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/)

# Building Play applications

Contents

[Usage](#4d34og8)

[Limitations](#2s8eyo1)

[Software Model](#17dp8vu)

[Project Layout](#3rdcrjn)

[Tasks](#26in1rg)

[Finding out more about your project](#lnxbz9)

[Running a Play application](#35nkun2)

[Configuring a Play application](#1ksv4uv)

[Multi-project Play applications](#44sinio)

[Packaging a Play application for distribution](#2jxsxqh)

[Building a Play application with an IDE](#z337ya)

[Resources](#3j2qqm3)

| **✨** | Support for building Play applications is currently [incubating](http://docs.google.com/feature_lifecycle.html#feature_lifecycle). Please be aware that the DSL, APIs and other configuration may change in later Gradle versions. |
| --- | --- |

[Play](https://www.playframework.com/) is a modern web application framework. The Play plugin adds support for building, testing and running Play applications with Gradle.

The Play plugin makes use of the Gradle [software model](http://docs.google.com/software_model.html#software_model).

[Usage](#4d34og8)

To use the Play plugin, include the following in your build script to apply the play plugin and add the Lightbend repositories:

[Example: Using the Play plugin](#1y810tw)

**build.gradle**

plugins {  
 id 'play'  
}  
  
repositories {  
 jcenter()  
 maven {  
 name "lightbend-maven-release"  
 url "https://repo.lightbend.com/lightbend/maven-releases"  
 }  
 ivy {  
 name "lightbend-ivy-release"  
 url "https://repo.lightbend.com/lightbend/ivy-releases"  
 layout "ivy"  
 }  
}

Note that defining the Lightbend repositories is necessary. In future versions of Gradle, this will be replaced with a more convenient syntax.

[Limitations](#2s8eyo1)

The Play plugin currently has a few limitations.

* Gradle does not yet support aggregate reverse routes introduced in Play 2.4.x.
* A given project may only define a single Play application. This means that a single project cannot build more than one Play application. However, a multi-project build can have many projects that each define their own Play application.
* Play applications can only target a single “platform” (combination of Play, Scala and Java version) at a time. This means that it is currently not possible to define multiple variants of a Play application that, for example, produce jars for both Scala 2.10 and 2.11. This limitation may be lifted in future Gradle versions.
* Support for generating IDE configurations for Play applications is limited to [IDEA](http://docs.google.com/idea_plugin.html#idea_plugin).

[Software Model](#17dp8vu)

The Play plugin uses a *software model* to describe a Play application and how to build it. The Play software model extends the base Gradle [software model](http://docs.google.com/software_model_concepts.html#software_model_concepts) to add support for building Play applications. A Play application is represented by a [PlayApplicationSpec](http://docs.google.com/javadoc/org/gradle/play/PlayApplicationSpec.html) component type. The plugin automatically creates a single [PlayApplicationBinarySpec](http://docs.google.com/javadoc/org/gradle/play/PlayApplicationBinarySpec.html) instance when it is applied. Additional Play components cannot be added to a project.



*Figure 1. Play plugin - software model*

[The Play application component](#4i7ojhp)

A Play application component describes the application to be built and consists of several configuration elements. One type of element that describes the application are the source sets that define where the application controller, route, template and model class source files should be found. These source sets are logical groupings of files of a particular type and a default source set for each type is created when the play plugin is applied.

Table 1. Default Play source sets

| **Source Set** | **Type** | **Directory** | **Filters** |
| --- | --- | --- | --- |
| java | [JavaSourceSet](http://docs.google.com/javadoc/org/gradle/language/java/JavaSourceSet.html) | app | \*\*/\*.java |
| scala | [ScalaLanguageSourceSet](http://docs.google.com/javadoc/org/gradle/language/scala/ScalaLanguageSourceSet.html) | app | \*\*/\*.scala |
| routes | [RoutesSourceSet](http://docs.google.com/javadoc/org/gradle/language/routes/RoutesSourceSet.html) | conf | routes, \*.routes |
| twirlTemplates | [TwirlSourceSet](http://docs.google.com/javadoc/org/gradle/language/twirl/TwirlSourceSet.html) | app | \*\*/\*.scala.\* |
| javaScript | [JavaScriptSourceSet](http://docs.google.com/javadoc/org/gradle/language/javascript/JavaScriptSourceSet.html) | app/assets | \*\*/\*.js |

These source sets can be configured or additional source sets can be added to the Play component. See [Configuring Play](#1ksv4uv) for further information.

