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 Eclipse Plugins

Contents

[Usage](#4d34og8)

[Tasks](#2s8eyo1)

[Configuration](#17dp8vu)

[Customizing the generated files](#3rdcrjn)

The Eclipse plugins generate files that are used by the [Eclipse IDE](http://eclipse.org), thus making it possible to import the project into Eclipse (File - Import…​ - Existing Projects into Workspace).

The eclipse-wtp is automatically applied whenever the eclipse plugin is applied to a [War](http://docs.google.com/war_plugin.html#war_plugin) or [Ear](http://docs.google.com/ear_plugin.html#ear_plugin) project. For utility projects (i.e. [Java](http://docs.google.com/java_plugin.html#java_plugin) projects used by other web projects), you need to apply the eclipse-wtp plugin explicitly.

What exactly the eclipse plugin generates depends on which other plugins are used:

Table 1. Eclipse plugin behavior

| **Plugin** | **Description** |
| --- | --- |
| None | Generates minimal .project file. |
| [Java](http://docs.google.com/java_plugin.html#java_plugin) | Adds Java configuration to .project. Generates .classpath and JDT settings file. |
| [Groovy](http://docs.google.com/groovy_plugin.html#groovy_plugin) | Adds Groovy configuration to .project file. |
| [Scala](http://docs.google.com/scala_plugin.html#scala_plugin) | Adds Scala support to .project and .classpath files. |
| [War](http://docs.google.com/war_plugin.html#war_plugin) | Adds web application support to .project file. |
| [Ear](http://docs.google.com/ear_plugin.html#ear_plugin) | Adds ear application support to .project file. |

The eclipse-wtp plugin generates all WTP settings files and enhances the .project file. If a [Java](http://docs.google.com/java_plugin.html#java_plugin) or [War](http://docs.google.com/war_plugin.html#war_plugin) is applied, .classpath will be extended to get a proper packaging structure for this utility library or web application project.

Both Eclipse plugins are open to customization and provide a standardized set of hooks for adding and removing content from the generated files.

[Usage](#4d34og8)

To use either the Eclipse or the Eclipse WTP plugin, include one of the lines in your build script:

[Example: Using the Eclipse plugin](#26in1rg)

**build.gradle**

apply plugin: 'eclipse'

[Example: Using the Eclipse WTP plugin](#lnxbz9)

**build.gradle**

apply plugin: 'eclipse-wtp'

*Note:* Internally, the eclipse-wtp plugin also applies the eclipse plugin so you don’t need to apply both.

Both Eclipse plugins add a number of tasks to your projects. The main tasks that you will use are the eclipse and cleanEclipse tasks.

[Tasks](#2s8eyo1)

The Eclipse plugins add the tasks shown below to a project.

[Eclipse Plugin tasks](#35nkun2)

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

*Depends on*: all Eclipse configuration file generation tasks

Generates all Eclipse configuration files

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

*Depends on*: all Eclipse configuration file clean tasks

Removes all Eclipse configuration files

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

Removes the .project file.

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

Removes the .classpath file.

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

Removes the .settings/org.eclipse.jdt.core.prefs file.

eclipseProject — [GenerateEclipseProject](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.GenerateEclipseProject.html)

Generates the .project file.

eclipseClasspath — [GenerateEclipseClasspath](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.GenerateEclipseClasspath.html)

Generates the .classpath file.

eclipseJdt — [GenerateEclipseJdt](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.GenerateEclipseJdt.html)

Generates the .settings/org.eclipse.jdt.core.prefs file.

[Eclipse WTP Plugin — additional tasks](#1ksv4uv)

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

Removes the .settings/org.eclipse.wst.common.component file.

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

Removes the .settings/org.eclipse.wst.common.project.facet.core.xml file.

eclipseWtpComponent — [GenerateEclipseWtpComponent](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.GenerateEclipseWtpComponent.html)

Generates the .settings/org.eclipse.wst.common.component file.

eclipseWtpFacet — [GenerateEclipseWtpFacet](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.GenerateEclipseWtpFacet.html)

Generates the .settings/org.eclipse.wst.common.project.facet.core.xml file.

[Configuration](#17dp8vu)

Table 2. Configuration of the Eclipse plugins

| **Model** | **Reference name** | **Description** |
| --- | --- | --- |
| [EclipseModel](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseModel.html) | eclipse | Top level element that enables configuration of the Eclipse plugin in a DSL-friendly fashion. |
| [EclipseProject](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseProject.html) | eclipse.project | Allows configuring project information |
| [EclipseClasspath](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseClasspath.html) | eclipse.classpath | Allows configuring classpath information. |
| [EclipseJdt](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseJdt.html) | eclipse.jdt | Allows configuring jdt information (source/target Java compatibility). |
| [EclipseWtpComponent](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseWtpComponent.html) | eclipse.wtp.component | Allows configuring wtp component information only if eclipse-wtp plugin was applied. |
| [EclipseWtpFacet](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseWtpFacet.html) | eclipse.wtp.facet | Allows configuring wtp facet information only if eclipse-wtp plugin was applied. |

[Customizing the generated files](#3rdcrjn)

The Eclipse plugins allow you to customize the generated metadata files. The plugins provide a DSL for configuring model objects that model the Eclipse view of the project. These model objects are then merged with the existing Eclipse XML metadata to ultimately generate new metadata. The model objects provide lower level hooks for working with domain objects representing the file content before and after merging with the model configuration. They also provide a very low level hook for working directly with the raw XML for adjustment before it is persisted, for fine tuning and configuration that the Eclipse and Eclipse WTP plugins do not model.

