Methodology & Practice in (.NET) Development.

Xianyi Cui

Content

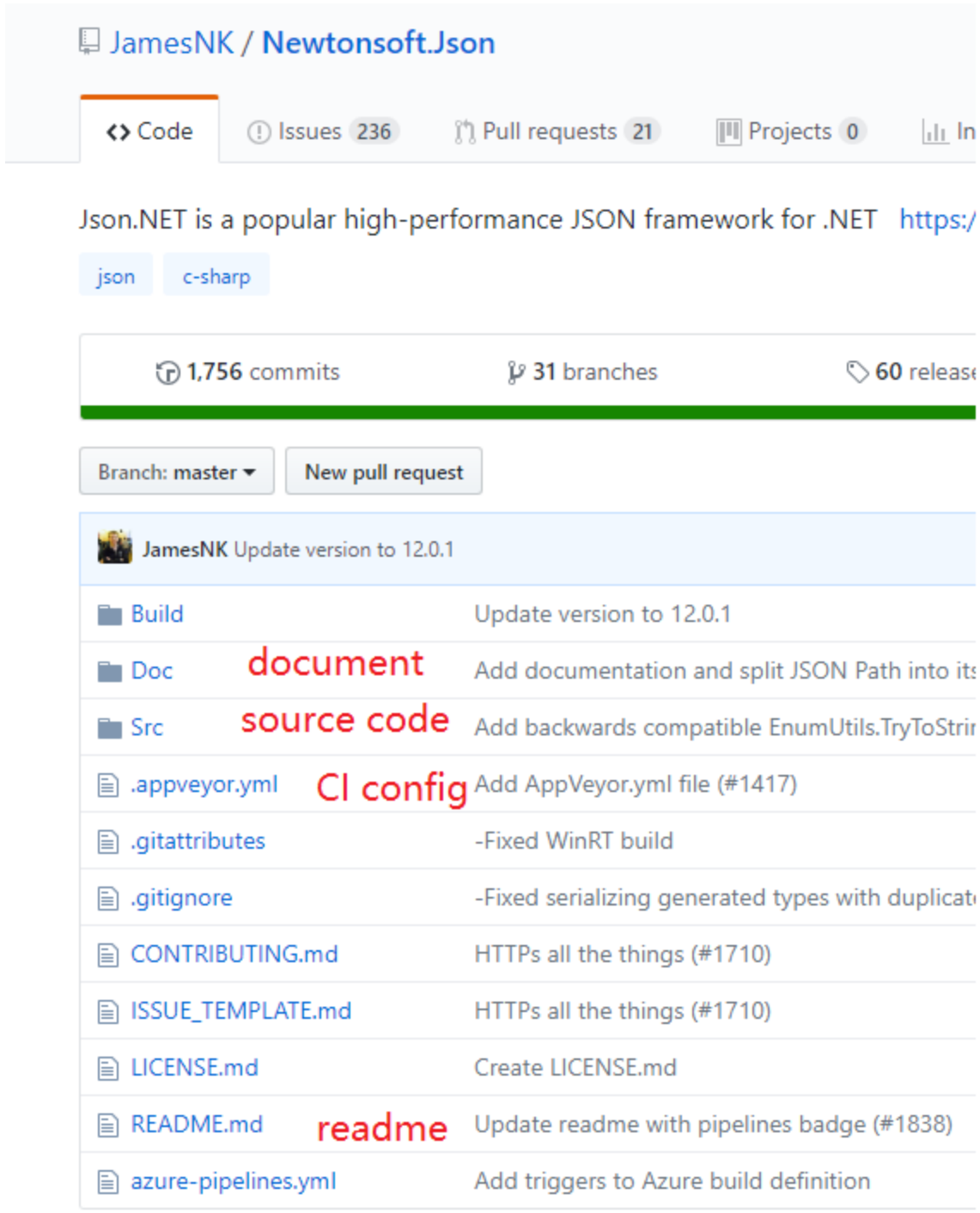
Part 1: Before coding

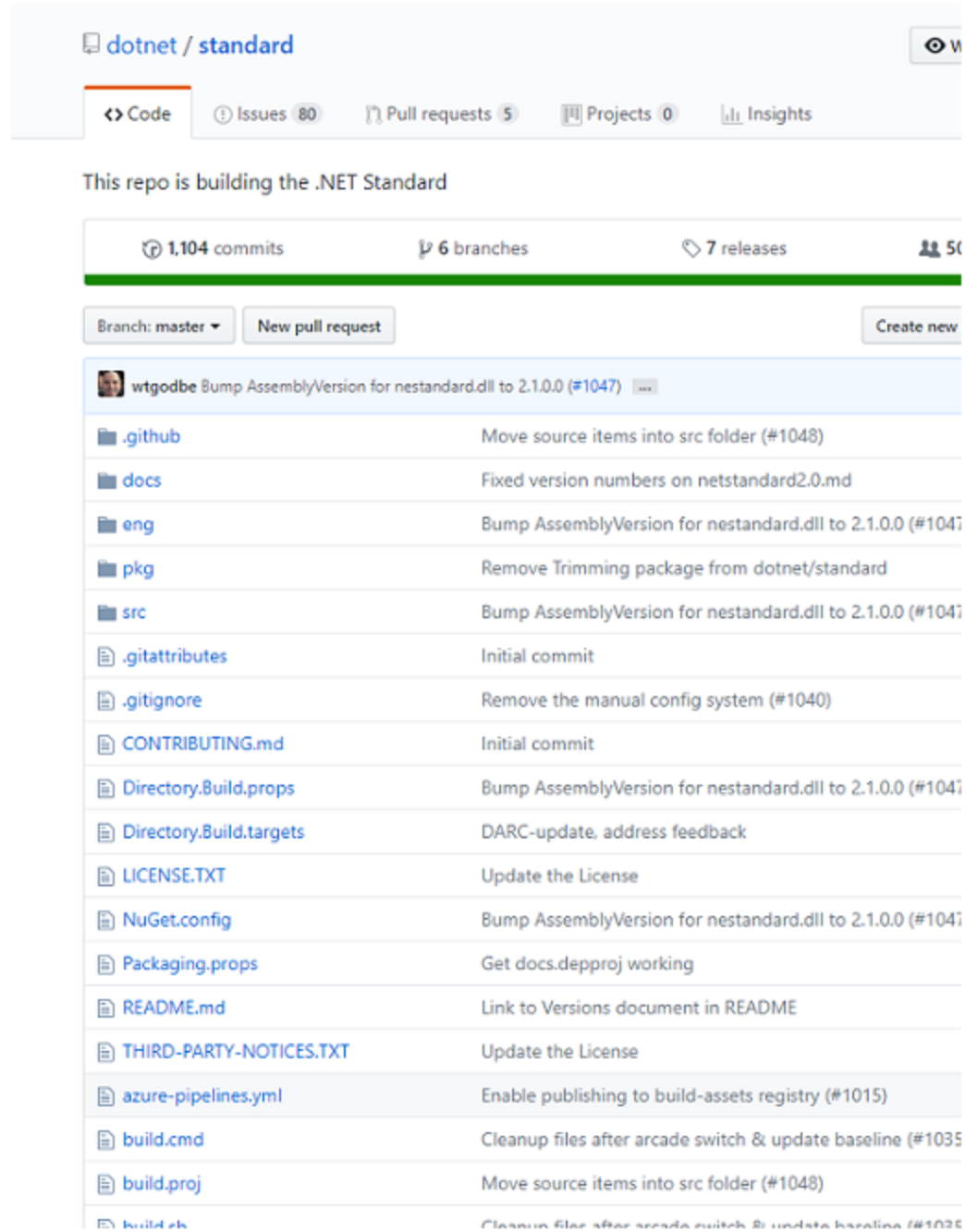
Code Repository

What is the structure?

Category	Samples	Usage
Source Code	/src	contain the source codes or tests
Documents	/docs *.rst; *.md	docs folder or markdown/reStructuredText for documents
Other functional files	git config files CI* config files(appveyor,travis,circle) package dependency(package.json,requirement.txt) ...	For extended requirements & advanced features
Scripts	init build	including some script for us to execute for environment preparing. Save time from duplicate typing

- All of them required?
- Why not put source code out side?