Another element of configuring a Play application is the *platform*. To build a Play application, Gradle needs to understand which versions of Play, Scala and Java to use. The Play component specifies this requirement as a [PlayPlatform](http://docs.google.com/javadoc/org/gradle/play/platform/PlayPlatform.html). If these values are not configured, a default version of Play, Scala and Java will be used. See [Targeting a certain version of Play](#2xcytpi) for information on configuring the Play platform.

Note that only a single platform can be specified for a given Play component. This means that only a single version of Play, Scala and Java can be used to build a Play component. In other words, a Play component can only produce one set of outputs, and those outputs will be built using the versions specified by the platform configured on the component.

[The Play application binary](#1ci93xb)

A Play application component is compiled and packaged to produce a set of outputs which are represented by a [PlayApplicationBinarySpec](http://docs.google.com/javadoc/org/gradle/play/PlayApplicationBinarySpec.html). The Play binary specifies the jar files produced by building the component as well as providing elements by which additional content can be added to those jar files. It also exposes the tasks involved in building the component and creating the binary.

See [Configuring Play](#1ksv4uv) for examples of configuring the Play binary.

[Project Layout](#3rdcrjn)

The Play plugin follows the typical Play application layout. You can [configure source sets](#3whwml4) to include additional directories or change the defaults.

├── app → Application source code.  
│   ├── assets → Assets that require compilation.  
│   │   └── javascripts → JavaScript source code to be minified.  
│   ├── controllers → Application controller source code.  
│   ├── models → Application business source code.  
│   └── views → Application UI templates.  
├── build.gradle → Your project's build script.  
├── conf → Main application configuration file and routes files.  
├── public → Public assets.  
│   ├── images → Application image files.  
│   ├── javascripts → Typically JavaScript source code.  
│   └── stylesheets → Typically CSS source code.  
└── test → Test source code.

[Tasks](#26in1rg)

The Play plugin hooks into the normal Gradle lifecycle tasks such as assemble, check and build, but it also adds several additional tasks which form the lifecycle of a Play project:

[Play Plugin — lifecycle tasks](#2bn6wsx)

playBinary — [Task](http://docs.google.com/dsl/org.gradle.api.Task.html)

*Depends on*: All compile tasks for source sets added to the Play application.

Performs a build of just the Play application.

dist — [Task](http://docs.google.com/dsl/org.gradle.api.Task.html)

*Depends on*: createPlayBinaryZipDist, createPlayBinaryTarDist

Assembles the Play distribution.

stage — [Task](http://docs.google.com/dsl/org.gradle.api.Task.html)

*Depends on*: stagePlayBinaryDist

Stages the Play distribution.

The plugin also provides tasks for running, testing and packaging your Play application:

[Play Plugin — running and testing tasks](#qsh70q)

runPlayBinary — [PlayRun](http://docs.google.com/dsl/org.gradle.play.tasks.PlayRun.html)

*Depends on*: playBinary to build Play application.

Runs the Play application for local development. See [how this works with continuous build.](#35nkun2)

testPlayBinary — [Test](http://docs.google.com/dsl/org.gradle.api.tasks.testing.Test.html)

*Depends on*: playBinary to build Play application and compilePlayBinaryTests.

Runs JUnit/TestNG tests for the Play application.

