Public Annotations

Code is data that communicates your intent. If you have no special relationship with your compiler, you don’t need any special data to communicate additional intent.

Once you’re in an open compiler world, you may need to communicate with your compiler. This features has been called “design time attributes” and “annotations.” I’m adding this feature to RoslynDom and calling it “PublicAnnotations.”

I’ll try to always remember to say “public annotations” to differentiate it from Roslyn private annotations.

# Why not just use attributes?

Attributes do not work very well to communicate with the compiler for at least these reasons:

* Can’t tell what’s available at runtime
* If design attributes are visible at runtime, they become a contract
* Can result in build dependency
  + If one player removes attributes it’s done with to avoid runtime contracts
* Must follow attribute syntax
* Only constants allowed
  + No lambda expressions
  + No generic types
  + No expressions
* Can’t be placed in all desired locations
  + Not on namespaces, files, or in random locations inside or outside methods

I think the first is actually the biggest issue. I think it’s important to differentiate communications with the compiler pipeline, including design time with the Visual Studio/Roslyn linkage, and runtime attributes. But even if you disagree with that, attributes simply don’t work because of limitations in attribute content and attribute location.

# The Syntax

Eventually there will be enough examples of public annotations that an obvious syntax can be included in the languages. I’m not willing to wait as I need public annotations right now, like today.

The current syntax has to reside inside a comment. That’s the only way to solve the content and location limitations of attributes without changing the compiler.

The syntax should be clearly differentiated from all other lines of code to allow easy recognition by human, parser/loader and later IDE colorization. The syntax should also be easily found via RegEx to allow updates if we get language support.

RoslynDom now supports the following, with any desired whitespace within the line.

//[[ NameOfAnnotation(stuff) ]]

This currently requires the annotation appear on a single line and I’m not currently supporting end of line comments.

Because it’s familiar to you, the annotation looks like an attribute, except for that funny double square bracket. It does not need an attribute class to exist anywhere, and one generally will not exist.

Just like attributes, the following variations are all supported, along with the logical combinations:

//[[ NameOfAnnotation() ]]

//[[ NameOfAnnotation(stuff) ]]

//[[ NameOfAnnotation(name:stuff, name2 : stuff) ]]

//[[ NameOfAnnotation(name=stuff, name2 = stuff) ]]

The common way to add annotations will be to include them in your source code. You can also add annotations explicitly with the AddPublicAnnotationValue(string name, object value) and the AddPublicAnnotationValue(string name, string key, object value) methods.

Public annotations with no parameters are just accessed to see if the public annotation exists via its name.

A single positional value is supported and is accessed via the public annotation name.

Named values are accessed by the public annotation name and the value name as a key.

# Legal locations for public annotations

Annotations are currently legal on using statements, namespaces, types and members. They are also legal at the file or root level.

[TestMethod]

public void Can\_get\_and\_retrieve\_public\_annotations()

{

var csharpCode = @"

//[[ file: kad\_Test4(val1 = ""George"", val2 = 43) ]]

//[[ kad\_Test1(val1 : ""Fred"", val2 : 40) ]]

using Foo;

//[[ kad\_Test2(""Bill"", val2 : 41) ]]

//[[ kad\_Test3(val1 =""Percy"", val2 : 42) ]]

public class MyClass

{ }

";

This illustrates a challenge. A likely location for public annotations is the file level. But I then need to distinguish between the file or root level public annotation and annotations on the first item in the file. I decided to do this by prefixing the file public annotation. I am currently supporting both file and root.

# Accessing your values

The following methods are available for accessing public annotations

bool HasPublicAnnotation(string name);

void AddPublicAnnotationValue(string name, string key, object value);

void AddPublicAnnotationValue(string name, object value);

object GetPublicAnnotationValue(string name, string key);

object GetPublicAnnotationValue(string name);

T GetPublicAnnotationValue<T>(string name);

T GetPublicAnnotationValue<T>(string name, string key);

These methods are available on all items via the IDom interface.

# Full example

Here’s the full example from the Scenario\_PatternMatchingSelection class of the RoslynDomExampleTests project.

[TestMethod]

public void Can\_get\_and\_retrieve\_public\_annotations()

{

var csharpCode = @"

//[[ file: kad\_Test4(val1 = ""George"", val2 = 43) ]]

//[[ kad\_Test1(val1 : ""Fred"", val2 : 40) ]]

using Foo;

//[[ kad\_Test2(""Bill"", val2 : 41) ]]

//[[ kad\_Test3(val1 =""Percy"", val2 : 42) ]]

public class MyClass

{ }

";

var root = RDomFactory.GetRootFromString(csharpCode);

var using1 = root.Usings.First();

Assert.AreEqual("Fred",using1.GetPublicAnnotationValue <string>("kad\_Test1","val1"));

Assert.AreEqual("Fred",using1.GetPublicAnnotationValue("kad\_Test1","val1"));

Assert.AreEqual(40, using1.GetPublicAnnotationValue <int>("kad\_Test1","val2"));

Assert.AreEqual(40, using1.GetPublicAnnotationValue("kad\_Test1","val2"));

var class1 = root.RootClasses.First();

Assert.AreEqual("Bill", class1.GetPublicAnnotationValue( "kad\_Test2"));

Assert.AreEqual(41, class1.GetPublicAnnotationValue("kad\_Test2", "val2"));

Assert.AreEqual("Percy", class1.GetPublicAnnotationValue("kad\_Test3", "val1"));

Assert.AreEqual(42, class1.GetPublicAnnotationValue("kad\_Test3", "val2"));

Assert.AreEqual("George", root.GetPublicAnnotationValue("kad\_Test4", "val1"));

Assert.AreEqual(43, root.GetPublicAnnotationValue("kad\_Test4", "val2"));

}