Working with files and Providers](#17dp8vu)

In [Working with Files](http://docs.google.com/working_with_files.html#working_with_files), we introduced four collection types for File-like objects:

Table 1. Collection of files recap

| **Read-only Type** | **Configurable Type** |
| --- | --- |
| [FileCollection](http://docs.google.com/javadoc/org/gradle/api/file/FileCollection.html) | [ConfigurableFileCollection](http://docs.google.com/javadoc/org/gradle/api/file/ConfigurableFileCollection.html) |
| [FileTree](http://docs.google.com/javadoc/org/gradle/api/file/FileTree.html) | [ConfigurableFileTree](http://docs.google.com/javadoc/org/gradle/api/file/ConfigurableFileTree.html) |

All of these types are also considered Provider types.

In this section, we are going to introduce more strongly typed models for a [FileSystemLocation](http://docs.google.com/javadoc/org/gradle/api/file/FileSystemLocation.html): [Directory](http://docs.google.com/javadoc/org/gradle/api/file/Directory.html) and [RegularFile](http://docs.google.com/javadoc/org/gradle/api/file/RegularFile.html). These types shouldn’t be confused with the standard Java [java.io.File](https://docs.oracle.com/javase/7/docs/api/java/io/File.html) type as they tell Gradle to expect more specific values (a directory or a non-directory, regular file).

Gradle provides two specialized Property subtypes for dealing with these types: [RegularFileProperty](http://docs.google.com/javadoc/org/gradle/api/file/RegularFileProperty.html) and [DirectoryProperty](http://docs.google.com/javadoc/org/gradle/api/file/DirectoryProperty.html). [ProjectLayout](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html) has methods to create these: [ProjectLayout.fileProperty()](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#fileProperty--) and [ProjectLayout.directoryProperty()](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#directoryProperty--).

A DirectoryProperty can also be used to create a lazily evaluated Provider for a Directory and RegularFile via [DirectoryProperty.dir(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/DirectoryProperty.html#dir-java.lang.String-) and [DirectoryProperty.file(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/DirectoryProperty.html#file-java.lang.String-) respectively. These methods create paths that are relative to the location set for the original DirectoryProperty.

[Example: Using file and directory property](#1y810tw)

**build.gradle**

class FooExtension {  
 final DirectoryProperty someDirectory  
 final RegularFileProperty someFile  
 final ConfigurableFileCollection someFiles  
  
 FooExtension(Project project) {  
 someDirectory = project.layout.directoryProperty()  
 someFile = project.layout.fileProperty()  
 someFiles = project.layout.configurableFiles()  
 }  
}  
  
project.extensions.create('foo', FooExtension, project)  
  
foo {  
 someDirectory = project.layout.projectDirectory.dir('some-directory')  
 someFile = project.layout.buildDirectory.file('some-file')  
 someFiles.from project.layout.configurableFiles(someDirectory, someFile)  
}  
  
task print {  
 doLast {  
 def someDirectory = project.foo.someDirectory.get().asFile  
 logger.quiet("foo.someDirectory = " + someDirectory)  
 logger.quiet("foo.someFiles contains someDirectory? " + project.foo.someFiles.contains(someDirectory))  
  
 def someFile = project.foo.someFile.get().asFile  
 logger.quiet("foo.someFile = " + someFile)  
 logger.quiet("foo.someFiles contains someFile? " + project.foo.someFiles.contains(someFile))  
 }  
}

**Output of** gradle print

> gradle print  
  
> Task :print  
foo.someDirectory = /home/user/gradle/samples/some-directory  
foo.someFiles contains someDirectory? true  
foo.someFile = /home/user/gradle/samples/build/some-file  
foo.someFiles contains someFile? true  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

This example shows how Provider types can be used inside an extension. Lazy values for [Project.getBuildDir()](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:buildDir) and [Project.getProjectDir()](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:projectDir) can be accessed through [Project.getLayout()](http://docs.google.com/javadoc/org/gradle/api/Project.html#getLayout--) with [ProjectLayout.getBuildDirectory()](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#getBuildDirectory--) and [ProjectLayout.getProjectDirectory()](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#getProjectDirectory--).