For the different types of sources in a Play application, the plugin adds the following compilation tasks:

[Play Plugin — source set tasks](#3as4poj)

compilePlayBinaryScala — [PlatformScalaCompile](http://docs.google.com/javadoc/org/gradle/language/scala/tasks/PlatformScalaCompile.html)

*Depends on*: Scala and Java

Compiles all Scala and Java sources defined by the Play application.

compilePlayBinaryPlayTwirlTemplates — [TwirlCompile](http://docs.google.com/dsl/org.gradle.play.tasks.TwirlCompile.html)

*Depends on*: Twirl templates

Compiles Twirl templates with the Twirl compiler. Gradle supports all of the built-in Twirl template formats (HTML, XML, TXT and JavaScript). Twirl templates need to match the pattern \*.scala.\*.

compilePlayBinaryPlayRoutes — [RoutesCompile](http://docs.google.com/dsl/org.gradle.play.tasks.RoutesCompile.html)

*Depends on*: Play Route files

Compiles routes files into Scala sources.

minifyPlayBinaryJavaScript — [JavaScriptMinify](http://docs.google.com/dsl/org.gradle.play.tasks.JavaScriptMinify.html)

*Depends on*: JavaScript files

Minifies JavaScript files with the Google Closure compiler.

[Finding out more about your project](#lnxbz9)

Gradle provides a report that you can run from the command-line that shows some details about the components and binaries that your project produces. To use this report, just run gradle components. Below is an example of running this report for one of the sample projects:

[Example: The components report](#1pxezwc)

**Output of** gradle components

> gradle components  
  
> Task :components  
  
------------------------------------------------------------  
Root project  
------------------------------------------------------------  
  
Play Application 'play'  
-----------------------  
  
Source sets  
 Java source 'play:java'  
 srcDir: app  
 includes: \*\*/\*.java  
 JavaScript source 'play:javaScript'  
 srcDir: app/assets  
 includes: \*\*/\*.js  
 JVM resources 'play:resources'  
 srcDir: conf  
 Routes source 'play:routes'  
 srcDir: conf  
 includes: routes, \*.routes  
 Scala source 'play:scala'  
 srcDir: app  
 includes: \*\*/\*.scala  
 Twirl template source 'play:twirlTemplates'  
 srcDir: app  
 includes: \*\*/\*.scala.\*  
  
Binaries  
 Play Application Jar 'play:binary'  
 build using task: :playBinary  
 target platform: Play Platform (Play 2.6.15, Scala: 2.12, Java: Java SE 8)  
 toolchain: Default Play Toolchain  
 classes dir: build/playBinary/classes  
 resources dir: build/playBinary/resources  
 JAR file: build/playBinary/lib/basic.jar  
  
Note: currently not all plugins register their components, so some components may not be visible here.  
  
BUILD SUCCESSFUL in 0s  
1 actionable task: 1 executed

[Running a Play application](#35nkun2)

The runPlayBinary task starts the Play application under development. During development it is beneficial to execute this task as a [continuous build](http://docs.google.com/command_line_interface.html#sec:continuous_build). Continuous build is a generic feature that supports automatically re-running a build when inputs change. The runPlayBinary task is “continuous build aware” in that it behaves differently when run as part of a continuous build.

When not run as part of a continuous build, the runPlayBinary task will *block* the build. That is, the task will not complete as long as the application is running. When running as part of a continuous build, the task will start the application if not running and otherwise propagate any changes to the code of the application to the running instance. This is useful for quickly iterating on your Play application with an edit->rebuild->refresh cycle. Changes to your application will not take affect until the end of the overall build.

To enable continuous build, run Gradle with -t runPlayBinary or --continuous runPlayBinary.

Users of Play used to such a workflow with Play’s default build system should note that compile errors are handled differently. If a build failure occurs during a continuous build, the Play application will not be reloaded. Instead, you will be presented with an exception message. The exception message will only contain the overall cause of the build failure. More detailed information will only be available from the console.

