Lost in Translation
Reguläre Ausdrücke als Englische Sätze

Tim Bourguignon

tim.bourguignon@mathema.de
www.mathema.de

The menu

- Foreword
- * Expression
  - VerbalExpression
  - SimpleExpression
  - MagicExpression
- Wrap up

Image: internationalpointofsale.com

TL; DR;

- Why express yourself like this?

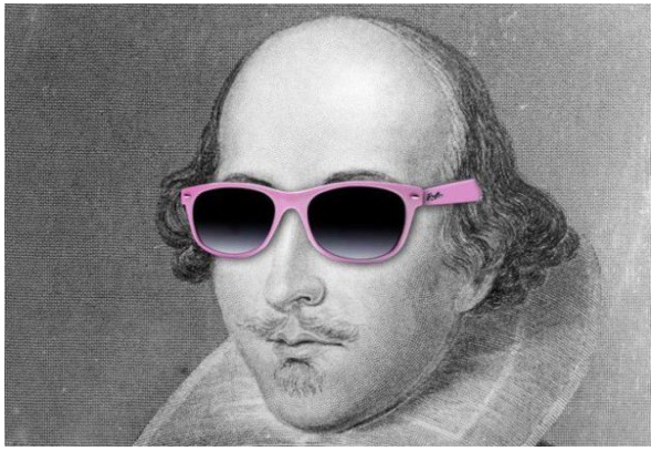
```
[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}
```

- When you can say it like Shakespeare?

"Thou shall match a string of letters
follow'd by @
then some other characters
a dram dot
and some moo stuff"

- A more coder friendly version maybe?

```
Thou.shallmatch(a-string-of-letters)
      .followdby("@")
      .then.someothercharacters()
      .adram(".")
      .and.some(moostuff);
```



A regular expression DSL?

- "SimpleExpression"
  - Syntax close to the English language
  - Build as a fluent API
  - Tailored for newbie's
    - Could it satisfy veterans too?
  - Outputs regular expressions

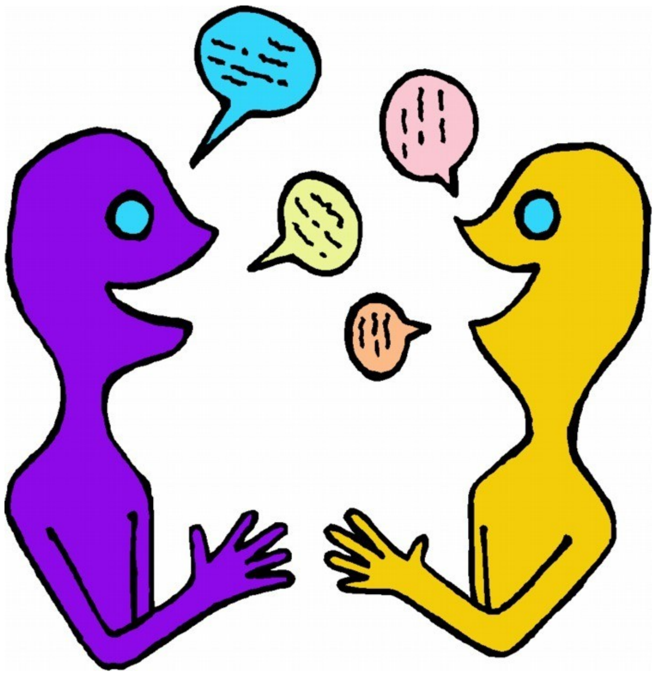


Image: scilogs.com

Why… Why… WHY?

- Example for a “C# dynamics” talk
- Write a real DSL (at least) once
- See if it works…
- Regular Expression knowledge refresh

- ~~Get rich and famous (bitches!)~~

- **Because I can!**

Why? ~~Two~~ Three Problems

- “Some people, when confronted with a problem, think:
  ‘I know, I'll use regular expressions.'
  Now they have two problems”

  - “Some people, when confronted with a problem, think:
    ‘I know, I'll create a DSL that wraps regular expressions.'
    Now they have three problems”

MATHEMA campus 2014

Why? What is there to lose?

