C# 6.0 und Roslyn

Was bringt die Zukunft von .NET?

.NET User Group Oldenburg

- 2007 gegründet von Dr. Yvette Teiken und Hilmar Bunjes
- Ziel: Zusammenschluss von begeisterten .NET Entwicklern in Oldenburg und Austausch mit Interessierten
- Damals Uni Oldenburg und OFFIS Sponsor
- Mangels Unterstützer Pause bis heute -> Das ändern wir!

erminas

.NET Oldenburg user group

.NET User Group Oldenburg

- Jetzige Sponsoren

sitrion erminas

- Treffen: 4. Donnerstag im Monat
- Ort: Sitrion (also hier)
- Einzugsbereich: Metropolregion Bremen / Oldenburg

erminas .NET Oldenburg user group

.NET User Group Oldenburg

- Geplante Vorträge
  - 28.8.: Big Data und Small Data – Hadoop und Microsoft (Dr. Yvette Teiken)
  - 25.9.: tbd (Dr. Jonas Jacobi)
  - 23.10.: Perceptual Computing / Natural User Interaction (Lars Keller)
- Neue jederzeit Herzlich Willkommen

erminas

.NET Oldenburg user group

C# 6.0 und Roslyn

- Was ist Roslyn?
- Was bringt C# 6.0 Neues?
- Muss ich das wissen und nutzen?
- Wie kann ich das nutzen?

erminas

.NET Oldenburg user group

Was ist Roslyn?

erminas

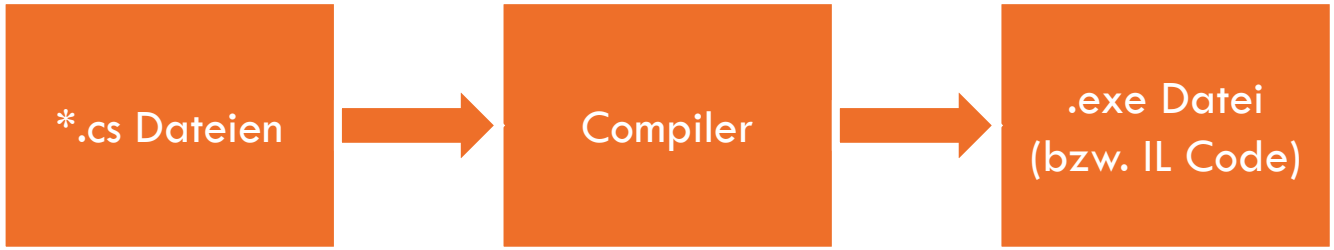
.NET Oldenburg
user group

Was ist Roslyn noch?

- Eine Compiler-Infrastruktur in .NET
- C# Compiler in C# und VB.NET Compiler in VB.NET (F# ist außen vor)
- Compiler-as-a-Service

