2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Dynamisch und Gefährlich?

C# Dynamics in freier Wildbahn

Tim Bourguignon

Mathema Software GmbH

v1.1

Dynamics? Nooooooooo....

Dynamics? Nooooooooo....

Dynamics? Nooooooooo....

Dynamics? Noooooooo....

Compiler says *meep*

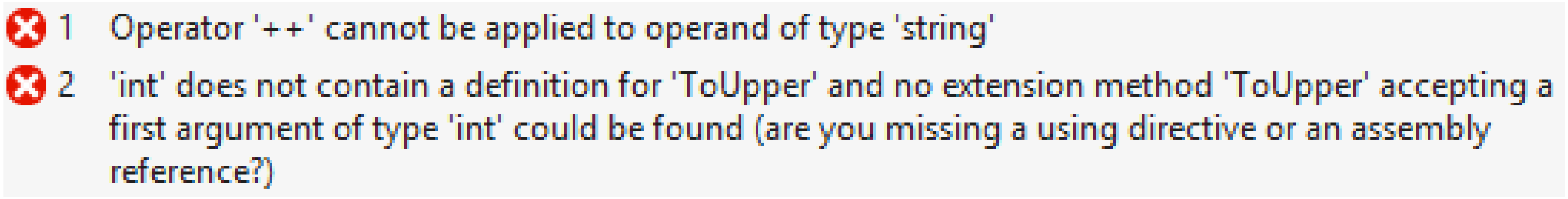
```
string lang = "C#";
lang++;

int theAnswer = 42;
theAnswer.ToUpper();
```

Compiler says *meep*

```
string lang = "C#";
lang++;

int theAnswer = 42;
theAnswer.ToUpper();
```



1 Operator '++' cannot be applied to operand of type 'string'

2 'int' does not contain a definition for 'ToUpper' and no extension method 'ToUpper' accepting a first argument of type 'int' could be found (are you missing a using directive or an assembly reference?)

Dynamics to the rescue

```
dynamic lang = "C#";
lang++;

dynamic theAnswer = 42;
theAnswer.ToUpper();
```

Dynamics to the rescue

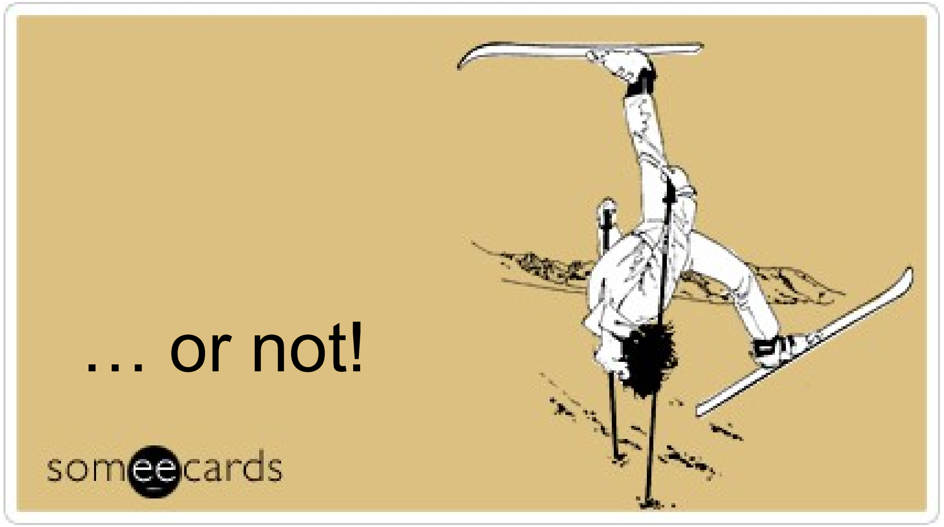
```
dynamic lang = "C#";
lang++;

dynamic theAnswer = 42;
theAnswer.ToUpper();
```

Dynamics to the rescue

```
dynamic lang = "C#";
lang++;

dynamic theAnswer = 42;
theAnswer.ToUpper();
```



What about 'object' or reflection?

```
Calculator calc = new Calculator();
int sum = calc.Add(10, 20);
```

What about 'object' or reflection?

```
Calculator calc = new Calculator();
int sum = calc.Add(10, 20);
```

```
object calc = new Calculator();
int sum = calc.Add(10, 20);
```

What about 'object' or reflection?

```
Calculator calc = new Calculator();
int sum = calc.Add(10, 20);
```

```
object calc = new Calculator();
int sum = calc.Add(10, 20);
```