(© Franck Mée, a “friend” who ~~likes~~ loves regular expressions)

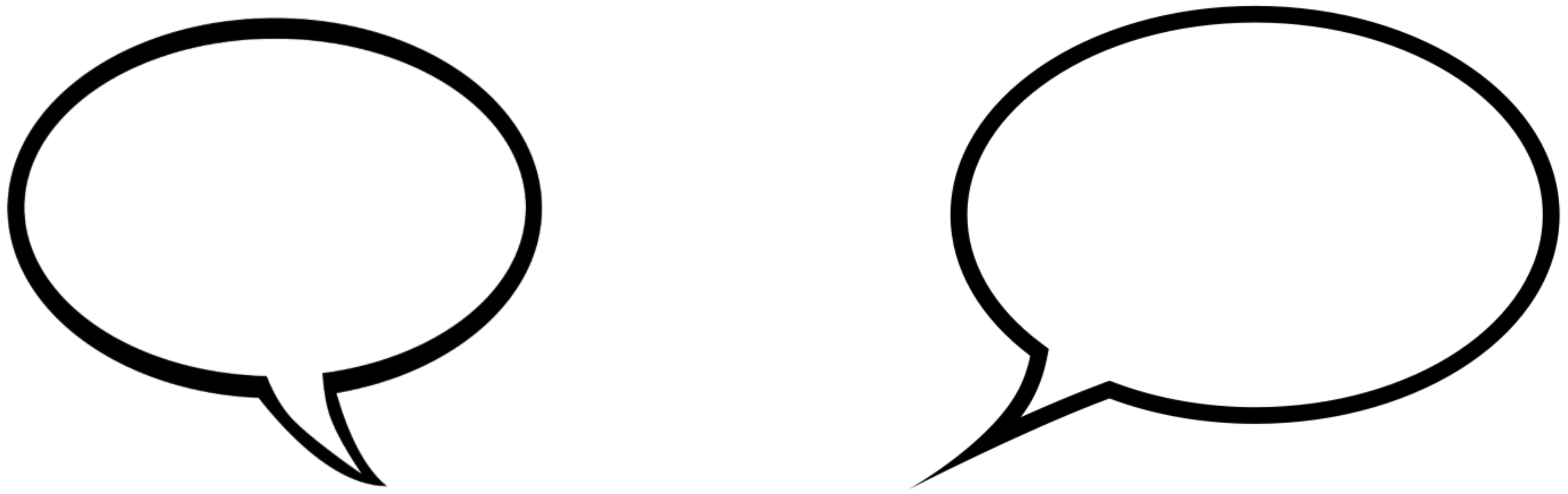
- The more complex the expression, the more surprised and god-like you’ll feel when it works
- When you write one that works and you know no-one will ever understand, feel like Houdini mystifying everyone
- You can strip down someone’s regex to pieces and yet never figure it out. Which makes you feel like looking at Houdini and God’s work combined
- Old regexes (of yours) are like teenage kids, you know they came out of you, but you don’t quite get them anymore

What we will NOT talk about

- How to write regular expressions
- Do’s and don’ts working with Regexes
- Optimization & performance
- Devise on the RFC822 Email Regex

```
[illegible]
```

?
==



VerbalExpression

Prior-Art? Yes, “VerbalExpressions”

- “JavaScript Regular expressions made easy”
  - On github (github.com/VerbalExpressions)
  - Forks for: Ruby, C#, Python, Java, Groovy, PHP, Haskell, C++ and Objective-C

```
var tester = VerEx()
        .startOfLine()
        .then( "http" )
        .maybe( "s" )
        .then( "://" )
        .maybe( "www." )
        .anythingBut( " " )
        .endOfLine();
```

NotSoClearExpressions…

- Inconsistent “Find()” function

```
var expression = VerEx()
        .find( "http" )
        .maybe( "s" )
        .then( "://" )
        .or()
        .then( "ftp://" )
```

