TRINUG

APPLIEDINFORMATIONSCIENCES

Robert Half Technology

INFRAGISTICS
DESIGN / DEVELOP / EXPERIENCE

2013 Raleigh Code Camp

About me...

- Technology lead for a team at LexisNexis
- Blog: http://seanmccarthy.me
- Twitter: mccartsc
- G+: scmccart

About this talk...

Slides and Code at, https://github.com/scmccart/MetaprogrammingEssentialsDemo

Or,

http://bit.ly/1go2IsO

Metaprogramming, huh?

- Writing code that can write code.
- Metaprogramming uses a Metalanguage
  - S-Expressions, Attributes, Conventions
- Wide support, in languages such as ruby, python, javascript, C++, lisp, template haskell
- Compile time vs Runtime

Metalanguages in .Net

- S-Expressions?
  - Not available, unless you are using IronScheme or clojureclr. Kind of via Expression<T>
- Attributes
  - Available everywhere, widely understood.
- Conventions
  - Common, but custom.
  - Asp.Net's routing.
  - Entity Framework's code first method.
- T4 kind of counts, kind of.

Metaprogramming methods

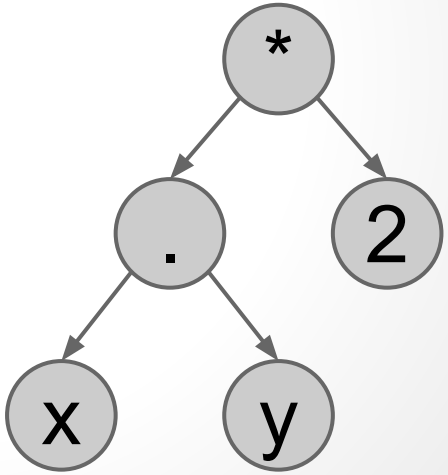
- Reflection.Emit
  - Powerful, can build anything.
  - Very manual, hard to use, error prone
- System.Linq.Expressions
  - AKA Expression Trees
  - Easyish API, type checks
  - Can only do lambdas, not as fine grained
- Roslyn
  - Compiler as a service
  - Not released yet

Expression Trees

Expression Trees map very closely to an AST

An example, x => x.y * 2

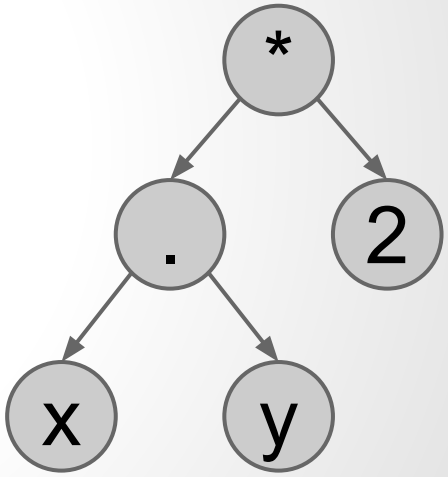
As a tree, would be:



Expression Trees

Trees maps to the Expression builder's use.

```
Expression.Multiply(
    Expression.Property(
        x,
        "y"),
    Expression.Constant(2))
```



Using Expression Trees

Repeatable flow with the Expression builder,

1. Declare parameters
2. Build out body
3. Pull together in lambda
4. Compile
5. Use!

1. Declare Parameters

```
var value = Expression.Parameter(typeof(int), "value");
```

2. Build out body

```
var value = Expression.Parameter(typeof(int), "value");
var body = Expression.Multiply(value, Expression.Constant(2));
```

3. Pull together in Lambda

```
var value = Expression.Parameter(typeof(int), "value");
var body = Expression.Multiply(value, Expression.Constant(2));
var lambda = Expression.Lambda<Func<int, int>>(body, value);
```

4. Compile

```
var value = Expression.Parameter(typeof(int), "value");
var body = Expression.Multiply(value, Expression.Constant(2));
var lambda = Expression.Lambda<Func<int, int>>(body, value);
var compiled = lambda.Compile();
```