What about 'object' or reflection?

```
Calculator calc = new Calculator();
int sum = calc.Add(10, 20);
```

```
object calc = new Calculator();
int sum = calc.Add(10, 20);
```

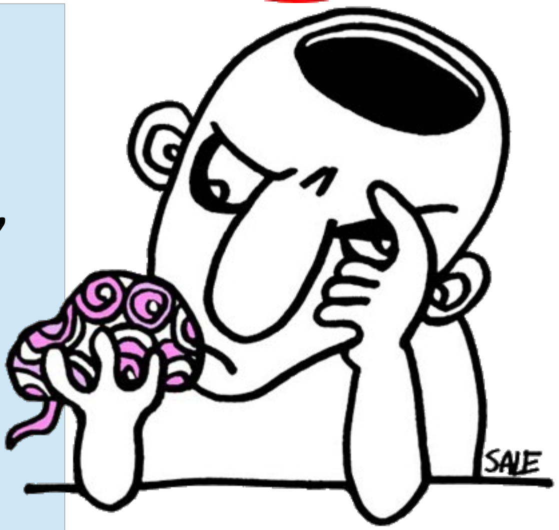
```
object reflectionCalc = new Calculator();
Type calcType = reflectionCalc.GetType();
object result = calcType.InvokeMember("Add",
     BindingFlags.InvokeMethod, null,
     Activator.CreateInstance(calcType),
      new object[] { 10, 20 });
int sum2 = Convert.ToInt32(result);
```

What about 'object' or reflection?

```
Calculator calc = new Calculator();
int sum = calc.Add(10, 20);
```

```
object calc = new Calculator();
int sum = calc.Add(10, 20);
```

```
object reflectionCalc = new Calculator();
Type calcType = reflectionCalc.GetType();
object result = calcType.InvokeMember("Add",
     BindingFlags.InvokeMethod, null,
     Activator.CreateInstance(calcType),
      new object[] { 10, 20 });
int sum2 = Convert.ToInt32(result);
```

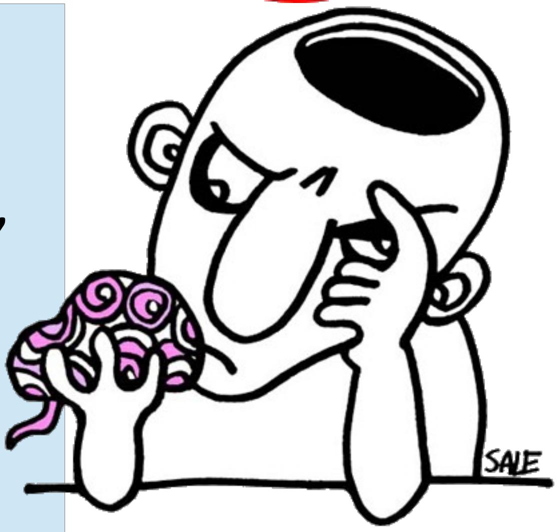


What about 'object' or reflection?

```
Calculator calc = new Calculator();
int sum = calc.Add(10, 20);
```

```
object calc = new Calculator();
int sum = calc.Add(10, 20);
```

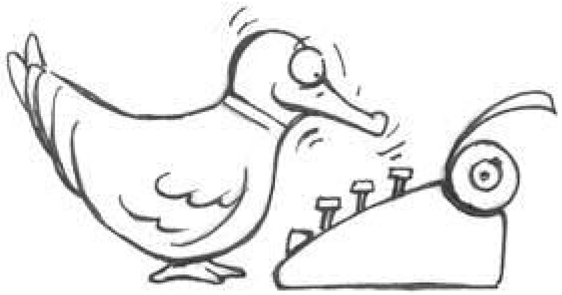
```
object reflectionCalc = new Calculator();
Type calcType = reflectionCalc.GetType();
object result = calcType.InvokeMember("Add",
     BindingFlags.InvokeMethod, null,
     Activator.CreateInstance(calcType),
      new object[] { 10, 20 });
int sum2 = Convert.ToInt32(result);
```



```
dynamic calc = new Calculator();
int sum = calc.Add(10, 20);
```