NotSoClearExpressions…

- Branching

Could you please get me a burger and fries or a pizza?

```
var expression = VerEx()
        .find( "http" ).maybe( "s" ).then( "://" )
        .or()
        .then( "ftp://" )
```

?

```
var expression = VerEx()
        .find( "http" )
       .maybe( "s" )
       .then( "://" )
```

```
var expression = VerEx()
        .find( "http" )
       .maybe( "s" )
       .then( "ftp://" )
```

NotSoClearExpressions…

Can you tell what this VerbalExpression does?

```
VerEx().then( "." ).replace( my_paragraph, ". Stop." );
```

- Is this intuitive?
- Why not this?

```
VerEx().find( "." ).in( my_paragraph).replaceWith(". Stop." );
```

SimpleExpression

SMILEYS

Emoticons: brusheezy.com/members/tijo

How to manipulate SimpleExpressions?

- Here’s how you use a SimpleExpression

```
dynamic simpex = new SimpleExpression();

simpex.here.I.can.chain.my.commands
                .Generate();

Console.Write(simpex.Expression);
```

- „dynamic“

```
dynamic someInt = 4;
someInt.ICanWriteHereWhateverIWantAndItCompiles(“doh”);
// … but will crash & burn in flames at runtime
```

Examples 1 & 2

- Floating point number matching

```
simpex
    .Maybe('-')
    .Numbers                        //Default is „zero or more“
    .One('.')
    .Numbers.AtLeast(1)
    .Generate();
```

- Hexadecimal Color

```
simpex
    .One('#')
    .Numbers.And(“abcdef“).Exactly(3)
        .Or
    .Numbers.And(“abcdef“).Exactly(6)
    .Generate();
```

Example 3

- Email validation

```
string allowedChars = @"!#$%&'*+/=?^_`{|}~-";

Simpex
    .Group
        .Alphanumerics.And(allowedChars).AtLeast(1)
    .Together.As("beforeAt")
    .One('@')
    .Group
        .Letters.And(allowedChars).AtLeast(1)
        .Group
            .One(".")
            .Alphanumerics.And(allowedChars).AtLeast(1)
        .Together.As("dotAndAfter")
    .Together.As("afterAt")
    .Generate();
```

Critics: Syntax Hunting Season

- Isn’t the following gorgeous to read? (the answer is YES ;)

```
simpex.Letters.AtLeast(3).AtMost(4)
simpex.Text("abcd").Except("a")
simpex.Group.Text("aeiouy").Together.As("SomeLetters")
```

- What about the following?

```
simpex.Group.Text("aeiouy").Together.Exactly(2)
```

  - You wanted to say ‘twice’ instead of ‘two’, didn’t you?

Critics: Repetition Mess

- What does the following mean?

```
Simpex
.Group.AtLeast(4).AtMost(5).Numbers.Exactly(2)
.Together.One(' ').AtMost(2)
```

  - *“4 to 5 groups of 2 numbers followed by at most 2 spaces“*
  - *“group at least 4 and at most 5 numbers, twice”*
- The problem here:

```
Group.AtMost.X.Exactly.Together.Y.AtMost
```

  - And no, pushing it after the ‘together’ wouldn’t solve the issue

```
Group.X.Exactly.Together.AtMost.Y.AtMost
```

  - How can we solve this?

MATHEMA campus 2014

Critics: Stuttering

- “Stuttering”, one of the limits of that prose

```
Simpex
    .Group
        .Group
            .Text(abcd)
            .Group
                .Letters.And("-")
            .Together
        .Together
        .Text("cde")
    .Together
```

SubExpressions, a sub solution

- Create now, join later

```
var abcd = new SimpleExpression().Text("abcd").Generate();
var efgh = new SimpleExpression().Text("efgh").Generate();

simpex.Sub(abcd).Or.Sub(efgh).Generate();
```

- That encapsulated grouping example

```
var innerMostGp = new SimpleExpression().
      Goup.Letters.And("-").Together.Generate();

var innerGp = new SimpleExpression()
.Group.Text(abcd).Sub(innerMostGp).Together.Generate();

var outerGp = new SimpleExpression()
.Group.Sub(innerGp).Text("cde").Together.Generate();
```

Critic: Implicit Cardinality

- What is meant here?

```
simpex.EitherOf("a|b|c").AtLeast(2)
```

  - *“a, b or c, at least two of them”*
  - *“twice a or twice b or twice c“*

  - How do I do I express the other one?