[Working with task dependencies and Providers](#3rdcrjn)

Many builds have several tasks that depend on each other. This usually means that one task processes the outputs of another task as an input. For these outputs and inputs, we need to know their locations on the file system and appropriately configure each task to know where to look. This can be cumbersome if any of these values are configurable by a user or configured by multiple plugins.

To make this easier, Gradle offers convenient APIs for defining files or directories as task inputs and outputs in a descriptive way. As an example consider the following plugin with a producer and consumer task, which are wired together via inputs and outputs:

[Example: Implicit task dependency](#4i7ojhp)

**build.gradle**

class Producer extends DefaultTask {  
 @OutputFile  
 final RegularFileProperty outputFile = newOutputFile()  
  
 @TaskAction  
 void produce() {  
 String message = 'Hello, World!'  
 def output = outputFile.get().asFile  
 output.text = message  
 logger.quiet("Wrote '${message}' to ${output}")  
 }  
}  
  
class Consumer extends DefaultTask {  
 @InputFile  
 final RegularFileProperty inputFile = newInputFile()  
  
 @TaskAction  
 void consume() {  
 def input = inputFile.get().asFile  
 def message = input.text  
 logger.quiet("Read '${message}' from ${input}")  
 }  
}  
  
task producer(type: Producer)  
task consumer(type: Consumer)  
  
// Wire property from producer to consumer task  
consumer.inputFile = producer.outputFile  
  
// Set values for the producer lazily  
// Note that the consumer does not need to be changed again.  
producer.outputFile = layout.buildDirectory.file('file.txt')  
  
// Change the base output directory.  
// Note that this automatically changes producer.outputFile and consumer.inputFile  
buildDir = 'output'

**Output of** gradle consumer

> gradle consumer  
  
> Task :producer  
Wrote 'Hello, World!' to /home/user/gradle/samples/output/file.txt  
  
> Task :consumer  
Read 'Hello, World!' from /home/user/gradle/samples/output/file.txt  
  
BUILD SUCCESSFUL in 0s  
2 actionable tasks: 2 executed

In the example above, the task outputs and inputs are connected before any location is defined. This is possible because the input and output properties use the Provider API. The output property is created with [DefaultTask.newOutputFile()](http://docs.google.com/javadoc/org/gradle/api/DefaultTask.html#newOutputFile--) and the input property is created with [DefaultTask.newInputFile()](http://docs.google.com/javadoc/org/gradle/api/DefaultTask.html#newInputFile--). Values are only resolved when they are needed during execution. The setters can be called at any time before the task is executed and the change will automatically affect all related input and output properties.

Another thing to note is the absence of any explicit task dependency. Properties created via newOutputFile() and newOutputDirectory() bring knowledge about which task is generating them, so using them as task input will implicitly link tasks together.

[Working with collection Providers](#26in1rg)

In this section, we are going to explore lazy collections. They work exactly like any other Provider and, just like FileSystemLocation providers, they have additional modeling around them. There are two provider interfaces available, one for List values and another for Set values:

* For List values the interface is called [ListProperty](http://docs.google.com/javadoc/org/gradle/api/provider/ListProperty.html). You can create a new ListProperty using [ObjectFactory.listProperty(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html#listProperty-java.lang.Class-) and specifying the element’s type.
* For Set values the interface is called [SetProperty](http://docs.google.com/javadoc/org/gradle/api/provider/SetProperty.html). You can create a new SetProperty using [ObjectFactory.setProperty(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html#setProperty-java.lang.Class-) and specifying the element’s type.

This type of property allows you to overwrite the entire collection value with [HasMultipleValues.set(java.lang.Iterable)](http://docs.google.com/javadoc/org/gradle/api/provider/HasMultipleValues.html#set-java.lang.Iterable-) and [HasMultipleValues.set(org.gradle.api.provider.Provider)](http://docs.google.com/javadoc/org/gradle/api/provider/HasMultipleValues.html#set-org.gradle.api.provider.Provider-) or add new elements through the various add methods:

* [HasMultipleValues.add(T)](http://docs.google.com/javadoc/org/gradle/api/provider/HasMultipleValues.html#add-T-): Add a single concrete element to the collection
* [HasMultipleValues.add(org.gradle.api.provider.Provider)](http://docs.google.com/javadoc/org/gradle/api/provider/HasMultipleValues.html#add-org.gradle.api.provider.Provider-): Add a lazily evaluated element to the collection
* [HasMultipleValues.addAll(org.gradle.api.provider.Provider)](http://docs.google.com/javadoc/org/gradle/api/provider/HasMultipleValues.html#addAll-org.gradle.api.provider.Provider-): Add a lazily evaluated collection of elements to the list

