Updates to RoslynDom 1.0.10 Alpha

# New Statement Load Mechanism

Updates to RoslynDom 1.0.9 Alpha

# New Parent property on all items

IDom now contains a Parent property of IDom type

In any tree things can become, well interesting, if nodes appear in more than one location. This is particularly damaging in a tree that takes characteristics from context – which happens with naming (namespaces and nested classes) in the .NET class model. Thus, by intent, no item may appear in more than one location in the tree.

When a member is cloned, its parent is not copied with it. Also, parent and parent properties are not used in determining same intent.

# Real-time Namespace property

Previously, Namespace was stored from the symbol when the instance was created. Because Namespace ***is*** contextual, this was incorrect. Namespace is now calculated from the parent hierarchy when the namespace is requested for all classes except RDomReferencedType. This resulted in some changes in Namespace results, including the result from

Namespace testing.Foo

Which previously returned Foo and now returns testing.Foo.

The Namespace in RDomRefernecedType is the namespace of the type being referenced, so is still retrieved from the symbol on load.

# AddOrMoveMember and RemoveMember methods

Methods to add members to containers have been added to new IRDomStemContainer, IRDomTypeContainer and IRDomCodeContainer interfaces.

As discussed under the heading “New Parent property on all items,” IDom items may not appear in more than one location in the tree. The AddOrMove semantics reflect this. I actually think moving will be a rare task, but if you accidently add an item to a new location in the tree, RoslynDom will remove it from the prior location and I wanted naming to clarify this.

I may add an “AddCloneOfMember” to simplify the process of cloning a member and adding it to a new location after changes. This is the anticipated use case.

# ICodeContainer and ICodeMember interfaces

There are new ICodeContainer and ICodeMember interfaces. Support for intra-member features (code) remains almost non-existent in this version.

# RawItem and OriginalRawItem semantic changes

RawItem and the new OriginalRawItem on the IDom interface represent the underlying data in an agnostic way. IDom is agnostic on mutability so there may be future implementations where RawItem and OriginalRawItem are always the same. I want the semantics to be clear that RawItem is the best current capturing of the tree, and OriginalRawItem is the original unchanged item. This intentionally implies that the original must be maintained.

TypedSyntax and OriginalTypedSyntax are the RDom implementations of these generalized ideas.

# AddMember method added to RDomStemContainer and RDomBaseType

To support mutability, AddMember methods were added to these two base classes. This makes the ability to add types and type members available to appropriate types, namespaces, and the root.

# Changed return of PublicAnnotationList.GetValue(string key)

Previously this returned the default value, which blocked access to other values. It now returns the PublicAnnotation. The default value remains accessible by GetValue(name, name).

# Changed PublicAnnotation to a Class

PublicAnnotation was a struct. This was the only struct in the system and I felt the value/reference semantic difference would be detrimental to maintenance. As part of this, I removed the equality testing and added a SameIntent method.

# Added IHasSameIntentMethod interface

Another characteristic interface was added for the SameIntent methods. This is for consistency with other characteristic interface usage.

# Moved SameIntent to a subsystem in RoslynDom.Common

This code may eventually run with a DI, but for now, if the interface data matches, they match.

# Changed SameIntent method type parameter

Previously the SameIntent method appeared on the strongly typed IDom<T> interface and could only be called on items of the same type. This was overly restrictive, so the method was changed to have a local strongly typed parameter, constraint only to be a class. Comparing different IDom types of the current implementations will always return false, although it is possible that a derived class could be created that had different behavior, but the same intent, as one of the existing implementation classes, and could therefore return true as the same intent. This was also done to support scenarios where the type is not known, such as public annotations that might be IDom types.

# Changed inheritance semantics of SameIntent() method

The previous inheritance semantics of the SameIntent method were to directly override the public SameIntent method. This method is no longer virtual. Instead override the CheckSameIntent protected method. Be sure to call the base CheckSameIntent method for correct behavior.

# SameIntent and names

Type members (fields, properties, methods and enum) do not include outer name when considering same intent.

Stem members (types, namespaces) do not include namespace/qualified name in same intent.

# Added IHasLookupValues interface

Added this interface to reduce dependencies in an upcoming project.

# Virtual Matches method added to IDom

Immediately this allows CheckSameIntentChild to better find the other child to compare to. It also provides a generalized way to find items in a list.

# Changed name of RDomTypeParameter. HasReferenceTypeConstraint

Was previously HasReferenceConstraint. Changed for consistency. Also changed ITypeParameter

# Changed name of MemberKind, StemMemberKind and LiteralKind

The suffix “type” is confusing. Switched these enums and property names to “kind”

# BuildSyntax

Implementation of syntax recreation from changed nodes is begun, not complete.

# Internal cleanup

* Separated RDomBase classes into separate files
* Created SameIntentHelper
* Changes to how IHasNamespace properties are stored and used
* IHasNamespace moved from IStemMember to IType and INamespace
* IUsing now includes IStemMember
* StemMembers property of Namespaces and Root now include usings
* Fixed some bugs in RDomField attributes

Updates to RoslynDom 1.0.8 Alpha

*Thanks to* [*Llewellyn Falco*](http://llewellynfalco.blogspot.com/) *for his ongoing support and insight. He is encouraging my frequent releases of RoslynDom, and to get a preliminary release of CodeFirstMetadata to NuGet as well as GitHub real soon.*

You can get the bits [here](https://github.com/KathleenDollard/RoslynDOM) and the download the NuGet package through Visual Studio package manager or another NuGet client.

These are experimental releases, and as such are not signed.

# SameIntent methods

For the work I am doing, I am more interested in the intent of the code than the details of it. There are a number of ways different code can result in identical behavior including ordering of members, attribute syntax details, namespace nesting, and use of named parameters. The first version of the SameIntent methods are fairly conservative – not all code with identical results will be found, just the big, common issues.

# Cloning as Copy methods

I added a feature to clone RoslynDom items. This is a precursor to adding mutability, but mutability is not yet available. This involved changing a number of items from direct access to the underlying trees to retrieving this information into local fields. All tests pass, but if you find a missing feature or anything funny, let me know.

# PublicAnnotationList replaces IEnumerable<PublicAnnotation>

Previously RDomBase managed a list of PublicAnnotation. This was a bad refactoring of concerns, so I added a PublicAnnotationList class. This cleaned up the code in RDomBase and will make it easier to evolve the PublicAnnotationList.

# Removed RDomSyntaxNodeBase from hierarchy

At one point this class seemed appropriate in the hierarchy. It wasn’t doing anything and was removed.

# NonEmptyNamespaces renamed to NonemptyNamespaces

Cleanup issue found by FxCop.

# Improved code analysis (FxCop) and test coverage

I may separately blog about how positive the code analysis exercise was – in spite of my deep dread of what I would find. The recommended rules had only one issue – which I thought was pretty cool. Switching to All Microsoft Rules for the non-testing libraries resulted in about 100 issues. I dropped this to under 25 and almost all the changes were things I was really happy to find – insufficient checks for nulls on method entry, a couple of naming fixes.

Future Features

Helper methods:

* Find
* ReplaceInIds, probably include RegEx here