Critic: Experts, get lost!

- Regular expression “experts” cannot help it, they have to fall back onto what they know
- Does this create a class? A Group? Capturing or not?

```
simpex.Letters.Except("aeiou").And("§$%&").AtLeast(2).AtMost(4)
```

- The more you know what regular expressions can do, the more disturbing SimpleExpression is

Critic: dynamics Architecture

- Simple.Data (a C# model for this kind of architecture) has a true reason for using dynamics: the functions are not known at compile time
  - In SimpleExpression‘s case, everything is known beforehand
  - No reason for using dynamics :’(
  - Fully “implement-able” via a Fluent API
- No Intellisense support

MATHEMA campus 2014

Abstract Syntax Tree

- SimpleExpression commands cannot be linearly parsed
- Simple repeat count

```
Simpex.One(“x“).AtLeast(3).AtMost(“5”)

// x => x{3,} => x{3,5}
```

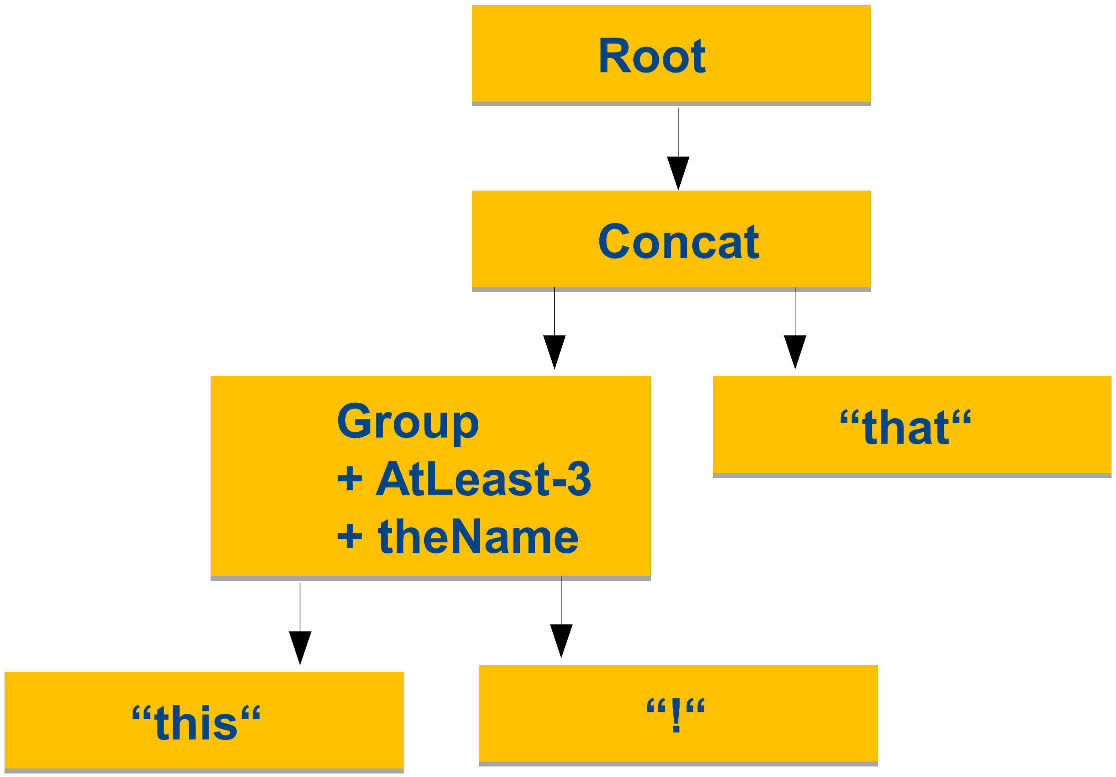
- Named groups and repeat count

```
simpex
    .Group.AtLeast(3).Text("something").Together.As("theName")

// (<theName>something){3,})
```

Abstract Syntax Tree

```
simpex
        .Group.AtLeast(3)
                .Text("this").One(“!“)
        .Together.As("theName")
        .Text(“that“)
        .Generate();
```



```
(<theName>this!){3,})that
```

MATHEMA campus 2014

What then?

