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/)

# The Scala Plugin

Contents

[Usage](#4d34og8)

[Tasks](#2s8eyo1)

[Project layout](#17dp8vu)

[Dependency management](#3rdcrjn)

[Automatic configuration of scalaClasspath](#26in1rg)

[Configuring the Zinc compiler](#lnxbz9)

[Convention properties](#35nkun2)

[Source set properties](#1ksv4uv)

[Compiling in external process](#44sinio)

[Incremental compilation](#2jxsxqh)

[Compiling and testing for Java 6 or Java 7](#z337ya)

[Eclipse Integration](#3j2qqm3)

[IntelliJ IDEA Integration](#1y810tw)

The Scala plugin extends the Java plugin to add support for Scala projects. It can deal with Scala code, mixed Scala and Java code, and even pure Java code (although we don’t necessarily recommend to use it for the latter). The plugin supports *joint compilation*, which allows you to freely mix and match Scala and Java code, with dependencies in both directions. For example, a Scala class can extend a Java class that in turn extends a Scala class. This makes it possible to use the best language for the job, and to rewrite any class in the other language if needed.

[Usage](#4d34og8)

To use the Scala plugin, include the following in your build script:

[Example: Using the Scala plugin](#4i7ojhp)

**build.gradle**

apply plugin: 'scala'

[Tasks](#2s8eyo1)

The Scala plugin adds the following tasks to the project.

compileScala — [ScalaCompile](http://docs.google.com/dsl/org.gradle.api.tasks.scala.ScalaCompile.html)

*Depends on*: compileJava

Compiles production Scala source files.

compileTestScala — [ScalaCompile](http://docs.google.com/dsl/org.gradle.api.tasks.scala.ScalaCompile.html)

*Depends on*: compileTestJava

Compiles test Scala source files.

compile*SourceSet*Scala — [ScalaCompile](http://docs.google.com/dsl/org.gradle.api.tasks.scala.ScalaCompile.html)

*Depends on*: compile*SourceSet*Java

Compiles the given source set’s Scala source files.

scaladoc — [ScalaDoc](http://docs.google.com/dsl/org.gradle.api.tasks.scala.ScalaDoc.html)

Generates API documentation for the production Scala source files.

The Scala plugin adds the following dependencies to tasks added by the Java plugin.

Table 1. Scala plugin - additional task dependencies

| **Task name** | **Depends on** |
| --- | --- |
| classes | compileScala |
| testClasses | compileTestScala |
| *sourceSet*Classes | compile*SourceSet*Scala |



*Figure 1. Scala plugin - tasks*

[Project layout](#17dp8vu)

The Scala plugin assumes the project layout shown below. All the Scala source directories can contain Scala *and* Java code. The Java source directories may only contain Java source code. None of these directories need to exist or have anything in them; the Scala plugin will simply compile whatever it finds.

src/main/java

Production Java source.

src/main/resources

Production resources, such as XML and properties files.

src/main/scala

Production Scala source. May also contain Java source files for joint compilation.

src/test/java

Test Java source.

src/test/resources

Test resources.

src/test/scala

Test Scala source. May also contain Java source files for joint compilation.

src/*sourceSet*/java

Java source for the source set named *sourceSet*.

src/*sourceSet*/resources

Resources for the source set named *sourceSet*.

src/*sourceSet*/scala

Scala source files for the given source set. May also contain Java source files for joint compilation.

[Changing the project layout](#2xcytpi)

Just like the Java plugin, the Scala plugin allows you to configure custom locations for Scala production and test source files.

[Example: Custom Scala source layout](#1ci93xb)

**build.gradle**

sourceSets {  
 main {  
 scala {  
 srcDirs = ['src/scala']  
 }  
 }  
 test {  
 scala {  
 srcDirs = ['test/scala']  
 }  
 }  
}

[Dependency management](#3rdcrjn)

Scala projects need to declare a scala-library dependency. This dependency will then be used on compile and runtime class paths. It will also be used to get hold of the Scala compiler and Scaladoc tool, respectively.[[1](#3whwml4)]

If Scala is used for production code, the scala-library dependency should be added to the compile configuration:

[Example: Declaring a Scala dependency for production code](#2bn6wsx)

**build.gradle**

repositories {  
 mavenCentral()  
}  
  
dependencies {  
 compile 'org.scala-lang:scala-library:2.11.12'  
 testCompile 'org.scalatest:scalatest\_2.11:3.0.0'  
 testCompile 'junit:junit:4.12'  
}

If Scala is only used for test code, the scala-library dependency should be added to the testCompile configuration:

[Example: Declaring a Scala dependency for test code](#qsh70q)

**build.gradle**

dependencies {  
 testCompile "org.scala-lang:scala-library:2.11.1"  
}

[Automatic configuration of scalaClasspath](#26in1rg)

The ScalaCompile and ScalaDoc tasks consume Scala code in two ways: on their classpath, and on their scalaClasspath. The former is used to locate classes referenced by the source code, and will typically contain scala-library along with other libraries. The latter is used to load and execute the Scala compiler and Scaladoc tool, respectively, and should only contain the scala-compiler library and its dependencies.