Duck-Typing

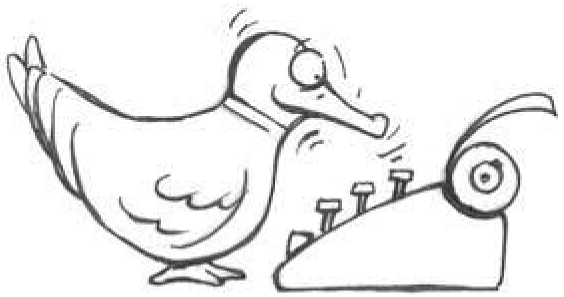
Duck-Typing



Duck-Typing

- When I see a bird that
  walks like a duck
  swims like a duck
  and quacks like a duck,
  I call that bird a duck

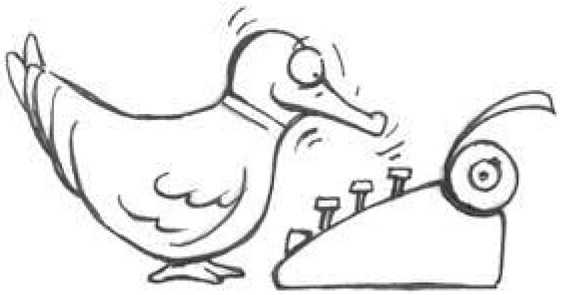
  James Whitcomb Riley



Duck-Typing

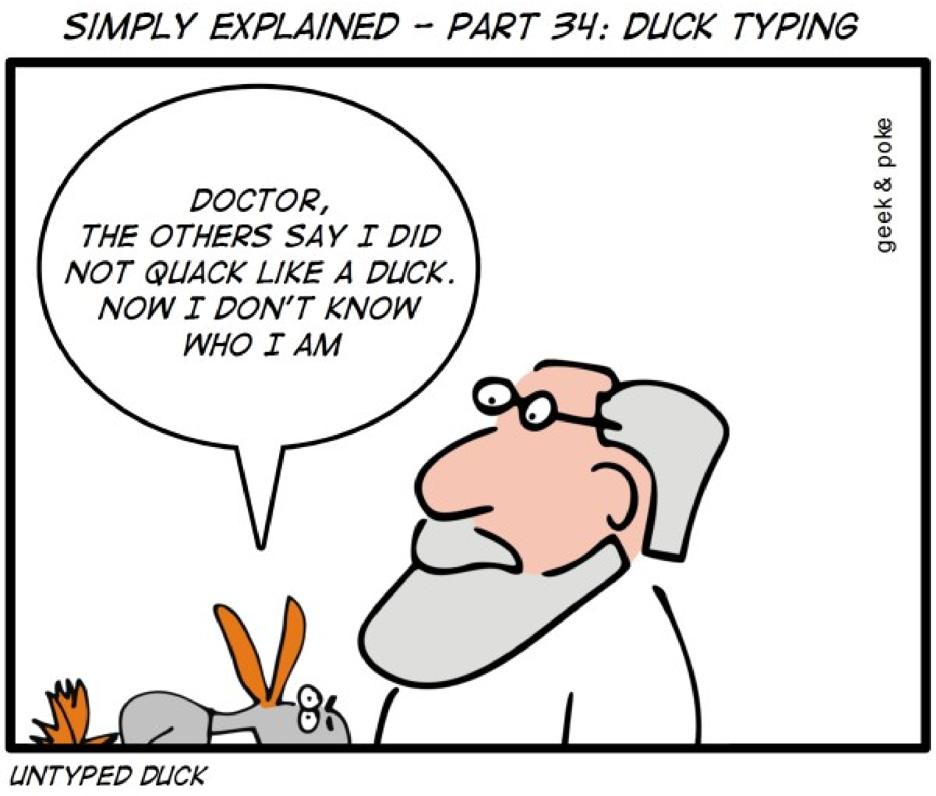
- When I see a bird that
  walks like a duck
  swims like a duck
  and quacks like a duck,
  I call that bird a duck

  James Whitcomb Riley



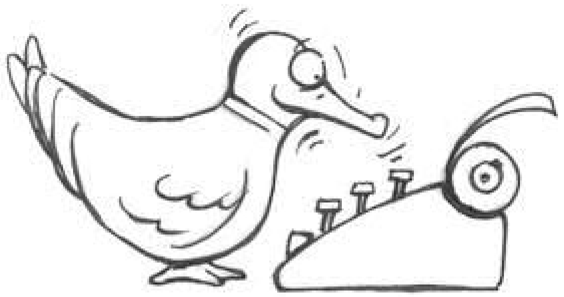
- Look like vs Be
- Methods & Attributes vs Class