- SimpleExpression’s semantic is really nice... but it is like *“death by 1000 paper cuts”*
- Many edge cases where the grammatic doesn't fit that well and tend to pull down the concept as a whole
- Could be changed using parenthesis again, reordering some elements in a logical way instead of a grammatical order, thus losing some readability for the sake of precision

MagicExpression

TL; DR;

- Evolutions compared to SimpleExpression
  - Loses the dynamic part of the SimpleExpression and comes back to a fluent API
  - Replaces parts of the commands with their less-funky but non-ambiguous functional equivalents
    - As a direct consequence, gets rid of the cumbersome Abstract Syntax Tree and thus of half of the complexity of SimpleExpression's implementation
  - Pushes the DSL way further than SimpleExpression was ever able to by adding many regular expression concepts to the equation

„MagicExpression for Muggles“ (©Ghusse)

- Install via Nuget

```
Install-Package MagicExpression
```

- Instanciation

```
var magicWand = Magex.New();

magicWand.The.Functions.Here;        //no lame .Generate() here

Console.WriteLine(magicWand.Expression);
```

Example 1: floating point match

```
var magicWand = Magex.New();

MagicWand
    .Character('-').Repeat.AtMostOnce()
    .CharacterIn(Characters.Numeral).Repeat.Any()
    .Character('.')
    .CharacterIn(Characters.Numeral).Repeat.AtLeastOnce();

// Creates the following regex:    -?[0-9]*\.[0-9]+
// Matches "1.234", "-1.234", "0.0", ".01"
// Doesn't Match "0", "1,234", "0x234", "#1a4f66"
```

- Character() & CharacterIn()
- .Repeat trigger
- Optional block handled via .AtMostOnce()
  - .Any() or .Between(0, uint) would also do the trick

Example 2: XML Tag Matching

```
var magicWand = Magex.New()
    .Character('<')
    .CaptureAs("tag", x =>
        x.CharacterNotIn('>').Repeat.AtLeastOnce())
    .Character('>')
    .Character().Repeat.Any().Lazy()
    .String("</")
    .BackReference("tag")
    .Character('>');

// Matches "<strong>hello world</strong>" & "<h1>A title</h1>"
// Doesn't match "<h1>A tag mismatch</strong>"
```

- Group() → Non-capturing group
- Capture() → Capturing group
- CaptureAs() → Named capturing group
- BackReference(string) → Back reference on a named group

Example 3: URL Matching

```
var magicWand = Magex.New();
magicWand.Options = RegexOptions.IgnoreCase;
const string allowedChars = @"!#$%&'*+/=?^_`{|}~-";

MagicWand
        .Alternative(
                Magex.New().String("http"),
                Magex.New().String("ftp"))
        .Character('s').Repeat.AtMostOnce()
        .String("://")
        .Group(Magex.New().String("www."))
                .Repeat.AtMostOnce()
        .CharacterIn(Characters.Alphanumeric, allowedChars);
```

- CharacterIn(params char[])

Example 4: ???