[Merging](#44sinio)

Sections of existing Eclipse files that are also the target of generated content will be amended or overwritten, depending on the particular section. The remaining sections will be left as-is.

[Disabling merging with a complete rewrite](#2jxsxqh)

To completely rewrite existing Eclipse files, execute a clean task together with its corresponding generation task, like “gradle cleanEclipse eclipse” (in that order). If you want to make this the default behavior, add “tasks.eclipse.dependsOn(cleanEclipse)” to your build script. This makes it unnecessary to execute the clean task explicitly.

This strategy can also be used for individual files that the plugins would generate. For instance, this can be done for the “.classpath” file with “gradle cleanEclipseClasspath eclipseClasspath”.

[Hooking into the generation lifecycle](#z337ya)

The Eclipse plugins provide objects modeling the sections of the Eclipse files that are generated by Gradle. The generation lifecycle is as follows:

1. The file is read; or a default version provided by Gradle is used if it does not exist
2. The beforeMerged hook is executed with a domain object representing the existing file
3. The existing content is merged with the configuration inferred from the Gradle build or defined explicitly in the eclipse DSL
4. The whenMerged hook is executed with a domain object representing contents of the file to be persisted
5. The withXml hook is executed with a raw representation of the XML that will be persisted
6. The final XML is persisted

[Advanced configuration hooks](#3j2qqm3)

The following list covers the domain object used for each of the Eclipse model types:

[EclipseProject](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseProject.html)

* beforeMerged { [Project](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/Project.html) arg -> …​ }
* whenMerged { [Project](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/Project.html) arg -> …​ }
* withXml { [XmlProvider](http://docs.google.com/javadoc/org/gradle/api/XmlProvider.html) arg -> …​ }

[EclipseClasspath](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseClasspath.html)

* beforeMerged { [Classpath](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/Classpath.html) arg -> …​ }
* whenMerged { [Classpath](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/Classpath.html) arg -> …​ }
* withXml { [XmlProvider](http://docs.google.com/javadoc/org/gradle/api/XmlProvider.html) arg -> …​ }

[EclipseWtpComponent](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseWtpComponent.html)

* beforeMerged { [WtpComponent](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/WtpComponent.html) arg -> …​ }
* whenMerged { [WtpComponent](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/WtpComponent.html) arg -> …​ }
* withXml { [XmlProvider](http://docs.google.com/javadoc/org/gradle/api/XmlProvider.html) arg -> …​ }

[EclipseWtpFacet](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseWtpFacet.html)

* beforeMerged { [WtpFacet](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/WtpFacet.html) arg -> …​ }
* whenMerged { [WtpFacet](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/WtpFacet.html) arg -> …​ }
* withXml { [XmlProvider](http://docs.google.com/javadoc/org/gradle/api/XmlProvider.html) arg -> …​ }

[EclipseJdt](http://docs.google.com/dsl/org.gradle.plugins.ide.eclipse.model.EclipseJdt.html)

* beforeMerged { [Jdt](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/Jdt.html) arg -> …​ }
* whenMerged { [Jdt](http://docs.google.com/javadoc/org/gradle/plugins/ide/eclipse/model/Jdt.html) arg -> …​ }
* withProperties { arg -> } argument type ⇒ [java.util.Properties](http://docs.oracle.com/javase/7/docs/api/java/util/Properties.html)

[Partial overwrite of existing content](#1y810tw)

A complete overwrite causes all existing content to be discarded, thereby losing any changes made directly in the IDE. Alternatively, the beforeMerged hook makes it possible to overwrite just certain parts of the existing content. The following example removes all existing dependencies from the Classpath domain object:

[Example: Partial Overwrite for Classpath](#4i7ojhp)

**build.gradle**

eclipse.classpath.file {  
 beforeMerged { classpath ->  
 classpath.entries.removeAll { entry -> entry.kind == 'lib' || entry.kind == 'var' }  
 }  
}

The resulting .classpath file will only contain Gradle-generated dependency entries, but not any other dependency entries that may have been present in the original file. (In the case of dependency entries, this is also the default behavior.) Other sections of the .classpath file will be either left as-is or merged. The same could be done for the natures in the .project file:

[Example: Partial Overwrite for Project](#2xcytpi)

**build.gradle**

eclipse.project.file.beforeMerged { project ->  
 project.natures.clear()  
}

[Modifying the fully populated domain objects](#1ci93xb)

The whenMerged hook allows to manipulate the fully populated domain objects. Often this is the preferred way to customize Eclipse files. Here is how you would export all the dependencies of an Eclipse project:

[Example: Export Classpath Entries](#3whwml4)

**build.gradle**

eclipse.classpath.file {  
 whenMerged { classpath ->  
 classpath.entries.findAll { entry -> entry.kind == 'lib' }\*.exported = false  
 }  
}

[Modifying the XML representation](#2bn6wsx)

The withXml hook allows to manipulate the in-memory XML representation just before the file gets written to disk. Although Groovy’s XML support makes up for a lot, this approach is less convenient than manipulating the domain objects. In return, you get total control over the generated file, including sections not modeled by the domain objects.

[Example: Customizing the XML](#qsh70q)

**build.gradle**

apply plugin: 'eclipse-wtp'  
  
eclipse.wtp.facet.file.withXml { provider ->  
 provider.asNode().fixed.find { it.@facet == 'jst.java' }.@facet = 'jst2.java'  
}

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/)