C# Compiler bisher

- Monolithischer Aufruf, am Ende kommt bspw. eine .exe Datei raus



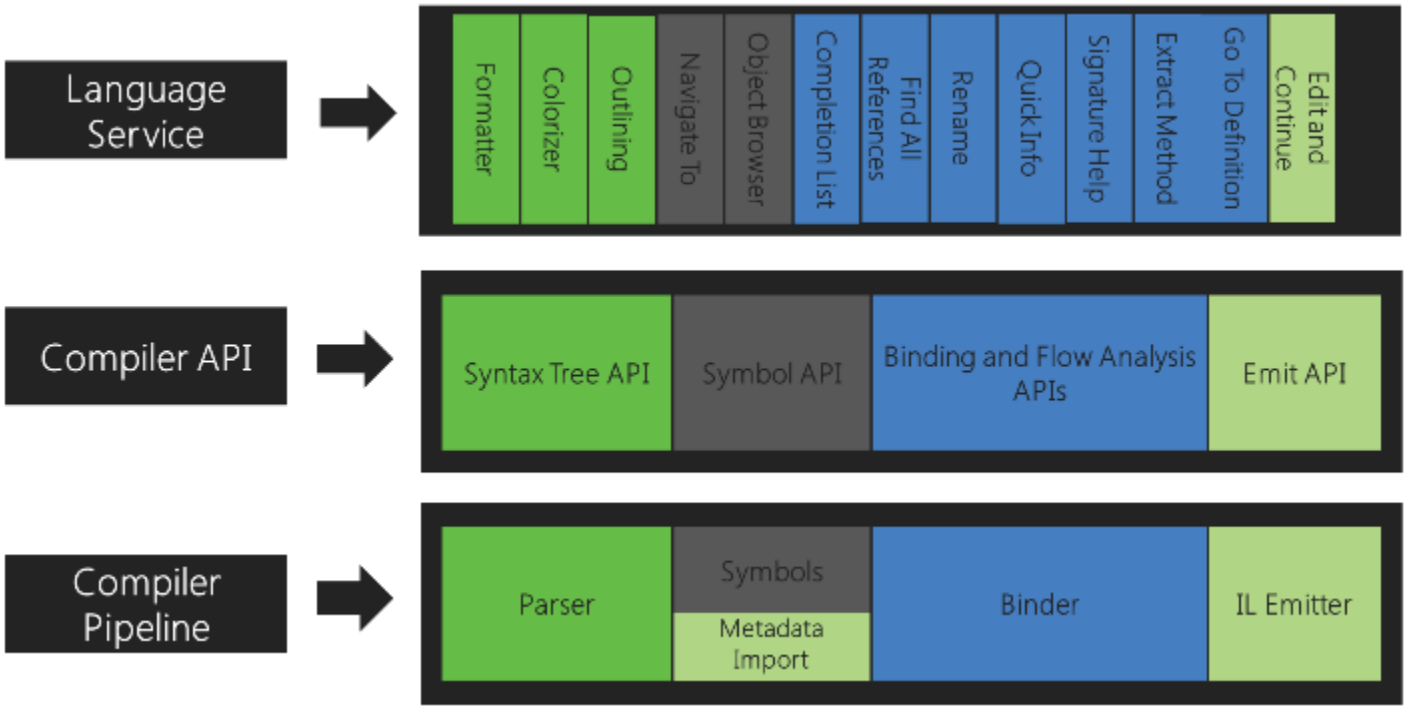
- C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe MyC#Test.cs

C# Compiler bisher

- Was passiert im Compiler?
  - Zugriff auf den Syntax Tree
  - Zugriff auf Meta-Daten
- Ausweg
  - Werkzeuge müssen selbst erzeugt werden -> ReSharper
  - Zugriff über Attribute -> Aspekt-orientierte Programmierung
  - Änderungen direkt am IL-Code
- Aber warum?
  - Der offizielle Compiler ist die einzige Referenz
  - Alles, was wir brauchen, macht dieser bereits

C# Compiler mit Roslyn

- Quelle: http://roslyn.codeplex.com/wikipage?title=Overview&referringTitle=Home



erminas

.NET Oldenburg user group

Was bringt mir das?

- Große Unterstützung für Tool Hersteller
- Semantic Diff (z.B. in Git) – z.B. Methoden vertauschen?
- C# Code und Scipting als Plugins in Applikation erlauben
- C# Code besser analysieren
- Schnelleres Kompilieren durch Task Parallel Library (TPL)
- „No-compile“ für ASP.NET vNext
  - Keine Neu-Kompilierung des Projekts bei Änderungen
  - Alles im Speicher, keine DLL auf Festplatte
  - http://www.hanselman.com/blog/IntroducingASPNETVNext.aspx
- Feature in C# gefällt nicht? -> Einfach ändern ☺

Beispiele

- http://roslyn.codeplex.com/wikipage?title=Samples%20and%20Walkthroughs&referringTitle=Home

Und C# 6.0 – Was sind neue Features?