[Configuring a Play application](#1ksv4uv)

[Targeting a certain version of Play](#2xcytpi)

By default, Gradle uses Play 2.6.15, Scala 2.12 and the version of Java used to start the build. A Play application can select a different version by specifying a target [PlayApplicationSpec.platform(java.lang.Object)](http://docs.google.com/dsl/org.gradle.play.PlayApplicationSpec.html#org.gradle.play.PlayApplicationSpec:platform(java.lang.Object)) on the Play application component.

[Example: Selecting a version of the Play Framework](#49x2ik5)

**build.gradle**

model {  
 components {  
 play {  
 platform play: '2.6.15', scala: '2.12', java: '1.8'  
 injectedRoutesGenerator = true  
 }  
 }  
}

The following versions of Play and Scala are supported:

Table 2. Play supported versions

| **Play** | **Scala** | **Java** |
| --- | --- | --- |
| 2.6.x | 2.11 and 2.12 | 1.8 |
| 2.5.x | 2.11 | 1.8 |
| 2.4.x | 2.10 and 2.11 | 1.8 |
| 2.3.x | 2.10 and 2.11 | 1.6, 1.7 and 1.8 |

[Adding dependencies](#2p2csry)

You can add compile, test and runtime dependencies to a Play application through [Configuration](http://docs.google.com/dsl/org.gradle.api.artifacts.Configuration.html) created by the Play plugin.

If you are coming from SBT, the Play SBT plugin provides short names for common dependencies. For instance, if your project has a dependency on ws, you will need to add a dependency to com.typesafe.play:play-ws\_2.11:2.3.9 where 2.11 is your Scala version and 2.3.9 is your Play framework version.

Other dependencies that have short names, such as jacksons may actually be multiple dependencies. For those dependencies, you will need to work out the dependency coordinates from a dependency report.

* play is used for compile time dependencies.
* playTest is used for test compile time dependencies.
* playRun is used for run time dependencies.

[Example: Adding dependencies to a Play application](#147n2zr)

**build.gradle**

dependencies {  
 play "commons-lang:commons-lang:2.6"  
 play "com.typesafe.play:play-guice\_2.12:2.6.15"  
 play "ch.qos.logback:logback-classic:1.2.3"  
}

Play 2.6 has a more modular architecture and, because of that, you may need to add some dependencies manually. For example, [Guice support was moved to a separated module](https://playframework.com/documentation/2.6.x/Migration26#Guice-DI-support-moved-to-separate-module). Considering the following definition for a Play 2.6 project:

[Example: A Play 2.6 project](#3o7alnk)

**build.gradle**

model {  
 components {  
 play {  
 platform play: '2.6.7', scala: '2.12', java: '1.8'  
 injectedRoutesGenerator = true  
 }  
 }  
}

You can add Guice dependency like:

[Example: Adding Guice dependency in Play 2.6 project](#23ckvvd)

**build.gradle**

dependencies {  
 play "com.typesafe.play:play-guice\_2.12:2.6.7"  
}

Of course, pay attention to keep the Play version and Scala version for the dependency consistent with the platform versions.

[Configuring the default source sets](#3whwml4)

You can further configure the default source sets to do things like add new directories, add filters, etc.

[Example: Configuring extra source sets to a Play application](#ihv636)

**build.gradle**

model {  
 components {  
 play {  
 sources {  
 java {  
 source.srcDir "additional/java"  
 }  
 javaScript {  
 source {  
 srcDir "additional/javascript"  
 exclude "\*\*/old\_\*.js"  
 }  
 }  
 }  
 }  
 }  
}

[Adding extra source sets](#32hioqz)

If your Play application has additional sources that exist in non-standard directories, you can add extra source sets that Gradle will automatically add to the appropriate compile tasks.