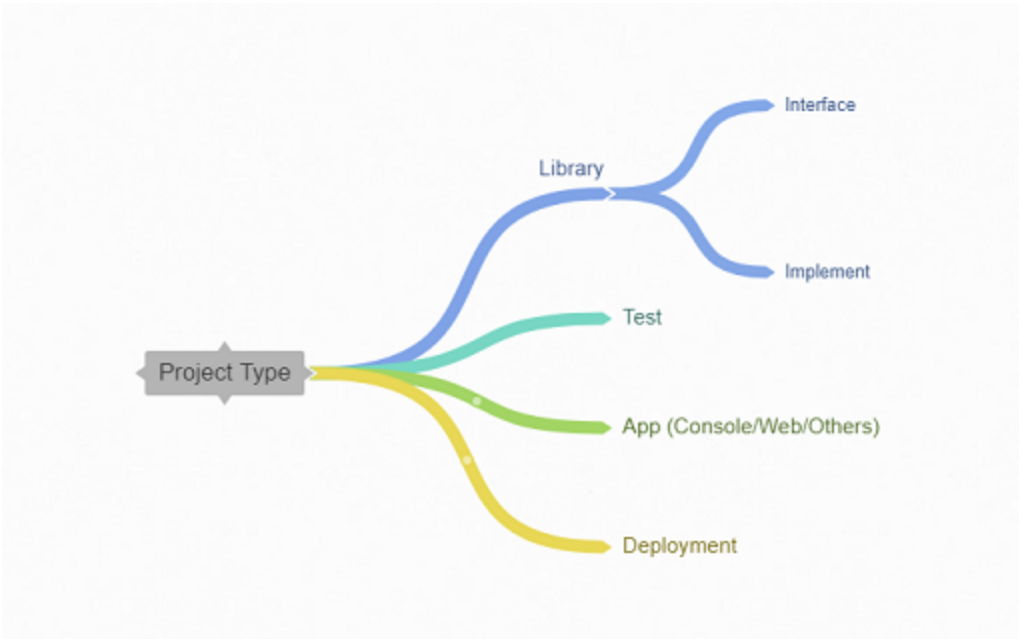
- **NewtonJson: https://github.com/JamesNK/Newtonsoft.Json**
- **Pyinstaller: https://github.com/pyinstaller/pyinstaller**
- **NetCore(multiple Readme): https://github.com/dotnet/core**