Duck-Typing


SIMPLY EXPLAINED - PART 34: DUCK TYPING
DOCTOR, THE OTHERS SAY I DID NOT QUACK LIKE A DUCK. NOW I DON'T KNOW WHO I AM
geek & poke
UNTYPED DUCK

- When I see a bird that
  walks like a duck
  swims like a duck
  and quacks like a duck,
  I call that bird a duck

  James Whitcomb Riley



- Look like vs Be
- Methods & Attributes vs Class

Dynamic languages: IronPython - IronRuby

Dynamic languages: IronPython - IronRuby

```
#Python script.py
def add(a, b):
    return a + b
```

```
var pythonRuntime = Python.CreateRuntime();
dynamic pythonScript =
     pythonRuntime.UseFile("script.py");
var result = pythonScript.add(100, 200)));
```

Base objects & Tools

Base objects & Tools

DynamicObject

Base objects & Tools

DynamicObject

ExpandoObject

Base objects & Tools

DynamicObject

ExpandoObject

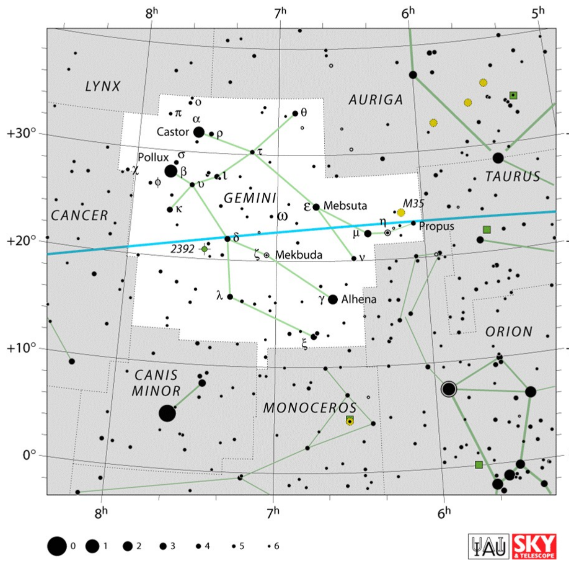
ElasticObject

Base objects & Tools

DynamicObject

ExpandoObject

ElasticObject



Gemini

Frameworks

Frameworks

Massive

Frameworks

Massive

Nancy

Frameworks

Massive

Nancy

Simple.Data

System.Dynamic.DynamicObject

- Exposes members at run time instead of at compile time
- Important methods
  - TrySetMember
  - TryGetMember
    - Is called when a member of a dynamic class is requested and no arguments are specified
  - TryInvokeMember
    - Is called when a member of a dynamic class is requested with arguments
- Combining those functions in a smart way is the key

System.Dynamic.ExpandoObject

- Represents an object whose members can be dynamically added and removed at run time
- Demo
  - Simple ExpandoObject
  - Expando structure vs Xml structure
  - ExpandoToXml
  - Linq-to-Object

http://blogs.msdn.com/b/csharpfaq/archive/2009/10/01/dynamic-in-c-4-0-introducing-the-expandoobject.aspx