[Example: Adding extra source sets to a Play application](#1hmsyys)

**build.gradle**

model {  
 components {  
 play {  
 sources {  
 extraJava(JavaSourceSet) {  
 source.srcDir "extra/java"  
 }  
 extraTwirl(TwirlSourceSet) {  
 source.srcDir "extra/twirl"  
 }  
 extraRoutes(RoutesSourceSet) {  
 source.srcDir "extra/routes"  
 }  
 }  
 }  
 }  
}

[Configuring compiler options](#41mghml)

If your Play application requires additional Scala compiler flags, you can add these arguments directly to the Scala compiler task.

[Example: Configuring Scala compiler options](#2grqrue)

**build.gradle**

model {  
 components {  
 play {  
 binaries.all {  
 tasks.withType(PlatformScalaCompile) {  
 scalaCompileOptions.additionalParameters = ["-feature", "-language:implicitConversions"]  
 }  
 }  
 }  
 }  
}

[Configuring routes style](#vx1227)

| **✨** | The injected router is only supported in Play Framework 2.4 or better. |
| --- | --- |

If your Play application’s router uses dependency injection to access your controllers, you’ll need to configure your application to *not* use the default static router. Under the covers, the Play plugin is using the InjectedRoutesGenerator instead of the default StaticRoutesGenerator to generate the router classes.

[Example: Configuring routes style](#3fwokq0)

**build.gradle**

model {  
 components {  
 play {  
 injectedRoutesGenerator = true  
 }  
 }  
}

[Configuring Twirl templates](#1v1yuxt)

A custom Twirl template format can be configured independently for each Twirl source set. See the [TwirlSourceSet](http://docs.google.com/javadoc/org/gradle/language/twirl/TwirlSourceSet.html) for an example.

[Injecting a custom asset pipeline](#4f1mdlm)

Gradle Play support comes with a simplistic asset processing pipeline that minifies JavaScript assets. However, many organizations have their own custom pipeline for processing assets. You can easily hook the results of your pipeline into the Play binary by utilizing the [PublicAssets](http://docs.google.com/javadoc/org/gradle/play/PublicAssets.html) property on the binary.

[Example: Configuring a custom asset pipeline](#2u6wntf)

**build.gradle**

model {  
 components {  
 play {  
 binaries.all { binary ->  
 tasks.create("addCopyrightToPlay${binary.name.capitalize()}Assets", AddCopyrights) { copyrightTask ->  
 source "raw-assets"  
 copyrightFile = project.file('copyright.txt')  
 destinationDir = project.file("${buildDir}/play${binary.name.capitalize()}/addCopyRights")  
  
 // Hook this task into the binary  
 binary.assets.addAssetDir destinationDir  
 binary.assets.builtBy copyrightTask  
 }  
 }  
 }  
 }  
}  
  
class AddCopyrights extends SourceTask {  
 @InputFile  
 File copyrightFile  
  
 @OutputDirectory  
 File destinationDir  
  
 @TaskAction  
 void generateAssets() {  
 String copyright = copyrightFile.text  
 getSource().each { File file ->  
 File outputFile = new File(destinationDir, file.name)  
 outputFile.text = "${copyright}\n${file.text}"  
 }  
 }  
}

[Multi-project Play applications](#44sinio)

Play applications can be built in multi-project builds as well. Simply apply the play plugin in the appropriate subprojects and create any project dependencies on the play configuration.

[Example: Configuring dependencies on Play subprojects](#19c6y18)

**build.gradle**

dependencies {  
 play project(":admin")  
 play project(":user")  
 play project(":util")  
}

See the play/multiproject sample provided in the Gradle distribution for a working example.

[Packaging a Play application for distribution](#2jxsxqh)

Gradle provides the capability to package your Play application so that it can easily be distributed and run in a target environment. The distribution package (zip file) contains the Play binary jars, all dependencies, and generated scripts that set up the classpath and run the application in a Play-specific [Netty](http://netty.io) container.

The distribution can be created by running the dist lifecycle task and places the distribution in the $buildDir/distributions directory. Alternatively, one can validate the contents by running the stage lifecycle task which copies the files to the $buildDir/stage directory using the layout of the distribution package.