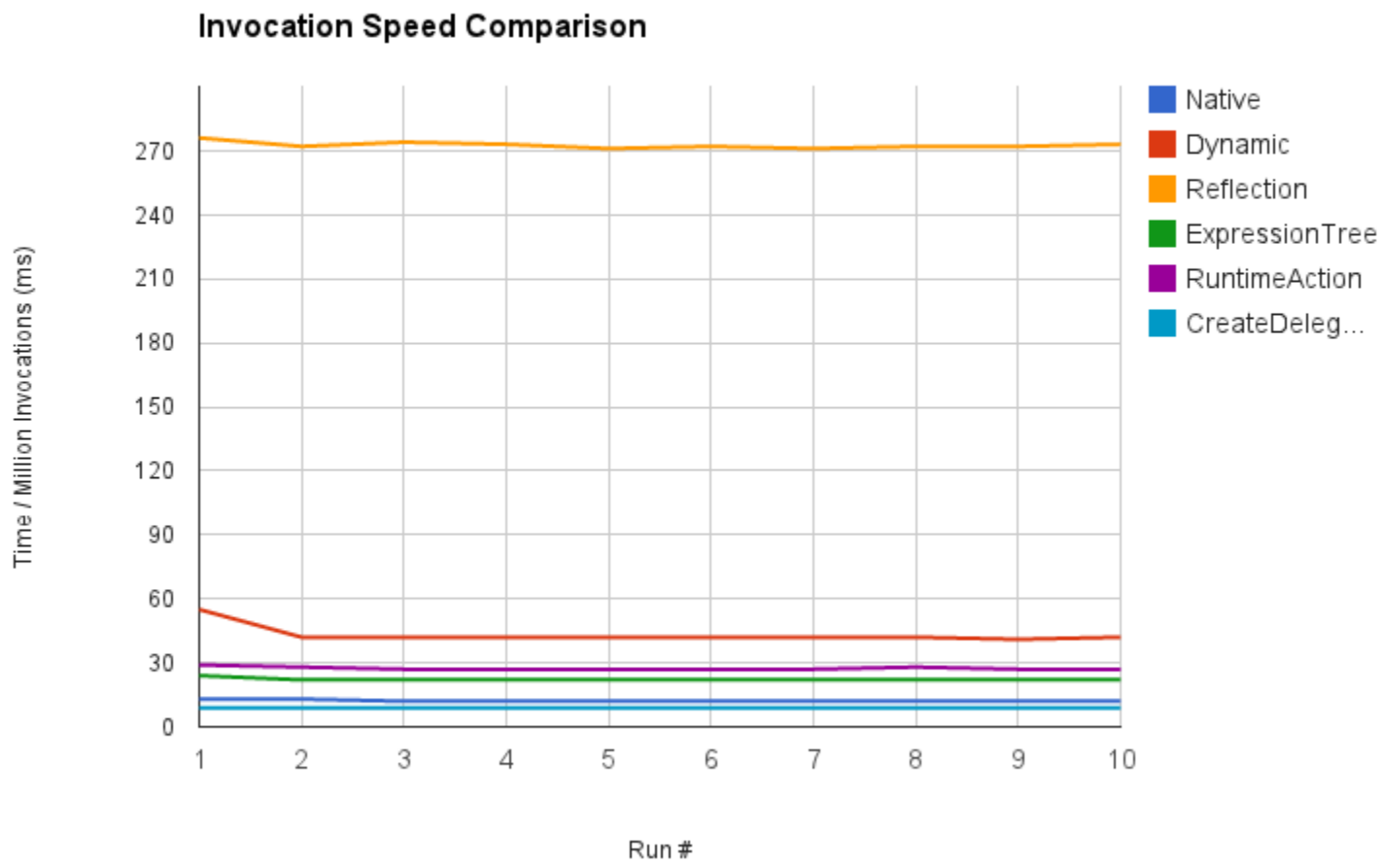
5. Use!

```
var value = Expression.Parameter(typeof(int), "value");
var body = Expression.Multiply(value, Expression.Constant(2));
var lambda = Expression.Lambda<Func<int, int>>(body, value);
var compiled = lambda.Compile();

Console.Write("input: ");
var input = int.Parse(Console.ReadLine());
var output = compiled(input);
Console.WriteLine("output: " + output);
```

Let’s see that in action.

Speed


Invocation Speed Comparison
Time / Million Invocations (ms)
270
240
210
180
150
120
90
60
30
0
1
2
3
4
5
6
7
8
9
10
Run #
Native
Dynamic
Reflection
ExpressionTree
RuntimeAction
CreateDeleg...

Quick Tips

- DebugView
- ToString
- Cache compiled trees
- Limits
  - Async/Await
  - Yield Return

Object Routing Example

- Routing for objects to eliminate nested switch statements.
- Uses Expressions for
  - Getting property values
  - Invoking methods
  - Ternary expressions
  - Constructing Arrays