Arkane Libraries

Release pre

Jul 09, 2019

Contents:

CHAPTER 1

Features

All the stuffs.

1.1 Dependencies

All the Arkane Libraries require the following NuGet packages:

- JetBrains.Annotations (2019.1.1)
- PostSharp (6.2.5)
- PostSharp.Patterns.Common (6.2.5)

1.2 License

These libraries are released under the MIT License, as included below.

1.2.1 MIT License

HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.2.2 Annotations

namespace ArkaneSystems.Arkane.Annotations

This namespace, found in the *Arkane.Annotations* assembly/NuGet package, contains annotative attributes which add information to an assembly or its members, chiefly for the use of external tools, but which can also be conveniently accessed via reflection.

AddGitStampAttribute

[AddGitStamp]

When added to an assembly, this attribute marks it, post-compilation, with precise version information derived from its git repository. Specifically, it extracts the git information from the repo, combines it with the assembly version, and places it in the *AssemblyGitVersionAttribute*, which see for more details.

Note: The use of this attribute requires git to be in the PATH at compile-time.

The function of this attribute/aspect was derived from that of the Fody equivalent, Fody.Stamp (https://github.com/Fody/Stamp), although independently implemented.

AssemblyGitVersionAttribute

Note: This attribute is automatically applied to the assembly post-compilation by the *AddGitStampAttribute*. It should not be added in the source code.

Properties

public string GitVersion { get; }

A git version string, as returned by *git log -1 –format=”%d %H”*. If the repository has been modified and the changes not yet committed, the string ” modified” is appended.

CodeQualityAttribute

[CodeQuality (*SoftwareQuality* quality = SoftwareQuality.Win, *SoftwareReliability* reliability = SoftwareReliability.Solid)]

An attribute for use in ongoing code review indicating a subjective assessment of the quality and reliability of the code it is attached to.

This information can be read out of the *Quality* and *Reliability* properties.

DoNotEditAttribute

[DoNotEdit (string byOrderOf, *Uneditability* reason)]

Indicates that the attached code should not be edited without consulting the specified person, for the given reason.

These values can be read out of the *ByOrderOf* and *Reason* properties, respectively.

ProgrammingLanguages

public enum ProgrammingLanguages

An enumeration of programming languages, primarily intended for use with the *SourceLanguageAttribute*.

The enumeration contains the following elements:

CSharp Indicates the C# language.

VisualBasic Indicates the Visual Basic.NET language.

FSharp Indicates the F# language.

Il Indicates raw IL, written directly.

Python Indicates the Python language.

PowerShell Indicates the PowerShell scripting language.

JavaScript Indicates the JavaScript language.

Simple Attributes

These attributes simply incorporate the associated information into the assembly, and have no active function of their own.

Author

[Author (string name, string email)]

This attribute embeds the authorship of an entire assembly or specific class, method, etc. Both parameters are required, and the *email* parameter must be a valid e-mail address.

This information can be read out using the *Name* and *EmailAddress* properties.

BugFix

[BugFix (int caseNumber, string Comments = “Comments.”)]

This attribute embeds the case number of a bug fix - for the specific class, method, etc. - in an issue-tracking system. The caseNumber parameter is required; additional comments may optionally be given.

This information can be read out of the *CaseNumber* and *Comments* properties.

Documentation

[Documentation (string uri)]

This attribute, applicable only to assemblies, embeds the URL where documentation for that assembly may be found. The *uri* parameter must be a valid URL using the http:, https:, or ftp: protocol.

This information can be read out using the *Location* property.

LegacyWrapper

[LegacyWrapper (string comments)]

This attribute indicates that the following class or method exists to insulate one from the liveliest awfulness of the legacy code that it's wrapped around.

Comments can be read out using the *Comments* property.

ObligatoryQuotation

[ObligatoryQuotation (string quotation, string source, string citation)]

It had to be said, so I did.

Why? Why not just in the comments? Well, it's because this sort of thing is useful insight into the mind of the developer, and - assuming we're not obfuscating - that should be available right there in the assembly as well as the source.

The citation for the original quotation should preferably, not necessarily, be supplied in URI format, and may be null.

The quotation can be read out using the *Quotation*, *Source*, and *Citation* properties.

SoftwareQuality

public enum SoftwareQuality

An enumeration of subjective assessments of code quality, for use with the *CodeQualityAttribute*.

The terms used in this enumeration are derived, in several cases, from the Jargon File (http://www.catb.org/jargon/html/), 4.4.8.

The values of the enumeration are as follows:

Monstrosity "A ridiculously elephantine program or system, esp. one that is buggy or only marginally functional."

BrainDamage "Obviously wrong; cretinous; demented. There is an implication that the person responsible must have suffered brain damage, because he should have known better. Calling something brain-damaged is really bad; it also implies it is unusable, and that its failure to work is due to poor design rather than some accident. "Only six monocase characters per file name? Now that's brain-damaged!""

Screw "A lose, usually in software. Especially used for user-visible misbehavior caused by a bug or misfeature."

Bug "An unwanted and unintended property of a program or piece of hardware, esp. one that causes it to malfunction. Antonym of feature."

Lose "To be exceptionally unaesthetic or crocky."

Misfeature "A feature that eventually causes lossage, possibly because it is not adequate for a new situation that has evolved. Since it results from a deliberate and properly implemented feature, a misfeature is not a bug. Nor is it a simple unforeseen side effect; the term implies that the feature in question was carefully planned, but its long-term consequences were not accurately or adequately predicted (which is quite different from not having thought ahead at all). A misfeature can be a particularly stubborn problem to resolve, because fixing it usually involves a substantial philosophical change to the structure of the system involved."