Just like every Provider, the collection is calculated when [Provider.get()](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html#get--) is called. The following example show the [ListProperty](http://docs.google.com/javadoc/org/gradle/api/provider/ListProperty.html) in action:

[Example: List property](#2xcytpi)

**build.gradle**

task print {  
 doLast {  
 ListProperty<String> list = project.objects.listProperty(String)  
  
 // Resolve the list  
 logger.quiet('The list contains: ' + list.get())  
  
 // Add elements to the empty list  
 list.add(project.provider { 'element-1' }) // Add a provider element  
 list.add('element-2') // Add a concrete element  
  
 // Resolve the list  
 logger.quiet('The list contains: ' + list.get())  
  
 // Overwrite the entire list with a new list  
 list.set(['element-3', 'element-4'])  
  
 // Resolve the list  
 logger.quiet('The list contains: ' + list.get())  
  
 // Add more elements through a list provider  
 list.addAll(project.provider { ['element-5', 'element-6'] })  
  
 // Resolve the list  
 logger.quiet('The list contains: ' + list.get())  
 }  
}

**Output of** gradle print

> gradle print  
  
> Task :print  
The list contains: []  
The list contains: [element-1, element-2]  
The list contains: [element-3, element-4]  
The list contains: [element-3, element-4, element-5, element-6]  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

[Guidelines](#lnxbz9)

This section will introduce guidelines to be successful with the Provider API. To see those guidelines in action, have a look at [gradle-site-plugin](https://github.com/gradle-guides/gradle-site-plugin), a Gradle plugin demonstrating established techniques and practices for plugin development.

* The [Property](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html) and [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) types have all of the overloads you need to query or configure a value. For this reason, you should follow the following guidelines:
  + For configurable properties, expose the [Property](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html) directly through a single getter.
  + For non-configurable properties, expose an [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) directly through a single getter.
* Avoid simplifying calls like obj.getProperty().get() and obj.getProperty().set(T) in your code by introducing additional getters and setters.
* When migrating your plugin to use providers, follow these guidelines:
  + If it’s a new property, expose it as a [Property](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html) or [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) using a single getter.
  + If it’s incubating, change it to use a [Property](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html) or [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) using a single getter.
  + If it’s a stable property, add a new [Property](http://docs.google.com/javadoc/org/gradle/api/provider/Property.html) or [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) and deprecate the old one. You should wire the old getter/setters into the new property as appropriate.

[Future development](#35nkun2)

Going forward, new properties will use the Provider API. The Groovy Gradle DSL adds convenience methods to make the use of Providers mostly transparent in build scripts. Existing tasks will have their existing "raw" properties replaced by Providers as needed and in a backwards compatible way. New tasks will be designed with the Provider API.

The Provider API is [incubating](http://docs.google.com/feature_lifecycle.html#feature_lifecycle). Please create new issues at [gradle/gradle](https://github.com/gradle/gradle/issues/new) to report bugs or to submit use cases for new features.

[Provider Files API Reference](#1ksv4uv)

Use these types for *read-only* values:

[Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html)<[RegularFile](http://docs.google.com/javadoc/org/gradle/api/file/RegularFile.html)>

File on disk

Factories

* [ProjectLayout.fileProperty()](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#fileProperty--)
* [DirectoryProperty.file(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/DirectoryProperty.html#file-java.lang.String-)

[Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html)<[Directory](http://docs.google.com/javadoc/org/gradle/api/file/Directory.html)>

Directory on disk

Factories

* [ProjectLayout.directoryProperty()](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#directoryProperty--)
* [DirectoryProperty.dir(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/DirectoryProperty.html#dir-java.lang.String-)

[FileCollection](http://docs.google.com/javadoc/org/gradle/api/file/FileCollection.html)

Unstructured collection of files

Factories

* [Project.files(java.lang.Object[])](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:files(java.lang.Object%5B%5D))
* [ProjectLayout.files(java.lang.Object...)](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#files-java.lang.Object...-)

[FileTree](http://docs.google.com/javadoc/org/gradle/api/file/FileTree.html)

