Ignite.NET Development

Introduction

Ignite.NET is built on top of Ignite and provides native APIs for .NET developers.

- Documentation: https://apacheignite-net.readme.io/docs
- Source code is in modules/platforms/dotnet folder

Requirements

For Linux instructions see below.

- Oracle JDK 7+
- .NET Framework 4.0+
- PowerShell 3.0+
- Visual Studio 2010+
- Apache Maven

Getting Started With Modern Tooling

Even though Ignite.NET can be developed and run on VS2010 and JDK7, it is more common to use latest Visual Studio and JDK.

Assuming that we are on up-to-date Windows x64 installation:

Install software

1. Install Java **JDK** (not JRE) 8.x (Java 9 is not supported): http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Set JAVA_HOME environment variable to point to the JDK installation directory (something like "C:\Program Files\Java\jdk1.8.0_111")
3. Download and unzip Apache Maven to some folder: https://maven.apache.org/download.cgi
4. Add Maven bin folder to PATH environment variable (example: "c:\Programs\Maven\apache-maven-3.3.9\bin\")
5. Install Visual Studio 2017 (free Community Edition will do).

Get source code

- git clone https://git-wip-us.apache.org/repos/asf/ignite

For more details on working with GitHub forks and pull requests see How to Contribute#GitProcess

Build Java sources

In root source folder execute "**mvn clean package -DskipTests -Dmaven.javadoc.skip=true -U -Plgpl,-examples,-clean-libs,-release,-scala,-clientDocs**" command.

When doing pure .NET development, you only need to run this every time you update your branch from master.

Build .NET sources

- Open **modules\platforms\dotnet\Apache.Ignite.sln** in Visual Studio
- Build solution

Running Unit Tests

Ignite.NET uses NUnit for unit tests. You can run them with ReSharper right away, or with NUnit Test Adapter for Visual Studio.

- Make sure to **disable assembly shadow copy** in unit test runner settings

Running Tests On TeamCity

- Ignite uses TeamCity-based continuous integration: http://ci.ignite.apache.org/
- If you do .NET-only development, you are interested only in test suites having "Platform.NET" in the name
- Create an account to be able to start new runs
- .NET NuGet suite can be used to produce release NuGet binaries (see Artifacts tab of a finished build)

Coding Guidelines

- Ignite.NET follows standard MSDN Framework Design Guidelines in regards to naming, project and namespace structure, type design, etc.
- Private fields should be _underscoreCamelCase (this is not covered by guidelines above)
- Non-private instance fields are not allowed
- Line length should not exceed 120 characters (use Editor Guidelines extension https://marketplace.visualstudio.com/items?itemName=PaulHarrington.EditorGuidelines)

Coding guidelines compliance is checked by .NET Inspections TeamCity suite (see below).

Static Code Analysis

Three code analysis tools are integrated with Ignite.NET project:

- NDepend: see modules/platforms/dotnet/Apache.Ignite.ndproj. We have 6 build machine licenses (see https://svn.apache.org/repos/private/pmc/ignite/licenses/NDepend.txt, PMC-only access), NDepend is installed on all Windows TeamCity agents and runs as part of .NET Inspections. Report can be seen right there on TeamCity build result page, on NDepend tab.
- ReSharper inspections (come integrated with TeamCity). There are two R# project files in modules/platforms/dotnet/, Apache.Ignite.sln.DotSettings and Apache.Ignite.sln.TeamCity.DotSettings. TeamCity file has less rules enabled, but all of them must pass. Another one is used when you open solution in Visual Studio.
- Microsoft FxCop. Since we officially support Visual Studio 2010, standalone FxCop 1.36 is used, see Apache.Ignite.FxCop in modules/platforms/dotnet/. Later Visual Studio versions have this functionality integrated, this can be enabled with "Run code analysis on build" in project settings.

All these tools assume that project has been built in **Debug Any CPU** mode.

Building Release Binaries

There is a build.ps1 file in modules/platforms/dotnet folder (and build.bat that just calls build.ps1 so you don't have to run PowerShell manually).

This build script performs end to end build, including Java, .NET and NuGet.

To build everything and produce release binaries (in bin folder) and NuGet packages (in nupkg folder), run the script without parameters.

Run `Get-Help .\build.ps1 -detailed` PowerShell command to view detailed build script documentation.

Getting Started on Linux and macOS

Ignite.NET can be built, developed, and tested on Linux and macOS, but only partially.
A subset of functionality and tests is available. See Apache.Ignite.DotNetCore.sln.

Requirements

- .NET Core SDK: https://www.microsoft.com/net/download/linux
- JDK: sudo apt-get install default-jdk
- Maven: sudo apt-get install maven
- IDE: Not required. Rider is recommended.

Getting started

- Build Java and .NET: **./build.sh**
- Run tests: **cd Apache.Ignite.Core.Tests.DotNetCore & dotnet test**
- Run specific test: **dotnet test --filter CacheTest**
- IDE: Open **Apache.Ignite.DotNetCore.sln**