# Contributing to NHibernate

This document describes the policies and procedures for working with NHibernate. It also describes how to quickly get going with a test case and optionally a fix for NHibernate.

For the least friction, please follow the steps in the order presented, being careful not to miss any. There are many details in this document that will help your contribution go as smoothly as possible. Please read it thoroughly.

## Create or find a GitHub Issue

GitHub is used to generate the release notes and serves as a central point of reference for all changes that have occurred to NHibernate.

Visit [https://github.com/nhibernate/nhibernate-core/issues][1] and search for your issue. If you see it, giving it a like is a good way to increase the visibility of the issue.

Before creating an issue, please do your best to verify the existence of the problem. This reduces noise in the issue tracker and helps conserve the resources of the team for more useful tasks. Note the issue number for future steps.

If you are familiar with NHibernate inner working and want to directly fix the issue, you may also contribute a PR without writing a separated issue, provided you describe the fixed issue in the PR.

## Fork and Clone from GitHub

The main GitHub repository is at [https://github.com/nhibernate/nhibernate-core][2]. If you plan to contribute your test case or improvement back to NHibernate, you should visit that page and fork the repository so you can commit your own changes and then submit a pull request.

## The Build Menu

\*\*ShowBuildMenu.bat\*\* will be your friend throughout this journey. A .sh version exists for Linux. It's easiest to work with if you make his box bigger by following these steps:

1. Run ShowBuildMenu.bat.

2. Right click on the title bar of the window.

3. Select "Properties".

4. Select the "Layout" tab.

5. Set the following options.

\* Screen Buffer Size

\* Width: 160

\* Height: 9999 Window Size

\* Width: 160

\* Height: 50

## Setting up For Development

1. Install your favorite database and optionally set up a user capable of creating and dropping a database called \*\*nhibernate\*\*. There are some per-database instructions in the lib/teamcity/\* folders, which you may find helpful. For SQL Server, you might be able to use the \*\*localhost\sqlexpress\*\* instance installed with Visual Studio. Often people already have databases set up, so you might not even need to do anything in this step.

2. Run the build menu and select option A to create a new test configuration. Notepad will pop up and you should edit the connection string information, saving it when you're done. These configurations will appear in the "available-test-configurations" folder.

3. Run the build menu and select option B to activate the test configuration you created. The appropriate configuration will be copied to the "current-test-configuration" folder.

4. (Optional) Run all the tests with option C or D and hopefully you will see no failing tests. The build may fail on certain databases; please ask on the mailing list if you are having trouble.

5. Before using the database for unit tests from an IDE, you'll need to create an empty database that matches your connection string. The unit test in NHibernate.TestDatabaseSetup supports some databases. If the one you have configured is supported, it will drop the database if it does already exist, then recreate it.

Compiling the solution under Linux requires installation of the [Mono package mono-complete](https://www.mono-project.com/download/stable) and of the [.NET Core SDK](https://www.microsoft.com/net/download).

## Creating a Test Case to Verify the Issue

In most cases, you will be adding your test to the NHibernate.Test project. If there is a test that only works with VisualBasic, then add it to the NHibernate.Test.VisualBasic project instead.

1. Open \*\*NHibernate.sln\*\* from the src folder.

2. If adding a VisualBasic test, go to the Issues folder in the NHibernate.Test.VisualBasic project. If adding a C# test, go to the NHibernate.Test project.

3. C# only: either find the suitable feature folder and add your test there without following the two next points, or go to the NHSpecificTest folder.

4. Copy and paste the GH0000 folder to create a duplicate test ("Copy of GH0000").

5. Replace the four instances of GH0000 with your issue number.

6. Edit the test as you see fit. Don't commit yet; there are details in a later step.

Do not use anymore the NHxxxx naming, they match issue numbers from https://nhibernate.jira.com/

NHibernate has migrated its issue tracking from Jira to GitHub, and using the Jira naming may lead to conflicts with previous Jira issues.

## Running Your Unit Test

### Debugging Through Included NUnit GUI

1. Right click on the project (ex. NHibernate.Test) in Visual Studio.

2. Select: Debug -> Start New Instance

3. Type the name of your unit test to quickly go to it. For example: GH2318

4. Select and run the test.

5. You can also make the test project your startup project and it will run NUnit when you press F5.

### Third Party NUnit Test Runner

This could be something like ReSharper.

1. Sometimes tests fail when run under x64. If required (ex. SQLite and Firebird), go to the project properties Build tab and set the target to x86.

2. Next, just run the tests as usual.

## Regenerate async code

NHibernate uses a code generator for its async implementation and tests. If your changes, including tests, involve any synchronous method having an async

counter-part, you should regenerate the async code. Use build-menu option H for this. Then test any async counter-part it may have generated from your tests.

## Commit your Test Case

Ensure that your e-mail address and name are configured appropriately in Git.

Create a feature branch so it's easy to keep it separate from other improvements. Having a pull request accepted might involve further commits based on community feedback, so having the feature branch provides a tidy place to work from. Using the issue number as the branch name is good practice.

## Implementing the Bug Fix or Improvement

Since you now have a failing test case, it should be straight-forward to step into NHibernate to attempt to ascertain what the problem is. While this may seem daunting at first, feel free to give it a go. It's just code after all. :)

### Ensure All Tests Pass

Once you've made changes to the NHibernate code base, you'll want to ensure that you haven't caused any previously passing tests to fail. The easiest way to check this is to select option C or D from the build menu, ensure the root tree node is selected, then press run to have all the tests run.

Please note that some tests assume a case insensitive accent sensitive database when performing string comparison. Some tests assume SQL user locales to be en-US. They will fail otherwise. With SQL Server, collation with supplementary characters (\\_SC suffix on collation name) are no supported by legacy tests on "text" types.

## Submit a Pull Request

If you are fixing an existing issue, please make sure to include this issue number in your GitHub pull request. (Use the # prefix instead of GH for automatically having a link to it.)

We use tabs for code indentation, not spaces. To make this easier, NHibernate has an [editorconfig][3] configuration file to switch Visual Studio automatically between tabs and spaces mode.

After submitting your pull request, come back later to check the outcome of automated builds. If some have failed, they will be listed in your pull request with a link to the corresponding TeamCity build. Find out in the build which tests are newly failing, and take appropriate action. Some of those builds may have known failing tests, which does not trigger a build failure. In this case a \*Comparison.txt\* file in build Artifacts may help finding which failing tests are not known failing tests and must be addressed.

## Further Discussion

The NHibernate team monitors GitHub regularly, so your request will be noticed. If you want to discuss it further, you are welcome to post to the [nhibernate-development mailing list][4].

## Happy Contributing!

The NHibernate community values your contributions. Thank you for the time you have invested.

[1]: https://github.com/nhibernate/nhibernate-core/issues/

[2]: https://github.com/nhibernate/nhibernate-core/

[3]: http://www.editorconfig.org/

[4]: http://groups.google.com/group/nhibernate-development