# [ This is the drafts of a series of blog posts, as the current best documentation ]

# RoslynDOM Release: aka, you are (I am) community

In April 2014, I gave an all day workshop on Roslyn metaprogramming. The morning was scheduled for metaprogramming basics, and the afternoon was Roslyn.

Everyone arrived after lunch all bright eyed and excited. I started explaining how to work with Roslyn, with frequent insertions of “I’m sure the community will develop libraries.” Crushed is the only word I can use for the change in demeanor on the faces of almost everyone in the room. Excitement went to “too hard for my team” and no amount of “the community will develop libraries” could compensate.

I taught a lot of Roslyn basics that day. But what kept rolling around in my head is that some of the most excited programmers about Roslyn, folks that had paid for my workshop, arrived at a “too hard for my team” conclusion.

And “the community will develop libraries” kept rolling around in my head. “I am community” is comfortable for me when it involves speaking, writing, lobbying for features and training. I planned to release the research I’m doing. But a general purpose Roslyn library based on my beliefs of what we need? Well, I’m still not sure how much hubris is involved, but I think it’s a cliff I need to dive off.

Instead of moving headlong to releasing my new stuff, I stepped sideways and began yanking my utilities out into a separate library and adding a bunch more unit tests on the library itself. Life intervened as well, so it’s been more weeks than I hoped for, but I’m releasing the first alpha of my library on NuGet (search for RoslynDOM) and the code’s available here on GitHub.

I really, really need your feedback, please send it and I’ll try to keep up.

I’ll follow up with the code first support library I showed in the workshop (the release was delayed to split out the library) and hopefully the new templates, assuming no brick walls appear in that research.

Thanks to the folks in my workshop for inspiring me.

## Why Roslyn is hard

## A couple of solutions

## Mentors, helpers and friends

Twice before I’ve released something to the wild, and it died.

I understand a lot more about Open Source than I did then, most particularly, I now know that I don’t know much about the mechanics of OSS.

# RoslynDOM Goals

* Hide Roslyn roundtrip details (attributes, fields, namespace nesting)
* Easily load into a simplified code tree
* Hide which features are easily accessed from syntactic and which semantic tree
* Retain access to full Roslyn details, by providing direct access to syntax node and symbol
* Provide generalized interface that is platform/compiler/underlying tree agnostic (IOW, it would work for Java and for some things JavaScript)

# RoslynDOM Features

# RoslynDOM Samples

# RoslynDOM Outstanding Questions

## What do you expect when you say something stupid?

public abstract static class

In cases like this, I can’t tell which you actually meant, so I’m returning the stupid answer

But, I’m somewhat inconsistent. If the keyword is never valid on the feature (an abstract field for example), I don’t let you ask the question.

Also, Roslyn is somewhat inconsistent – returning true for abstract and static above, but returning only the first variance (out) in:

public class Foo3< out in T2>

## What is the name of a nested class?

Consistent with the Roslyn library

## What is the name of a dotted namespace?

Consistent with the Roslyn library

# User Stories

## You have an individual file you want to understand in isolation

Example: CodeFirst metdata usage

## You want to explore a project or solution

## You want to modify the syntax tree (source code)