---

uid: contributing

---

# Contributing guide

## Development rules and regulations, code style

Follow this [document](https://github.com/linq2db/linq2db/files/1056002/Development.rules.and.regulations.docx)

## Project structure

#### Solution and folder structure

| Folder | Description |

|--------------------------- |----------------------------------------------------------------------------------------------------------------------------------|

|.\Build | Build and CI files, check readme.md in that folder |

|.\Data | Databases and database creation scripts for tests |

|.\NuGet | LINQ to DB NuGet packages build files |

|.\Redist | Binaries,unavailable officially at NuGet, used by tests and nugets |

|.\Source\LinqToDB | LINQ to DB source code |

|.\Source\LinqToDB.Tools | LINQ to DB Tools source code |

|.\Source\LinqToDB.Templates | LINQ to DB t4models source code |

|.\Tests | Unit test projects folder |

|.\Tests\Base | LINQ to DB testing framework |

|.\Tests\FSharp | F# models and tests |

|.\Tests\Linq | Main project for LINQ to DB unit tests |

|.\Tests\Model | Model classes for tests |

|.\Tests\Tests.T4 | T4 templates test project |

|.\Tests\Tests.Android | Xamarin Forms for Android test project |

|.\Tests\Tests.Benchmark | Benchmarks |

|.\Tests\Tests.PLayground | Test project for use with linq2db.playground.sln lite test solution<br>Used for work on specific test without full solution load |

|.\Tests\VisualBasic | Visual Basic models and tests support |

Solutions:

\* `.\linq2db.sln` - full linq2db VS2019 solution

\* `.\linq2db.playground.sln` - ligthweight linq2db VS2019 test solution. Used to work on specific test without loading of all payload of full solution

#### Source projects

| Project \ Target |.NET 4.5 |.NET 4.6 | .NET Standard 2.0 | .NET Core 2.1 | .NET Standard 2.1 | .NET Core 3.1 |

|-------------------------------------------------:|:-------:|:-------:|:-----------------:|:-------------:|:-----------------:|:-------------:|

| `.\Source\LinqToDB\LinqToDB.csproj` | ? | ? | ? | ? | ? | ? |

| `.\Source\LinqToDB\LinqToDB.Tools.csproj` | ? | | ? | | | |

Preferred target defines:

- `NETFRAMEWORK` - `net45` and `net46` target ifdef

- `!NETFRAMEWORK` - `netstandard2.0` and newer target ifdef

- `NETCOREAPP` - `netcoreapp2.1` and `netcoreapp3.1` target ifdef

- `NETSTANDARD2\_1PLUS` - `netstandard2.1` and `netcoreapp3.1` target ifdef

Other allowed target defines:

- `NETSTANDARD2\_1` - `netstandard2.1` target ifdef

- `NETCOREAPP3\_1` - `netcoreapp3.1` target ifdef

- `NETSTANDARD2\_0` - `netstandard2.0` target ifdef

- `NETCOREAPP2\_1` - `netcoreapp2.1` target ifdef

- `NET45` - `net45` target ifdef

- `NET46` - `net46` target ifdef

Allowed debugging defines:

- `TRACK\_BUILD`

- `DEBUG` - for debug code in debug build. To disable debug code use `DEBUG1` rename

- `OVERRIDETOSTRING` - enables ToString()` overrides for AST model (must be enabled in LinqToDB.csproj by renaming existing `OVERRIDETOSTRING1` define)

#### Test projects

| Project \ Target |.NET 4.6 |.NET 4.6.2 | .NET Core 2.1 | .NET Core 3.1 | Xamarin.Forms Android v8.1 |

|---------------------------------------------------:|:-------:|:---------:|:-------------:|:-------------:|:--------------------------:|

| `.\Tests\Base\Tests.Base.csproj` | ? | | ? | ? | |

| `.\Tests\FSharp\Tests.FSharp.fsproj` | ? | | ? | ? | |

| `.\Tests\Linq\Tests.csproj` | ? | | ? | ? | |

| `.\Tests\Model\Tests.Model.csproj` | ? | | ? | ? | |

| `.\Tests\Tests.Android\Tests.Android.csproj` | | | | | ? |

| `.\Tests\Tests.Benchmarks\Tests.Benchmarks.csproj` | | ? | ? | ? | |

| `.\Tests\Tests.Playground\Tests.Playground.csproj` | ? | | ? | ? | |

| `.\Tests\Tests.T4\Tests.T4.csproj` | ? | | ? | ? | |

| `.\Tests\VisualBasic\Tests.VisualBasic.vbproj` | ? | | ? | ? | |

Allowed target defines:

- `NETCOREAPP3\_1` - `netcoreapp3.1` target ifdef

- `NETCOREAPP2\_1` - `netcoreapp2.1` target ifdef

- `NET46` - `net46` target ifdef

- `AZURE` - for Azure Pipelines CI builds

## Building

You can use the solution to build and run tests. Also you can build whole solution or library using the following batch files:

\* `.\Build.cmd` - builds all the projects in the solution for Debug, Release and Azure configurations

\* `.\Compile.cmd` - builds LinqToDB project for Debug and Release configurations

\* `.\Clean.cmd` - cleanups solution projects for Debug, Release and Azure configurations

\* `.\Test.cmd` - build `Debug` configuration and run tests for `net46` and `netcoreapp2.1` targets. You can set other configuration by passing it as first paramenter, disable test targets by passing 0 to second(for `net46`) or third (for `netcoreapp2.1`) parameter and format (default:html) as 4th parameter.

Example of running Release build tests for `netcoreapp2.1` only with trx as output:

```

test.cmd Release 0 1 trx

```

### Different platforms support

Because of compiling for different platforms we do use:

\* Conditional compilation. See supported defines above

\* Implementing missing classes and enums. There are some under `.\Source\LinqToDB\Compatibility` folder

## Branches

\* `master` - current development branch for next release

\* `release` - branch with the latest release

## Run tests

NUnit3 is used as unit testing framework. Most of tests are run for all supported databases, and written in same pattern:

```cs

[TestFixture]

public class Test: TestBase // TestBase - base class, provides base methods and object data sources

{

// DataSourcesAttribute - implements NUnit IParameterDataSource to provide testcases for enabled database providers

[Test]

public void Test([DataSources] string context)

{

// TestBase.GetDataContext - creates new IDataContext

// also supports creation of remote client and server

// for remote contexts

using(var db = GetDataContext(context))

{

// Here is the most interesting

// this.Person - list of persons, corresponding Person table in database (derived from TestBase)

// db.Person - database table

// So test checks that LINQ to Objects query produces the same result as executed database query

AreEqual(this.Person.Where(\_ => \_.Name == "John"), db.Person.Where(\_ => \_.Name == "John"));

}

}

}

```

### Configure data providers for tests

`DataSourcesAttribute` generates tests for each enabled data provider. Configuration is taken

from `.\Tests\Linq\DataProviders.json` and `.\Tests\Linq\UserDataProviders.json` (used first, if exists).

Configuration file is used to specify user-specific settings such as connection strings to test databases and

list of providers to test.

The `[User]DataProviders.json` is a regular JSON file:

#### UserDataProviders.json example (with description)

```js

{

// .net framework 4.6 test configuration

"NET46" :

{

// base configuration to inherit settings from

// Inheritance rules:

// - DefaultConfiguration, TraceLevel, Providers - use value from base configuration only if it is not defined in current configuration

// - Connections - merge current and base connection strings

"BasedOn" : "LocalConnectionStrings",

// default provider, used as a source of reference data

// LINQ to DB uses SQLite for it and you hardly need to change it

"DefaultConfiguration" : "SQLite.Classic",

// logging level

// Supported values: Off, Error, Warning, Info, Verbose

// Default level: Info

"TraceLevel" : "Error",

// list of database providers, enabled for current test configuration

"Providers" :

[

"Access",

"SqlCe",

"SQLite.Classic",

"SQLite.MS",

"Northwind.SQLite",

"Northwind.SQLite.MS",

"SqlServer",

"SqlServer.2014",

"SqlServer.2012", "SqlServer.2012.1",

"SqlServer.2008", "SqlServer.2008.1",

"SqlServer.2005", "SqlServer.2005.1",

"SqlAzure",

"DB2",

"Firebird",

"Informix",

"MySql",

"MariaDB",

"Oracle.Native",

"Oracle.Managed",

"PostgreSQL",

"Sybase",

"Northwind",

"TestNoopProvider"

],

// list of test skip categories, disabled for current test configuration

// to set test skip category, use SkipCategoryAttribute on test method, class or whole assembly

"Skip" :

[

"Access.12"

]

},

// .net core 2.1 test configuration

"CORE21" :

{

"BasedOn" : "LocalConnectionStrings",

"Providers" :

[

"SQLite.MS",

"Northwind.SQLite.MS",

"SqlServer",

"SqlServer.2014",

"SqlServer.2012", "SqlServer.2012.1",

"SqlServer.2008", "SqlServer.2008.1",

"SqlServer.2005", "SqlServer.2005.1",

"SqlAzure",

"Firebird",

"MySql",

"MariaDB",

"PostgreSQL",

"Northwind",

"TestNoopProvider"

]

},

// list of connection strings for all providers

"LocalConnectionStrings":

{

"BasedOn" : "CommonConnectionStrings",

"Connections" :

{

// override connection string for SqlAzure provider

// all other providers will use default inherited connection strings from CommonConnectionStrings configuration

"SqlAzure" :

{

"Provider" : "System.Data.SqlClient",

"ConnectionString" : "Server=tcp:xxxxxxxxx.database.windows.net,1433;Database=TestData;User ID=TestUser@zzzzzzzzz;Password=TestPassword;Trusted\_Connection=False;Encrypt=True;"

}

}

}

}