Unless a task’s scalaClasspath is configured explicitly, the Scala (base) plugin will try to infer it from the task’s classpath. This is done as follows:

* If a scala-library jar is found on classpath, and the project has at least one repository declared, a corresponding scala-compiler repository dependency will be added to scalaClasspath.
* Otherwise, execution of the task will fail with a message saying that scalaClasspath could not be inferred.

[Configuring the Zinc compiler](#lnxbz9)

The Scala plugin uses a configuration named zinc to resolve the [Zinc compiler](https://github.com/typesafehub/zinc) and its dependencies. Gradle will provide a default version of Zinc, but if you need to use a particular Zinc version, you can add an explicit dependency like “com.typesafe.zinc:zinc:0.3.6” to the zinc configuration. Gradle supports version 0.3.0 of Zinc and above; however, due to a regression in the Zinc compiler, versions 0.3.2 through 0.3.5.2 cannot be used.

[Example: Declaring a version of the Zinc compiler to use](#3as4poj)

**build.gradle**

dependencies {  
 zinc 'com.typesafe.zinc:zinc:0.3.9'  
}

It is important to take care when declaring your scala-library dependency. The Zinc compiler itself needs a compatible version of scala-library that may be different from the version required by your application. Gradle takes care of adding a compatible version of scala-library for you, but over-broad dependency resolution rules could force an incompatible version to be used instead.

For example, using configurations.all to force a particular version of scala-library would also override the version used by the Zinc compiler:

[Example: Forcing a scala-library dependency for all configurations](#1pxezwc)

**build.gradle**

configurations.all {  
 resolutionStrategy.force "org.scala-lang:scala-library:2.11.12"  
}

The best way to avoid this problem is to be more selective when configuring the scala-library dependency (such as not using a configuration.all rule or using a conditional to prevent the rule from being applied to the zinc configuration). Sometimes this rule may come from a plugin or other code that you do not have control over. In such a case, you can force a correct version of the library on the zinc configuration only:

[Example: Forcing a scala-library dependency for the zinc configuration](#49x2ik5)

**build.gradle**

configurations.zinc {  
 resolutionStrategy.force "org.scala-lang:scala-library:2.10.5"  
}

You can diagnose problems with the version of the Zinc compiler selected by running [dependencyInsight](http://docs.google.com/inspecting_dependencies.html#inspecting_dependencies) for the zinc configuration.

[Convention properties](#35nkun2)

The Scala plugin does not add any convention properties to the project.

[Source set properties](#1ksv4uv)

The Scala plugin adds the following convention properties to each source set in the project. You can use these properties in your build script as though they were properties of the source set object.

scala — [SourceDirectorySet](http://docs.google.com/dsl/org.gradle.api.file.SourceDirectorySet.html) (read-only)

The Scala source files of this source set. Contains all .scala and .java files found in the Scala source directories, and excludes all other types of files. *Default value:* non-null.

scala.srcDirs — Set<File>

The source directories containing the Scala source files of this source set. May also contain Java source files for joint compilation. Can set using anything described in [Understanding implicit conversion to file collections](http://docs.google.com/working_with_files.html#sec:specifying_multiple_files). *Default value:* [*projectDir*/src/*name*/scala].

allScala — [FileTree](http://docs.google.com/javadoc/org/gradle/api/file/FileTree.html) (read-only)

All Scala source files of this source set. Contains only the .scala files found in the Scala source directories. *Default value:* non-null.

These convention properties are provided by a convention object of type [ScalaSourceSet](http://docs.google.com/dsl/org.gradle.api.tasks.ScalaSourceSet.html).

The Scala plugin also modifies some source set properties:

Table 2. Scala plugin - source set properties

| **Property name** | **Change** |
| --- | --- |
| allJava | Adds all .java files found in the Scala source directories. |
| allSource | Adds all source files found in the Scala source directories. |

[Compiling in external process](#44sinio)

Scala compilation takes place in an external process.

Memory settings for the external process default to the defaults of the JVM. To adjust memory settings, configure the scalaCompileOptions.forkOptions property as needed:

[Example: Adjusting memory settings](#2p2csry)

**build.gradle**

tasks.withType(ScalaCompile) {  
 configure(scalaCompileOptions.forkOptions) {  
 memoryMaximumSize = '1g'  
 jvmArgs = ['-XX:MaxPermSize=512m']  
 }  
}

[Incremental compilation](#2jxsxqh)

By compiling only classes whose source code has changed since the previous compilation, and classes affected by these changes, incremental compilation can significantly reduce Scala compilation time. It is particularly effective when frequently compiling small code increments, as is often done at development time.