- What do these match?

```
Magex.New()
    .Character('#')
    .Alternative(
        Magex.New().CharacterIn(Characters.Numeral, "abcdefABCDEF")
            .Repeat.Times(6),
        Magex.New().CharacterIn(Characters.Numeral, "abcdefABCDEF")
            .Repeat.Times(3).EndOfLine());
```

```
Magex.New()
    .Character('0')
    .CharacterIn("xX")
    .CharacterIn(Characters.Numeral, "abcdefABCDEF").Repeat.Times(6);
```

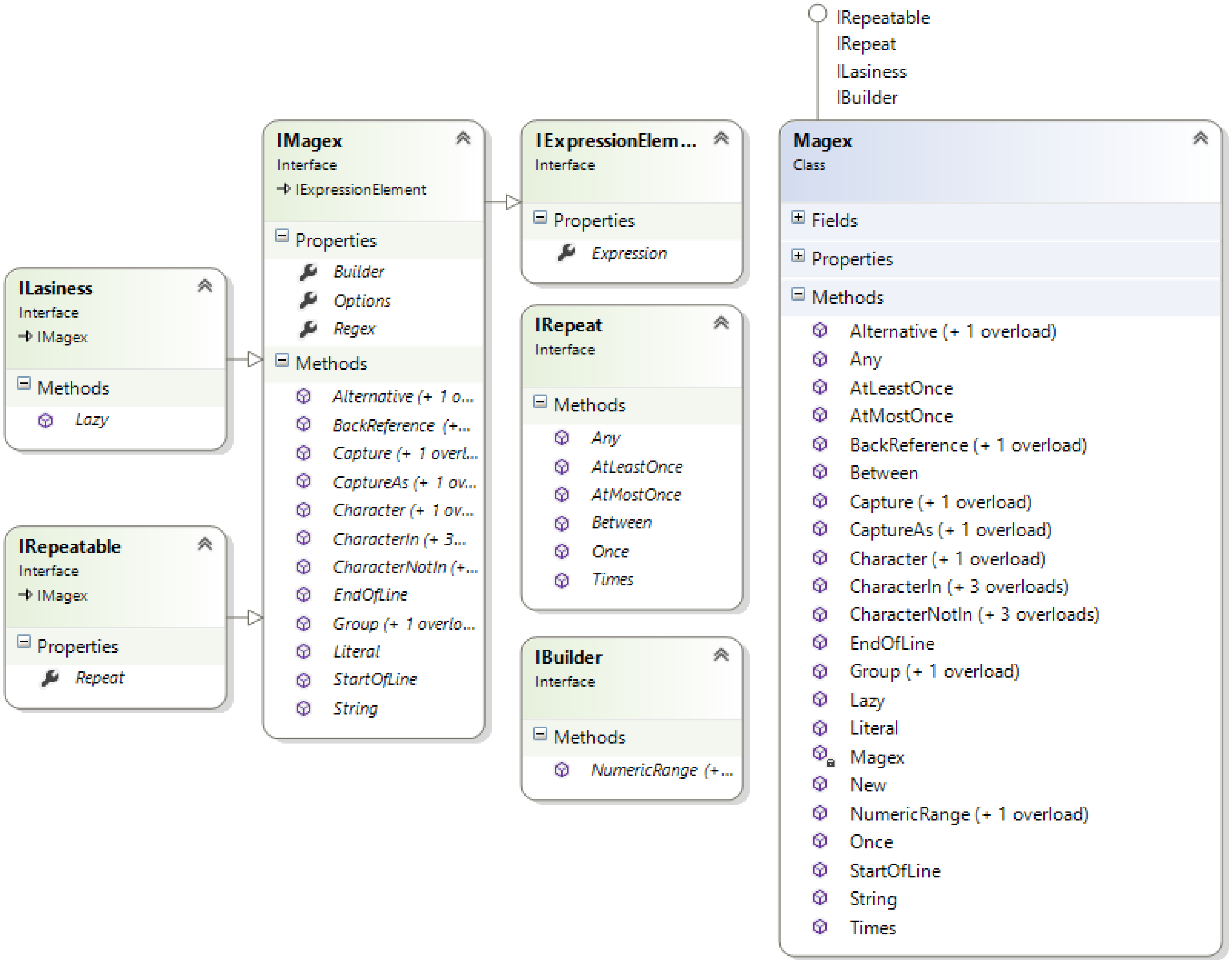
Example 5

- What does this match?

```
Magex.New()
    .Builder.NumericRange(1, 255).Character('.')
    .Builder.NumericRange(0, 255).Character('.')
    .Builder.NumericRange(0, 255).Character('.')
    .Builder.NumericRange(0, 255);
```

- Builder property to help you with predefined functions
  - Currently only NumericRange()
- Literal(string) function to add a predefined regular expression
- Other functions?
  - Email? Date with pseudo variable format → „yyyy-MM-dd“ ?
  - Hex, Floating point number... ? Any idea / wishes?