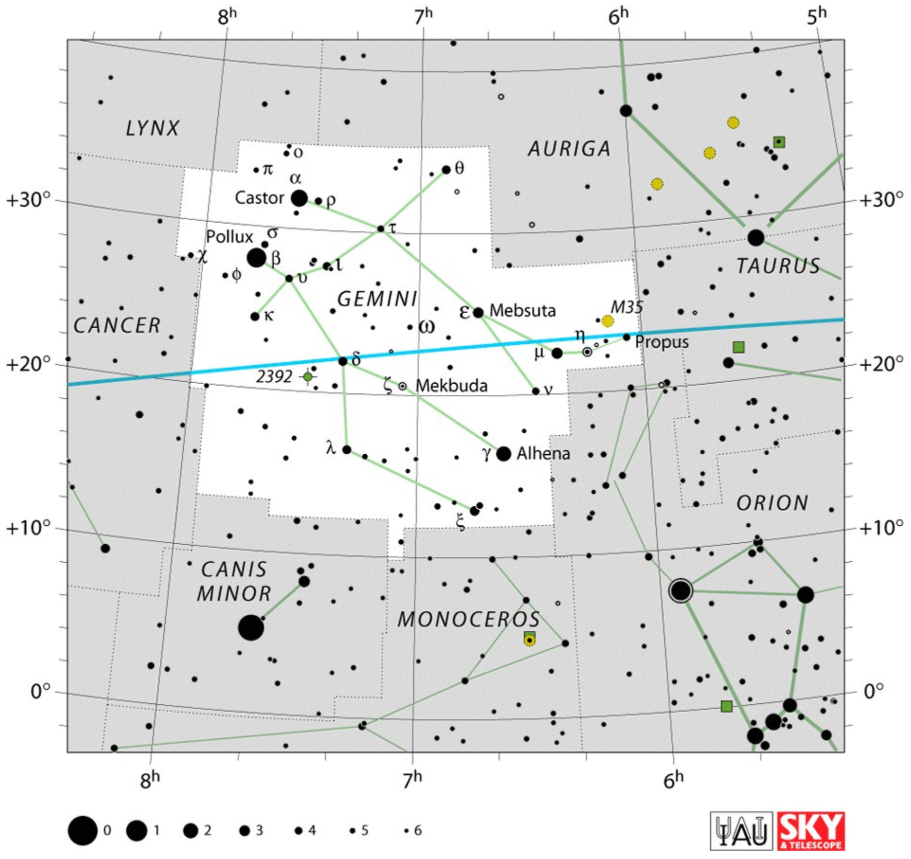
ElasticObject

- Multi level dynamic object implementation using .NET 4.0 dynamic features, for fluent access of data types like XML
- Demo
  - Expando vs Elastic
  - Elastic-to-Xml

https://github.com/amazedsaint/ElasticObject

Gemini

- „Brings the capabilities of a dynamic type system to C#“
- Demo
  - Members on the fly
  - Methods on the fly
  - Object graph
  - Responds to
  - Introspection



NancyFx

- Lightweight WebFramework
- Demo
  - Parameters
  - Return object

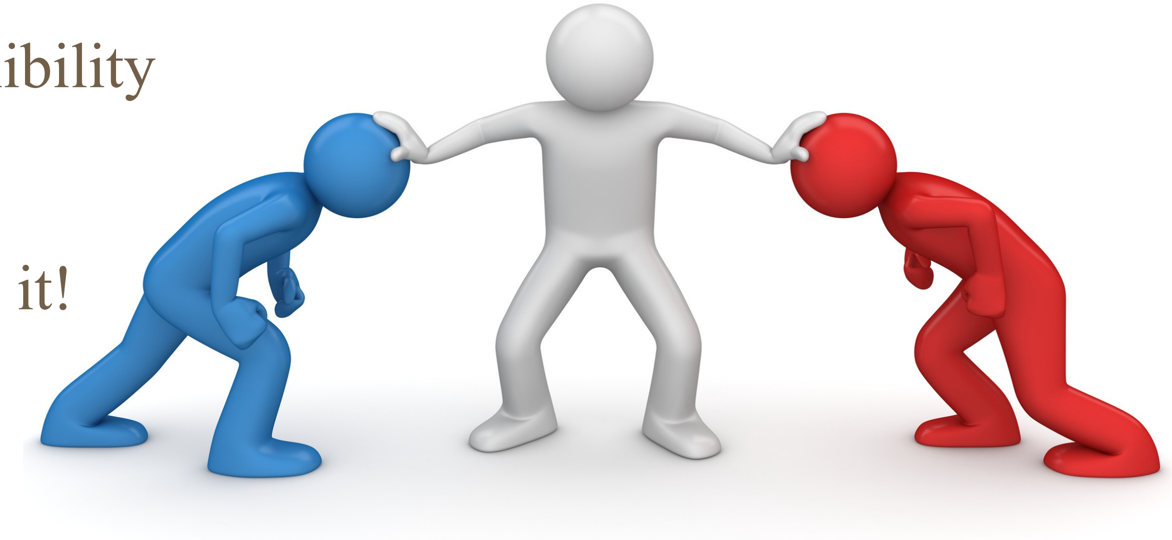
Massive

- Wrapper for DB tables that uses dynamics
- Create a class that wraps a table
- Query away
- Demo
  - Usage
  - Definition of TryGetMember

Conclusion

- Objects
  - Core Objects: DynamicObject, ExpandoObject
  - Variations: ElasticObject, Gemini
  - Usages: NancyFx, Massive, Simple.Data
- DTOs
- Architectural Flexibility
- API Design

- Think about using it!



Ich freue mich auf Eure Fragen!

tim.bourguignon@mathema.de
about.me/timbourguignon
@timothep