

General Build Concepts

This article talks about general build concepts used across WPILib. This does not go into specifics for how to build individual projects, as that is covered the by README in the project root.

Gradle Based Projects

With the exception of the SmartDashboard 2.0 projects, all WPILibSuite projects use Gradle to build. This means that, for the most part, you do not need to install anything on your local machine other than the FRC Toolchains (for projects that build for ARM) and Maven (for the Eclipse Plugins project only!). The Gradle wrapper script (gradlew.bat for Windows, gradlew for Mac and Linux) takes care of downloading and running the correct version of Gradle for your system. These scripts are located in the root of the Gradle-based projects. Gradle is generally run from the console, with the following format:

```
./gradlew task1 task2 ...
```

task1 and task2 should be replaced with the names of the actual tasks that you want to run. Gradle accepts an arbitrary number of tasks, separated by spaces. On the command line, these tasks are not case sensitive. For all Gradle projects, there are 4 main tasks that you will be using:

- **build** - executes the build process. Gradle is generally smart enough to only rebuild the sections that have changed since the last build.
- **clean** - removes all created build targets. The next build will fully rebuild from scratch.
- **publish** - publishes all Maven targets to your local FRC maven repository. The repository's function is talked about in more detail in the next section. This task depends on parts of the build task, and will ensure that all relevant libraries have been built before publishing.
- **tasks** - a meta-task that lists all available tasks, along with their descriptions.

Local FRC Maven Repo

Because FRC projects are hosted in separate repositories, they push dependencies to each other in a Maven repository. This works in a tiered fashion:

1. dependences are resolved in the local FRC repository on your computer and if not available there then
2. dependencies are resolved against the WPI/FIRST Maven server.

Local Maven Repository

General Build Concepts

This repository is located in `~/releases/maven/<build channel>` (where `~` refers to your home directory). Here, `<build channel>` is one of `development`, `beta`, `stable`, or `release`. These correspond to the 4 different stages of the WPILib release process. This default channel is `development`, which gets the latest changes that have been merged into the master branch of the project in question. To put artifacts in your local repository, run the `publish` task in the specific project. This also defaults to putting things in `~/releases/maven/development`.

WPI/FIRST Maven Repository

For dependencies that can't be resolved on your machine (because you haven't published that particular project), the build system retrieves them from the FIRST Maven Repository. The specifics of what artifacts are available and the URL of the repository is covered in the [Maven Artifacts](#) article. The repository also has `development`, `beta`, `stable`, and `release` channels, and it uses the same channel as was used for local resolution.

What all of this means is that in order for your changes in a project to appear in a project that depends on it, you must publish the changes to your local maven repo. For example, the `eclipse plugins` depends on artifacts published by the `allwpilib` project. If you make some changes in `allwpilib` and would like to test them in the `eclipse plugins`, simply run the `publish` task in `allwpilib`. Then, in the `eclipse plugins` project, run the traditional build step, `./gradlew build`, which will generate the plugins. This will prioritize the local maven repo over the official FRC server, pulling in your changed `allwpilib` builds and the latests development version of the rest of the utilities. If you want to go back to using the latest official development versions from the FRC server, you must delete the specific artifacts from your local repository.