The Scala plugin defaults to incremental compilation by integrating with [Zinc](https://github.com/typesafehub/zinc), a standalone version of [sbt](https://github.com/harrah/xsbt)'s incremental Scala compiler. If you want to disable the incremental compilation, set force = true in your build file:

[Example: Forcing all code to be compiled](#147n2zr)

**build.gradle**

tasks.withType(ScalaCompile) {  
 scalaCompileOptions.with {  
 force = true  
 }  
}

*Note:* This will only cause all classes to be recompiled if at least one input source file has changed. If there are no changes to the source files, the compileScala task will still be considered UP-TO-DATE as usual.

The Zinc-based Scala Compiler supports joint compilation of Java and Scala code. By default, all Java and Scala code under src/main/scala will participate in joint compilation. Even Java code will be compiled incrementally.

Incremental compilation requires dependency analysis of the source code. The results of this analysis are stored in the file designated by scalaCompileOptions.incrementalOptions.analysisFile (which has a sensible default). In a multi-project build, analysis files are passed on to downstream ScalaCompile tasks to enable incremental compilation across project boundaries. For ScalaCompile tasks added by the Scala plugin, no configuration is necessary to make this work. For other ScalaCompile tasks that you might add, the property scalaCompileOptions.incrementalOptions.publishedCode needs to be configured to point to the classes folder or Jar archive by which the code is passed on to compile class paths of downstream ScalaCompile tasks. Note that if publishedCode is not set correctly, downstream tasks may not recompile code affected by upstream changes, leading to incorrect compilation results.

Note that Zinc’s Nailgun based daemon mode is not supported. Instead, we plan to enhance Gradle’s own compiler daemon to stay alive across Gradle invocations, reusing the same Scala compiler. This is expected to yield another significant speedup for Scala compilation.

[Compiling and testing for Java 6 or Java 7](#z337ya)

The Scala compiler ignores Gradle’s targetCompatibility and sourceCompatibility settings. In Scala 2.11, the Scala compiler always compiles to Java 6 compatible bytecode. In Scala 2.12, the Scala compiler always compiles to Java 8 compatible bytecode. If you also have Java source, you can follow the same steps as for the [Java plugin](http://docs.google.com/building_java_projects.html#sec:java_cross_compilation) to ensure the correct Java compiler is used.

[Example: Configure Java 6 build for Scala](#3o7alnk)

**gradle.properties**

# in $HOME/.gradle/gradle.properties  
java6Home=/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home

**build.gradle**

sourceCompatibility = 1.6  
  
assert hasProperty('java6Home') : "Set the property 'java6Home' in your your gradle.properties pointing to a Java 6 installation"  
def javaExecutablesPath = new File(java6Home, 'bin')  
def javaExecutables = [:].withDefault { execName ->  
 def executable = new File(javaExecutablesPath, execName)  
 assert executable.exists() : "There is no ${execName} executable in ${javaExecutablesPath}"  
 executable  
}  
  
tasks.withType(AbstractCompile) {  
 options.with {  
 fork = true  
 forkOptions.javaHome = file(java6Home)  
 }  
}  
tasks.withType(Test) {  
 executable = javaExecutables.java  
}  
tasks.withType(JavaExec) {  
 executable = javaExecutables.java  
}  
tasks.withType(Javadoc) {  
 executable = javaExecutables.javadoc  
}

[Eclipse Integration](#3j2qqm3)

When the Eclipse plugin encounters a Scala project, it adds additional configuration to make the project work with Scala IDE out of the box. Specifically, the plugin adds a Scala nature and dependency container.

[IntelliJ IDEA Integration](#1y810tw)

When the IDEA plugin encounters a Scala project, it adds additional configuration to make the project work with IDEA out of the box. Specifically, the plugin adds a Scala SDK (IntelliJ IDEA 14+) and a Scala compiler library that matches the Scala version on the project’s class path. The Scala plugin is backwards compatible with earlier versions of IntelliJ IDEA and it is possible to add a Scala facet instead of the default Scala SDK by configuring targetVersion on [IdeaModel](http://docs.google.com/dsl/org.gradle.plugins.ide.idea.model.IdeaModel.html).

[Example: Explicitly specify a target IntelliJ IDEA version](#23ckvvd)

**build.gradle**

idea {  
 targetVersion = "13"  
}

[1](#ihv636). See [Automatic configuration of Scala classpath](#26in1rg).

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/)