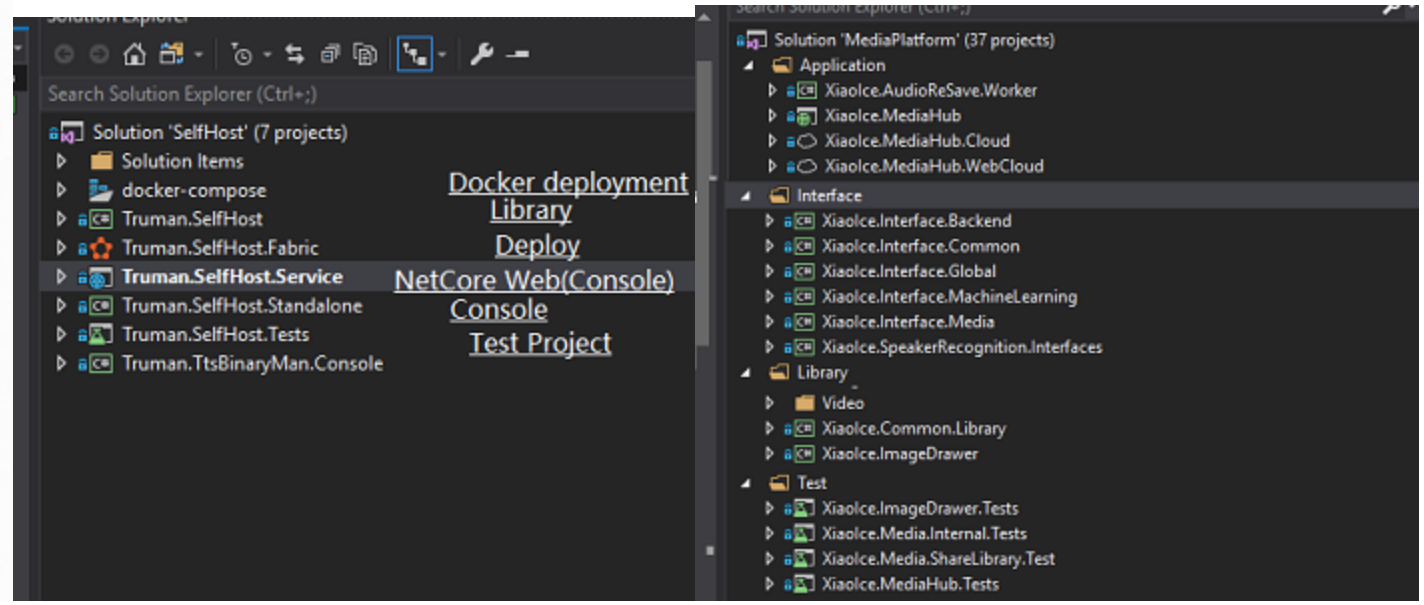
Projects

What is in project?

Items	Note
Source code	-
Config/resource file	Avoid put dependency like .dll/.jar except native
Dependency/Description file	.csproj/.vcxproj *package.config* in .NET, pom.xml/build.gradle in Java,

Types of projects?





Practice on Project

How to construct?

1. For large (extended) project. Create *Interface* Project. And also the 'Implement Project'(eg. MongoDB CSharp Driver, XiaoIce).
2. For task specific project. Just create library project.(NewtonJson)
3. Create Test project(NUnit, XUnit for .NET) for test. Keep your playground code. If you write in Console you might need to delete them.

Pakcage Reference?

Reference type	Pros	Cons
Project reference	Latest dependency change could be applyed Easy debug	Build dependency each time Extra large repo Even with git submodule Build break risk
NuGet reference	Fast on build	Debug require aditional nupkg or pdb Cost on package publish

.NET Project

Net Core & Net Framework

- Previous using Net framework(Heavy(dependency), Platform fixed(mono is a kind of trade off by third party))
- .Net Core is platform. Open source. Nuget managed packages.
- .Net Core also provide a new standard of project management(new csproj). And use PackageReference replace package.config. (Compatible with old one)
- For Application(Runtime)
- Not compatible between each other

Net Standard

- **Library** standard(constraint). Not for runtime.
- Could be referenced by Standard/Core/Framework with condition

Takeaway

1. Create/Reuse library/project depends on requirement.

2. Concept of interface! Concept of interface! Concept of interface!

- Large scale system: Interface project for behaviour reusing.

- Standalone feature: Interface & implement in same library/project.

3. App is just a wrapper for interactive.

4. Web is also a kind of Console -- Do more Self-host(Owin, Kestrel, Flask)

5. Project as a service.

6. Jump out of the sln concept for .NET development

Clean Environment

1. Build environment - DockerFile
2. Package dependency - NuGet.Config
3. Resource file dependency - Relative path/Embedded resource/resx file

Part 2: Working on project

Part 3: Other thinking

IDE vs CLI

Tools	Pros	Cons
IDE	1. Easy to use 2. Clear with UI 3. Plugins	1. Ignore a lot of details 2. Unstable
CLI	1. Make users understand the details 2. Developer friendly	1. Not intuitive 2. Cost time to learn

***IDE: *I*ntegrated *D*evelopment *E*nvironment**
***CLI: *C*ommand *L*ine *I*nterface**

Console vs Platform/Framework specific(web, worker)

Console:

- Flexible in development & deployment
- More and more library/framework support SelfHost/ConsoleHost (Owin, Kestrel)
- Understand the detail better.

Platform/Framework specific

- Provide a managed process on development. Avoid some mistake at beginning.
- Hard to migrate(1. Worker role/Web role 2. .NET Web application binded to IIS)

Summary

- Focus more on folder structure not project/sln.
- Cleaning environment & reusable module/library
- Use more 'vs' when searching (Mono vs Net Core).
- Try more command line tool.

Reference

Concepts	Explain	Tools/Service
CI/CD	Continous Integration/Continous Delivery	Appveyor, VSTS(Cloud build), Travis,Circle
Scaffolding	Structure of a project for languages	Yeoman
Package management	Dependency managment	NuGet, npm, pip, scoop, chocolatey

Thanks