- Aktuell siehe: https://roslyn.codeplex.com/wikipage?title=Language%20Feature%20Status
- Primary constructors
- Auto-property initializers / Getter-only auto-properties
- Declaration expressions
- Exception filters
- Using static members
- Expression-bodied members
- Event initializers
- Dictionary initializer
- Safe Navigation Operator
- Binary literals / Digit separators
- Params IEnumerable<T>
- Await in catch/finally
- ~~Indexed member initializer / Indexed member access~~

C# 6.0 – Wie nutzen?

- VS 2013 mit Roslyn Erweiterung
- VS 14 CTP
- Wichtig!

```
<Project ToolsVersion="14.0" ...>
  <PropertyGroup>
    ...
      <LangVersion>experimental</LangVersion>
  </PropertyGroup>
  ...
</Project>
```

erminas .NET Oldenburg user group

C# 6.0 – Primary Constructor

- Schonmal gesehen?

```
public class Car
{
    public Car(string manufacturer, int numberOfDoors,
        float fuelAmount)
    {
        FuelAmount = fuelAmount;
        NumberOfDoors = numberOfDoors;
        Manufacturer = manufacturer;
    }

    public float FuelAmount { get; set; }
    public int NumberOfDoors { get; private set; }
    public string Manufacturer { get; private set; }
}
```

C# 6.0 – Primary Constructor

- Besser (mit Auto-property initializers / Getter-only auto-properties)?

```
public class Car(string manufacturer, int
    numberOfDoors, float fuelAmount)
{
    public float FuelAmount { get; set; } = fuelAmount;
    public int NumberOfDoors { get;} = numberOfDoors;
    public string Manufacturer { get;} = manufacturer;
}
```

- Primary Constructor immer zuletzt aufgerufen
  (wenn überhaupt Constructor aufgerufen wird - Struct)

- Beispiel

C# 6.0 – Declaration expressions

```
int i;
if (int.TryParse("15",out i)) {
   ...
} else {
   ...
}
```

C# 6.0 – Declaration expressions

- Warum nicht so?

```
if (int.TryParse("15",out var i)) {
... // hier existiert i
}
else
{
... // hier existiert i ebenfalls
}

// hier gibt es i nicht mehr
```

- Funktioniert auch mit anderen Methodenaufrufen -> Beispiel

C# 6.0 – Exception filters

```
try
{
...
}
catch (ArgumentException e) if (MyMethod())
{
...
}
catch if (MyMethod2())
{
...
}
```

- Achtung bei der Reihenfolge der Ausführung (Beispiel)!

erminas .NET Oldenburg user group

C# 6.0 – Using static members

- Import von statische Methoden einer Klasse in aktuellen Namespace

```
using System.Console;
using System.IO.File;
using System.Math;
```

- Beispiel

C# 6.0 – Expression-bodied members

- Kurze Syntax von Methoden wie Lambda Ausdrücke

```
public double Dist => Sqrt(X * X + Y * Y);
```

- Derzeit geplant -> kein Beispiel ☹

C# 6.0 – Event initializers

- Wie in Object Initializer auch Events zuweisen

```
new Customer { Notify += MyHandler };
```

- Derzeit geplant -> kein Beispiel ☹

C# 6.0 – Dictionary initializer

- Erstellen von bspw. Dictionaries mit Inhalte per []-Operator

```
var dic = new Dictionary<int, string>
{
[0] = "hello",
[1] = "hello again"
}
```

- Bisher nur Collection Initializier:

```
var dic = new Dictionary<int, string>
{
  {0,"hello"},
  {1,"hello again"}
}
```

erminas .NET Oldenburg user group

C# 6.0 – Dictionary initializer

- Unterschied Dictionary und Collection Initializer:
  - []-Methode statt Add(…)
  - Klarere Syntax (Name-Value)
  - Für Objekte, die kein Add haben (XML Namespace)

- Kein Beispiel -> noch nicht in VS 14?