```

To define your own configurations \*\*DO NOT EDIT\*\* `DataProviders.json` - create `.\Tests\Linq\UserDataProviders.json` and define needed configurations.

Tests execution depends on `\_CreateData.\*` tests executed first. Those tests recreate test databases and populate them with test data, so if you are going to run one test be sure to run `\_CreateData` before it manually.

Also - if your test changes database data, be sure to revert those changes (!) to avoid side effects for other tests.

## Continuous Integration

We do run builds and tests with:

\* [Azure Pipelines](https://dev.azure.com/linq2db/linq2db/\_build?definitionId=1) [azure-pipelines.yml](https://github.com/linq2db/linq2db/blob/master/azure-pipelines.yml).

It builds solution, generate and publish nugets and runs tests for:

\* .Net 4.6

\* .Net Core 2.1 (Windows/Linux and MacOS)

For more details check [readme](https://github.com/linq2db/linq2db/blob/master/Build/Azure/README.md)

CI builds are done for all branches and PRs.

- Tests run for all branches and PRs except `release` branch

- Nugets publishing to [Azure feeds](https://dev.azure.com/linq2db/linq2db/\_packaging?\_a=feed&feed=linq2db) enabled only for `branch`

- Nugets publishing to [Nuget.org](https://www.nuget.org/profiles/LinqToDB) enabled only for `release` branch

### Skip CI build

If you want to skip building commit by CI (for example you have changed \*.md files only) check this [message](https://developercommunity.visualstudio.com/comments/503497/view.html).

### Publishing packages

\* \*\*"Nightly" builds\*\* packages are published to [Azure feeds](https://dev.azure.com/linq2db/linq2db/\_packaging?\_a=feed&feed=linq2db) for each successful build of \*\*master\*\* branch.

\* \*\*Release\*\* packages are published to [Nuget.org](https://www.nuget.org/profiles/LinqToDB) for each successful build of \*\*release\*\* branch.

## Building releases

1. Update [Release Notes](https://github.com/linq2db/linq2db/wiki/Releases-and-Roadmap) and create empty entry for vNext release

1. Create PR from `master` to `release` branch, in comments add [@testers](https://github.com/linq2db/linq2db/wiki/How-can-i-help#testing-how-to) to notify all testers that we are ready to release

1. Wait few days for feedback from testers and approval from contributors

1. Merge PR

1. [Tag release](https://github.com/linq2db/linq2db/releases)

1. Update versions in `master` branch (this will lead to publish all next `master` builds as new version RC):

\* in [.\azure-pipelines.yml](https://github.com/linq2db/linq2db/blob/master/azure-pipelines.yml) set `assemblyVersion` and `nugetVersion` parameters to next version. Always use next minor version and change it to major only before release, if it should be new major version release

## Process

In general you should follow simple rules:

\* Development rules and regulations, code style

\* Do not add new features without tests

\* Avoid direct pushes to `master` and `release` branches

\* To fix some issue or implement new feature create new branch and make pull request after you are ready to merge or create pull request as `work-in-progress` pull request. Merge your PR only after contributors' review.

\* If you do have repository write access, it is recommended to use central repository instead of fork

\* Do not add new public classes, properties, methods without XML documentation on them

\* Read issues and help users

\* Do not EF :)