# Setting up Unit Tests

The Unit Test runs a unit testing framework that produces a JUnit XML file with the results of the tests at a location stored in the `$CG_JUNIT_XML_LOCATION` environment variable.

The resulting score of this test is the fraction of successful tests divided by the total number of tests that were run, multiplied by the **weight** of the test.

1. Press the "**✚ Unit Test**" button to add a new Unit Test test to your Test Category.
2. Enter the command that will run your test suite. The command should write a JUnit XML file to the location given in the `$CG_JUNIT_XML_LOCATION` environment variable.

## Compatibility scripts

For most unit testing frameworks it is necessary to install additional software and to figure out how to make the framework output its results to the correct location. We provide wrapper scripts for a growing list of frameworks that handle all of this for you. Check out the Unit Test documentation for a list of frameworks that are already supported, or contact us at support@codegrade.com if your preferred framework is not listed there so we can discuss the possibilities!

All wrapper scripts share a similar interface and procedure for running tests, which we describe below.

### Installing dependencies

If extra software needs to be installed to run the testing framework the wrapper script provides an `install` command that will install all required dependencies. This should be run in the "Global setup script" section of your AutoTest configuration.

For example, to install JUnit 4 and its dependencies you would run

```
cg-junit4 install
```

## Compiling code and tests

For compiled languages it can be tricky to get the compiler to find all libraries and other dependencies needed to compile the students' code and your tests. For these languages the wrapper scripts provide a `compile` command that configures the compiler to be able to find the required libraries. The `compile` command takes file names as arguments and compiles those files.

For example, to compile all `.java` files in the current directory, including a JUnit version 4 test class, you would run

```
cg-junit4 compile *.java
```

## Running tests

Running the tests can be similarly tricky when a language needs to know about certain locations of libraries it depends on, but also because you need to figure out how you can make the framework output its results in the correct format, or in the correct location. The wrapper scripts handle this in their `run` command.

The scripts extract the output location from `$CG_JUNIT_XML_LOCATION` and then unset it before running the tests. The testing framework is then configured to output its results at that location and the tests are executed.

For example, to run a JUnit 4 test class named `MyTestClass` you would run

```
cg-junit4 run MyTestClass
```

**Note**

The exact arguments to the `run` command of each script may differ between scripts.