[Play Plugin — distribution tasks](#3tbugp1)

createPlayBinaryStartScripts — [CreateStartScripts](http://docs.google.com/javadoc/org/gradle/api/tasks/application/CreateStartScripts.html)

Generates scripts to run the Play application distribution.

stagePlayBinaryDist — [Copy](http://docs.google.com/dsl/org.gradle.api.tasks.Copy.html)

*Depends on*: playBinary, createPlayBinaryStartScripts

Copies all jar files, dependencies and scripts into a staging directory.

createPlayBinaryZipDist — [Zip](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.Zip.html)

Bundles the Play application as a standalone distribution packaged as a zip.

createPlayBinaryTarDist — [Tar](http://docs.google.com/dsl/org.gradle.api.tasks.bundling.Tar.html)

Bundles the Play application as a standalone distribution packaged as a tar.

stage — [Task](http://docs.google.com/dsl/org.gradle.api.Task.html)

*Depends on*: stagePlayBinaryDist

Lifecycle task for staging a Play distribution.

dist — [Task](http://docs.google.com/dsl/org.gradle.api.Task.html)

*Depends on*: createPlayBinaryZipDist, createPlayBinaryTarDist

Lifecycle task for creating a Play distribution.

[Adding additional files to your Play application distribution](#28h4qwu)

You can add additional files to the distribution package using the [Distribution](http://docs.google.com/javadoc/org/gradle/api/distribution/Distribution.html) API.

[Example: Add extra files to a Play application distribution](#nmf14n)

**build.gradle**

model {  
 distributions {  
 playBinary {  
 contents {  
 from("README.md")  
 from("scripts") {  
 into "bin"  
 }  
 }  
 }  
 }  
}

[Building a Play application with an IDE](#z337ya)

If you want to generate IDE metadata configuration for your Play project, you need to apply the appropriate IDE plugin. Gradle supports generating IDE metadata for IDEA only for Play projects at this time.

To generate IDEA’s metadata, apply the idea plugin along with the play plugin.

[Example: Applying both the Play and IDEA plugins](#37m2jsg)

**build.gradle**

plugins {  
 id 'play'  
 id 'idea'  
}

Source code generated by routes and Twirl templates cannot be generated by IDEA directly, so changes made to those files will not affect compilation until the next Gradle build. You can run the Play application with Gradle in [continuous build](http://docs.google.com/command_line_interface.html#sec:continuous_build) to automatically rebuild and reload the application whenever something changes.

[Resources](#3j2qqm3)

For additional information about developing Play applications:

* Play types in the Gradle DSL Guide:
  + [PlayApplicationBinarySpec](http://docs.google.com/javadoc/org/gradle/play/PlayApplicationBinarySpec.html)
  + [PlayApplicationSpec](http://docs.google.com/javadoc/org/gradle/play/PlayApplicationSpec.html)
  + [PlayPlatform](http://docs.google.com/javadoc/org/gradle/play/platform/PlayPlatform.html)
  + [JvmClasses](http://docs.google.com/javadoc/org/gradle/play/JvmClasses.html)
  + [PublicAssets](http://docs.google.com/javadoc/org/gradle/play/PublicAssets.html)
  + [PlayDistributionContainer](http://docs.google.com/javadoc/org/gradle/play/distribution/PlayDistributionContainer.html)
  + [JavaScriptMinify](http://docs.google.com/dsl/org.gradle.play.tasks.JavaScriptMinify.html)
  + [PlayRun](http://docs.google.com/dsl/org.gradle.play.tasks.PlayRun.html)
  + [RoutesCompile](http://docs.google.com/dsl/org.gradle.play.tasks.RoutesCompile.html)
  + [TwirlCompile](http://docs.google.com/dsl/org.gradle.play.tasks.TwirlCompile.html)
* [Play Framework Documentation](https://www.playframework.com/documentation).

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/)