C# 6.0 – Safe Navigation Operator

null?

```
var g1 = parent.child.child.child;
```

C# 6.0 – Safe Navigation Operator

```
var g1 = parent.child.child.child;
```

Variante 1

```
var item = parent;
item = (item == null) ? null
  : item.child;
item = (item == null) ? null
  : item.child;
var g1 = (parent == null) ? null
  : item.child;
if (g1 != null) { ... }
```

Variante 2

```
var g1 = (Child)null;
var item = parent;
if (item != null)
{
    item = item.child;
    if (item != null)
    {
        item = item.child;
        if (item != null)
        {
            g1 = item.child;
        }
    }
}
if (g1 != null) {...}
```

erminas

.NET Oldenburg user group

C# 6.0 – Safe Navigation Operator

```
var g1 = parent.child.child.child;


// Safe Navigation Operator
var g1 = parent?.child?.child?.child;
if (g1 != null) {…}
```

C# 6.0 – Safe Navigation Operator

Wie verhält sich das „?“ ?

```
var a = new { b=null };
var x = a?.b.c;

var a = null;
var x = a?.b.c;
```

- Lange Diskussion: https://roslyn.codeplex.com/discussions/540883 (links-assoziativ oder short-circuit)
- Anderer Name: Null Propagation
- Beispiel

C# 6.0 – Binary literals / Digit separators (Planned)

- Binäre Werte direkt angeben:

```
0b1100101
```

- Unterteilung von Zahlenketten:

```
1_234_567
0b11_00_10_1
0xAA_56_FF
```

- Planned -> kein Beispiel

C# 6.0 – Params IEnumerable (Planned)

```
public void MyMethod(params int[] x)
{}

public void MyMethod2()
{
        var x = new int[]{ 1,2,3};
        MyMethod(x.Where(x => x > 1)); // Fehler
}
```

- Alternative:

```
MyMethod(x.Where(x => x > 1).ToArray());
```

erminas

.NET Oldenburg user group

C# 6.0 – Params IEnumerable (Planned)

```
public void MyMethod(params IEnumerable<int> x)
{}


public void MyMethod2()
{
        var x = new int[]{ 1,2,3};
        MyMethod(x.Where(x => x > 1));
}
```

- Mehr Infos: (http://stackoverflow.com/questions/2128759/params-ienumerablet-c-sharp)
- Noch nicht in VS 14 -> Kein Beispiel

erminas .NET Oldenburg user group

C# 6.0 – Await in catch/finally

```
static async Task f()
{
  ExceptionDispatchInfo capturedException = null;
  try
  {
    await TaskThatFails();
  }
  catch (Exception ex)
  {
    capturedException =
         ExceptionDispatchInfo.Capture(ex);
  }
  if (capturedException != null)
  {
    await ExceptionHandler();
    capturedException.Throw();
  }
}
```

- Jetzt nicht mehr notwendig ☺ (ohne Beispiel)

erminas .NET Oldenburg user group

C# 6.0 – Indexed member initializer / Indexed member access

```
var dic = new Dictionary<string, string>
{
  $first="hello",
  $second="hello again"
}

var x = dic.$first;
```

- Sollte vom Compiler ausgewertet werden können
- (Gott sei Dank) nicht eingebaut!

erminas .NET Oldenburg user group

(Meine) Offene Wünsche

- Non-nullable types wie:

```
string! myString="hello";
```

- String interpolation:

```
var a = "Now is $(DateTime.Now).";

// oder

string.format(“Hallo {0.Name}, es ist {1.
ToShortTimeString()}”, personA, new DateTime());
```

erminas

.NET Oldenburg user group

Mehr Infos

- Build 2014 http://channel9.msdn.com/Events/Build/2014/2-577
- http://roslyn.codeplex.com/
- https://roslyn.codeplex.com/wikipage?title=Language%20Feature%20Status
- http://channel9.msdn.com/Events/Ch9Live/Channel-9-Live-at-Tech-Ed-Europe-2012/Dustin-Campbell-Roslyn:

erminas

.NET Oldenburg user group

Thank you.