Hierarchy of files

Factories

* [Project.fileTree(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:fileTree(java.lang.Object)) will produce a [ConfigurableFileTree](http://docs.google.com/javadoc/org/gradle/api/file/ConfigurableFileTree.html), or you can use [Project.zipTree(Object)](http://docs.google.com/javadoc/org/gradle/api/Project.html#zipTree-java.lang.Object-) and [Project.tarTree(Object)](http://docs.google.com/javadoc/org/gradle/api/Project.html#tarTree-java.lang.Object-)

[Property Files API Reference](#44sinio)

Use these types for *mutable* values:

[RegularFileProperty](http://docs.google.com/javadoc/org/gradle/api/file/RegularFileProperty.html)

File on disk

Factories

* [DefaultTask.newInputFile()](http://docs.google.com/javadoc/org/gradle/api/DefaultTask.html#newInputFile--) and [DefaultTask.newOutputFile()](http://docs.google.com/javadoc/org/gradle/api/DefaultTask.html#newOutputFile--) if used as task input/output
* [Directory.file(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/Directory.html#file-java.lang.String-) otherwise

[DirectoryProperty](http://docs.google.com/javadoc/org/gradle/api/file/DirectoryProperty.html)

Directory on disk

Factories

* [DefaultTask.newInputDirectory()](http://docs.google.com/javadoc/org/gradle/api/DefaultTask.html#newInputDirectory--) and [DefaultTask.newOutputDirectory()](http://docs.google.com/javadoc/org/gradle/api/DefaultTask.html#newOutputDirectory--) if used as task input/output
* [Directory.dir(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/file/Directory.html#dir-java.lang.String-) otherwise

[ConfigurableFileCollection](http://docs.google.com/javadoc/org/gradle/api/file/ConfigurableFileCollection.html)

Unstructured collection of files

Factories

* [ProjectLayout.configurableFiles(java.lang.Object...)](http://docs.google.com/javadoc/org/gradle/api/file/ProjectLayout.html#configurableFiles-java.lang.Object...-)

[ConfigurableFileTree](http://docs.google.com/javadoc/org/gradle/api/file/ConfigurableFileTree.html)

Hierarchy of files

Factories

* [Project.fileTree(java.lang.Object)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:fileTree(java.lang.Object))

[Lazy Collections API Reference](#2jxsxqh)

* For lists, use [ObjectFactory.listProperty(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html#listProperty-java.lang.Class-) to get a [ListProperty](http://docs.google.com/javadoc/org/gradle/api/provider/ListProperty.html) which is also a Provider<List<T>>
* For sets, use [ObjectFactory.setProperty(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html#setProperty-java.lang.Class-) to get a [SetProperty](http://docs.google.com/javadoc/org/gradle/api/provider/SetProperty.html) which is a Provider<Set<T>>

[Lazy Objects API Reference](#z337ya)

Use [ObjectFactory.property(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/model/ObjectFactory.html#property-java.lang.Class-) to construct a Property<T> which is a Provider<T>.

Docs

* [User Manual](http://docs.google.com/userguide/userguide.html)
* [DSL Reference](http://docs.google.com/dsl/)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Javadoc](http://docs.google.com/javadoc/)

News

* [Blog](https://blog.gradle.org/)
* [Newsletter](https://newsletter.gradle.com/)
* [Twitter](https://twitter.com/gradle)

Products

* [Build Scans](https://gradle.com/build-scans)
* [Build Cache](https://gradle.com/build-cache)
* [Enterprise Docs](https://gradle.com/enterprise/resources)

Get Help

* [Forums](https://discuss.gradle.org/c/help-discuss)
* [GitHub](https://github.com/gradle/)
* [Training](https://gradle.org/training/)
* [Services](https://gradle.org/services/)

Subscribe for important Gradle updates and news

Subscribe

By entering your email, you agree to our [Terms](https://gradle.org/terms/) and [Privacy Policy](https://gradle.org/privacy/), including receipt of emails. You can unsubscribe at any time.

© [Gradle Inc.](https://gradle.com) 2018 All rights reserved.

[Careers](https://gradle.com/careers) | [Privacy](https://gradle.org/privacy) | [Terms of Service](https://gradle.org/terms) | [Contact](https://gradle.org/contact/)