Crock "1. An awkward feature or programming technique that ought to be made cleaner. For example, using small integers to represent error codes without the program interpreting them to the user (as in, for example, Unix make(1), which returns code 139 for a process that dies due to segfault). 2. A technique that works acceptably, but which is quite prone to failure if disturbed in the least."

Kluge "1. A Rube Goldberg (or Heath Robinson) device, whether in hardware or software. 2. A clever programming trick intended to solve a particular nasty case in an expedient, if not clear, manner. Often used to repair bugs. Often involves ad-hockery and verges on being a crock. 3. Something that works for the wrong reason."

Hack "1. Originally, a quick job that produces what is needed, but not well. 2. An incredibly good, and perhaps very time-consuming, piece of work that produces exactly what is needed."

Win The assumed default quality. "1. To succeed. A program wins if no unexpected conditions arise, or (especially) if it is sufficiently robust to take exceptions in stride. 2. n. Success, or a specific instance thereof. A pleasing outcome."

Feature "1. A good property or behavior (as of a program). Whether it was intended or not is immaterial. 2. An intended property or behavior (as of a program). Whether it is good or not is immaterial (but if bad, it is also a misfeature)."

Elegance "Combining simplicity, power, and a certain ineffable grace of design. Higher praise than 'clever', 'winning', or even cuspy. The French aviator, adventurer, and author Antoine de Saint-Exupery, probably best known for his classic children's book The Little Prince, was also an aircraft designer. He gave us perhaps the best definition of engineering elegance when he said "A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away."

Perfection Unattainably brilliant.

Note: To reflect the above, it's actually not possible to use *SoftwareQuality.Perfection* in the *CodeQualityAttribute*.

Yes, I'm making a philosophical point here.

SoftwareReliability

public enum SoftwareReliability

An enumeration of subjective assessments of code reliability, for use with the *CodeQualityAttribute*.

The terms used in this enumeration are derived, in several cases, from the Jargon File (http://www.catb.org/jargon/html/), 4.4.8.

The values of the enumeration are as follows:

Broken "1. Not working according to design (of programs). This is the mainstream sense. 2. Improperly designed, This sense carries a more or less disparaging implication that the designer should have known better, while sense 1 doesn't necessarily assign blame. Which of senses 1 or 2 is intended is conveyed by context and nonverbal cues."

Flaky "Subject to frequent lossage. This use is of course related to the common slang use of the word to describe a person as eccentric, crazy, or just unreliable. A system that is flaky is working, sort of — enough that you are tempted to try to use it — but fails frequently enough that the odds in favor of finishing what you start are low."

Fragile "Said of software that is functional but easily broken by changes in operating environment or configuration, or by any minor tweak to the software itself. Also, any system that responds inappropriately and disastrously to abnormal but expected external stimuli; e.g., a file system that is usually totally scrambled by a power failure is said to be brittle [fragile]."

Solid The assumed default reliability.

Robust "Said of a system that has demonstrated an ability to recover gracefully from the whole range of exceptional inputs and situations in a given environment. One step below bulletproof. Carries the additional connotation of elegance in addition to just careful attention to detail."

Bulletproof "Used of an algorithm or implementation considered extremely robust; lossage-resistant; capable of correctly recovering from any imaginable exception condition — a rare and valued quality. Implies that the programmer has thought of all possible errors, and added code to protect against each one."

SourceLanguageAttribute

[SourceLanguage (*ProgrammingLanguages* language)]

An attribute, applicable only to assemblies, identifying the original programming language in which an assembly was developed, along with certain other related information.

Properties

public ProgrammingLanguages Language { get; }

The programming language in which this assembly was written.

public bool RewrittenByPostSharp { get; set; }

Has this assembly been rewritten post-compilation by PostSharp?

public bool Obfuscated { get; set; }

Has this assembly been obfuscated post-compilation in other ways, and how?

public string PostCompilationModifications { get; set; }

Uneditability

public enum Uneditability

An enumeration of reasons why the code to which the *DoNotEditAttribute* is applied should not be edited.

The terms used in this enumeration are derived, in several cases, from the Jargon File (http://www.catb.org/jargon/html/), 4.4.8.

The values of the enumeration are as follows:

DeepMagic "An awesomely arcane technique central to a program or system, esp. one neither generally published nor available to hackers at large (compare black art); one that could only have been composed by a true wizard. Compiler optimization techniques and many aspects of OS design used to be deep magic; many techniques in cryptography, signal processing, graphics, and AI still are."

DeeperMagic Like deep magic, but more so. Treat even more carefully than instances of deep magic in the same code.

BlackMagic "A technique that works, though nobody really understands why." Mostly ad-hoc. Since we don't know how it works, if it breaks, we probably can't fix it. All of which is to say - like deep magic, but also evil.

Fragile "Said of software that is functional but easily broken by changes in operating environment or configuration, or by any minor tweak to the software itself. Also, any system that responds inappropriately and disastrously to abnormal but expected external stimuli; e.g., a file system that is usually totally scrambled by a power failure is said to be brittle [fragile]."

Undocumented Relies on knowledge not kept anywhere but the implementer's head. At any rate, you don't have it.

YouAreNotExpectedToUnderstandThis "The canonical comment describing something magic or too complicated to bother explaining properly. From an infamous comment in the context-switching code of the V6 Unix kernel." Don't go there, okay? Just don't.

WarningAttribute

[Warning (string reason)]

The warning attribute is an active attribute that causes a compiler warning to be reported for the attached piece of code during compilation.

> **Warning:** If warnings are treated as errors, the *WarningAttribute* will actually prevent successful compilation.

Index