Architecture: Magex Interfaces

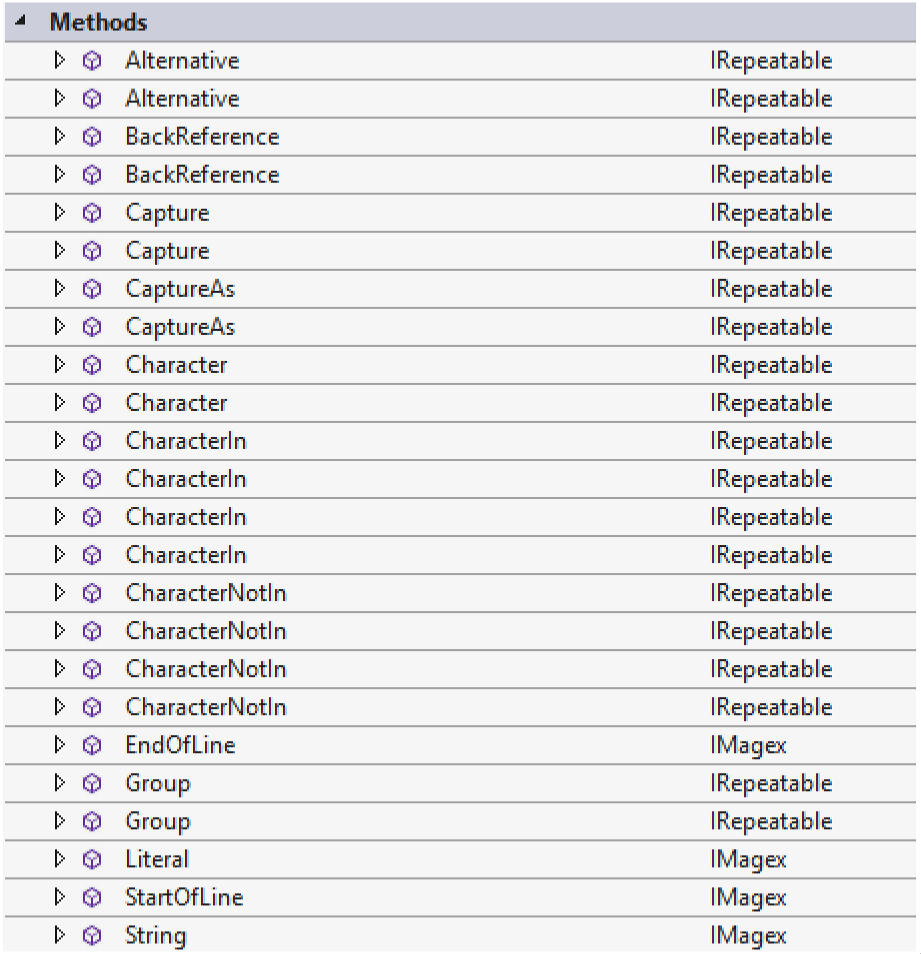


MATHEMA campus 2014

Architecture: Ghusse's Interfaces Game

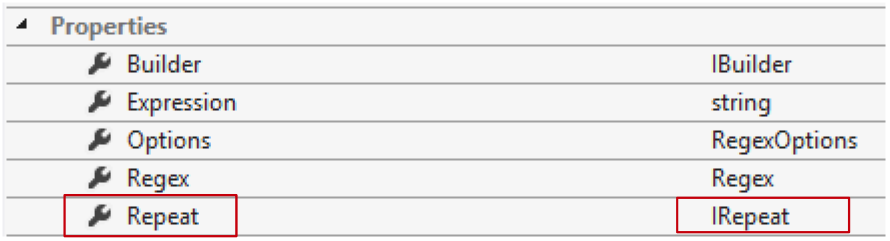
magex.Character('s').Repeat.AtMostOnce().Character...

