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/)

# Dependency Management for Java Projects

Contents

[Dissecting a typical build script](#4d34og8)

[Declaring module dependencies](#2s8eyo1)

[Using dependency configurations](#17dp8vu)

[Declaring common Java repositories](#3rdcrjn)

[Publishing artifacts](#26in1rg)

This chapter explains how to apply basic dependency management concepts to Java-based projects. For a detailed introduction to dependency management, see [Introduction to Dependency Management](http://docs.google.com/introduction_dependency_management.html#introduction_dependency_management).

[Dissecting a typical build script](#4d34og8)

Let’s have a look at a very simple build script for a Java-based project. It applies the [Java Library plugin](http://docs.google.com/java_library_plugin.html#java_library_plugin) which automatically introduces a standard project layout, provides tasks for performing typical work and adequate support for dependency management.

[Example: Dependency declarations for a Java-based project](#lnxbz9)

**build.gradle**

apply plugin: 'java-library'  
  
repositories {  
 mavenCentral()  
}  
  
dependencies {  
 implementation 'org.hibernate:hibernate-core:3.6.7.Final'  
 api 'com.google.guava:guava:23.0'  
 testImplementation 'junit:junit:4.+'  
}

The [Project.dependencies{}](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:dependencies(groovy.lang.Closure)) code block declares that Hibernate core 3.6.7.Final is required to compile the project’s production source code. It also states that junit >= 4.0 is required to compile the project’s tests. All dependencies are supposed to be looked up in the Maven Central repository as defined by [Project.repositories{}](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:repositories(groovy.lang.Closure)). The following sections explain each aspect in more detail.

[Declaring module dependencies](#2s8eyo1)

There are various [types of dependencies](http://docs.google.com/dependency_types.html#dependency_types) that you can declare. One such type is a *module dependency*. A [module dependency](http://docs.google.com/declaring_dependencies.html#sec:declaring_dependency_to_module) represents a dependency on a module with a specific version built outside the current build. Modules are usually stored in a repository, such as Maven Central, a corporate Maven or Ivy repository, or a directory in the local file system.

To define an module dependency, you add it to a [dependency configuration](#17dp8vu):

[Example: Definition of a module dependency](#35nkun2)

**build.gradle**

dependencies {  
 implementation 'org.hibernate:hibernate-core:3.6.7.Final'  
}

To find out more about defining dependencies, have a look at [Declaring Dependencies](http://docs.google.com/declaring_dependencies.html#declaring_dependencies).

[Using dependency configurations](#17dp8vu)

A [Configuration](http://docs.google.com/dsl/org.gradle.api.artifacts.Configuration.html) is a named set of dependencies and artifacts. There are three main purposes for a *configuration*:

Declaring dependencies

A plugin uses configurations to make it easy for build authors to declare what other subprojects or external artifacts are needed for various purposes during the execution of tasks defined by the plugin. For example a plugin may need the Spring web framework dependency to compile the source code.

Resolving dependencies

A plugin uses configurations to find (and possibly download) inputs to the tasks it defines. For example Gradle needs to download Spring web framework JAR files from Maven Central.

Exposing artifacts for consumption

A plugin uses configurations to define what *artifacts* it generates for other projects to consume. For example the project would like to publish its compiled source code packaged in the JAR file to an in-house Artifactory repository.

With those three purposes in mind, let’s take a look at a few of the [standard configurations defined by the Java Library Plugin](http://docs.google.com/java_library_plugin.html#sec:java_library_configurations_graph).

implementation

The dependencies required to compile the production source of the project which *are not* part of the API exposed by the project. For example the project uses Hibernate for its internal persistence layer implementation.

api

The dependencies required to compile the production source of the project which *are* part of the API exposed by the project. For example the project uses Guava and exposes public interfaces with Guava classes in their method signatures.

testImplementation

The dependencies required to compile and run the test source of the project. For example the project decided to write test code with the test framework JUnit.

Various plugins add further standard configurations. You can also define your own custom configurations in your build via [Project.configurations{}](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:configurations(groovy.lang.Closure)). See [Managing Dependency Configurations](http://docs.google.com/managing_dependency_configurations.html#managing_dependency_configurations) for the details of defining and customizing dependency configurations.

[Declaring common Java repositories](#3rdcrjn)

How does Gradle know where to find the files for external dependencies? Gradle looks for them in a *repository*. A repository is a collection of modules, organized by group, name and version. Gradle understands different [repository types](http://docs.google.com/repository_types.html#repository_types), such as Maven and Ivy, and supports various ways of accessing the repository via HTTP or other protocols.

By default, Gradle does not define any repositories. You need to define at least one with the help of [Project.repositories{}](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:repositories(groovy.lang.Closure)) before you can use module dependencies. One option is use the Maven Central repository:

[Example: Usage of Maven central repository](#1ksv4uv)

**build.gradle**

repositories {  
 mavenCentral()  
}

You can also have repositories on the local file system. This works for both Maven and Ivy repositories.

[Example: Usage of a local Ivy directory](#44sinio)

**build.gradle**

repositories {  
 ivy {  
 // URL can refer to a local directory  
 url "../local-repo"  
 }  
}

A project can have multiple repositories. Gradle will look for a dependency in each repository in the order they are specified, stopping at the first repository that contains the requested module.

To find out more about defining repositories, have a look at [Declaring Repositories](http://docs.google.com/declaring_repositories.html#declaring_repositories).

[Publishing artifacts](#26in1rg)

Dependency configurations are also used to publish files. Gradle calls these files *publication artifacts*, or usually just *artifacts*. As a user you will need to tell Gradle where to publish the artifacts. You do this by declaring repositories for the uploadArchives task. Here’s an example of publishing to a Maven repository:

[Example: Publishing to a Maven repository](#2jxsxqh)

**build.gradle**

apply plugin: 'maven'  
  
uploadArchives {  
 repositories {  
 mavenDeployer {  
 repository(url: "file://localhost/tmp/myRepo/")  
 }  
 }  
}

Now, when you run gradle uploadArchives, Gradle will build the JAR file, generate a .pom file and upload the artifacts.

To learn more about publishing artifacts, have a look at [Legacy Publishing](http://docs.google.com/artifact_management.html#artifact_management).

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/)