IMagex



Methods	
Alternative	IRepeatable
Alternative	IRepeatable
BackReference	IRepeatable
BackReference	IRepeatable
Capture	IRepeatable
Capture	IRepeatable
CaptureAs	IRepeatable
CaptureAs	IRepeatable
Character	IRepeatable
Character	IRepeatable
CharacterIn	IRepeatable
CharacterIn	IRepeatable
CharacterIn	IRepeatable
CharacterIn	IRepeatable
CharacterNotIn	IRepeatable
CharacterNotIn	IRepeatable
CharacterNotIn	IRepeatable
CharacterNotIn	IRepeatable
EndOfLine	IMagex
Group	IRepeatable
Group	IRepeatable
Literal	IMagex
StartOfLine	IMagex
String	IMagex

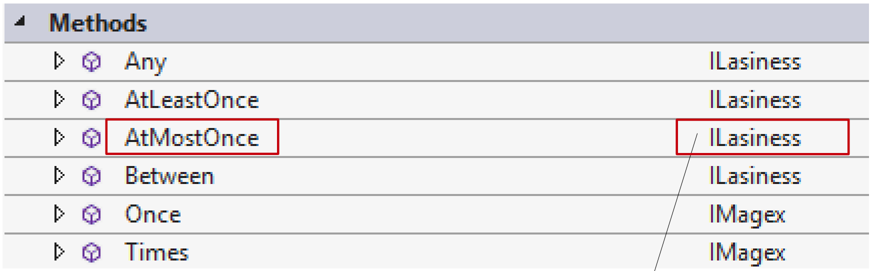
IMagex



Properties	
Builder	IBuilder
Expression	string
Options	RegexOptions
Regex	Regex
Repeat	IRepeat

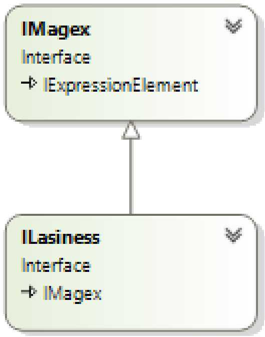
Only those methods are available after .Repeat

IRepeat



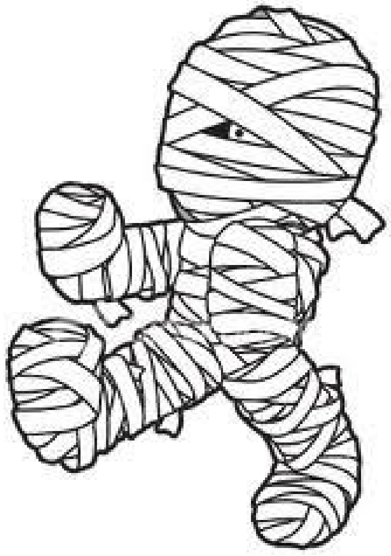
Methods	
Any	ILasiness
AtLeastOnce	ILasiness
AtMostOnce	ILasiness
Between	ILasiness
Once	IMagex
Times	IMagex

IMagex gives us access to all those Methods...



Gives an ILasiness Back, e.g. an IMagex

Let's wrap up



MATHEMA campus 2014

Wrap Up

- It is possible to write such a DSL!
  - ~~I'm going to be rich and famous~~
  - Our languages are not always a good thing to immitate
  - But (in this case) a pinch of DSL doesn't hurt
- SimpleExpression
  - Semantically (very?) attractive, but not viable as is
  - Early retirement?
- MagicExpression
  - Semantically less sexy, but architecturally gorgeous and easier to add features to
  - Next big feature → Reverse engineer regular expressions?

Fragen?

THANKS!

QUESTIONS?
SUGGESTIONS?
IDEAS?

- www.timbourguignon.fr
- tim.bourguignon@mathema.de