



BASTA!

Rainer Stropek | software architects

Der ultimative C#-4- Workshop

C# 4 steckt für viele Entwickler immer noch voller Geheimnisse und Überraschungen. Wussten Sie, dass sich fast alle foreach-Schleifen durch **LINQ** ersetzen lassen? Dass der **Zugriff auf Office** und generell COM-Bibliotheken mit C# 4 zum Kinderspiel wurde? Dass C# 4 voller Möglichkeiten steckt, Ihre Programme zu **parallelisieren**?

Wenn in Ihrer täglichen Arbeit die **Vorteile der aktuellen C#-Version** noch nicht in Fleisch und Blut übergegangen sind, sind Sie in diesem Workshop richtig. Ihr Trainer, Rainer Stropek, konzentriert sich auf **praktische Beispiele, Tipps und Tricks**, die Ihnen während des Workshops auch zum **Mitmachen** zur Verfügung stehen.

Agenda

- Was ist neu in **Visual Studio 2010** für C# Entwickler?
- **Office Interop** – COM, No PIA, Optional Parameters, etc.
- **Parallel Computing** mit Tasks, PLINQ, etc.
- **dynamic** Keyword und Dynamic Language Runtime (**DLR**)
- Managed Extensibility Framework (**MEF**)

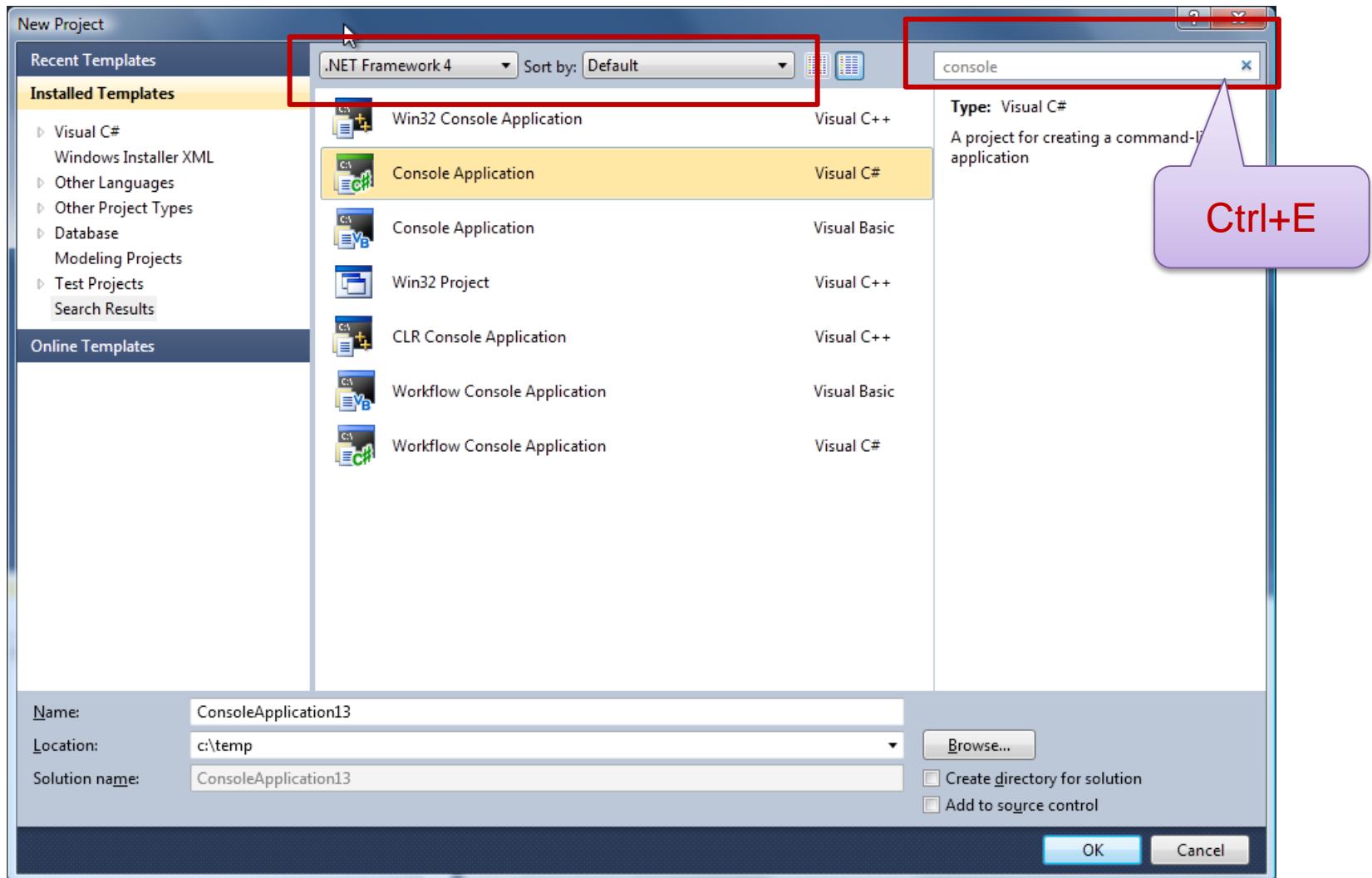
Was ist neu in Visual Studio 2010 für C# Entwickler?

VISUAL STUDIO 2010

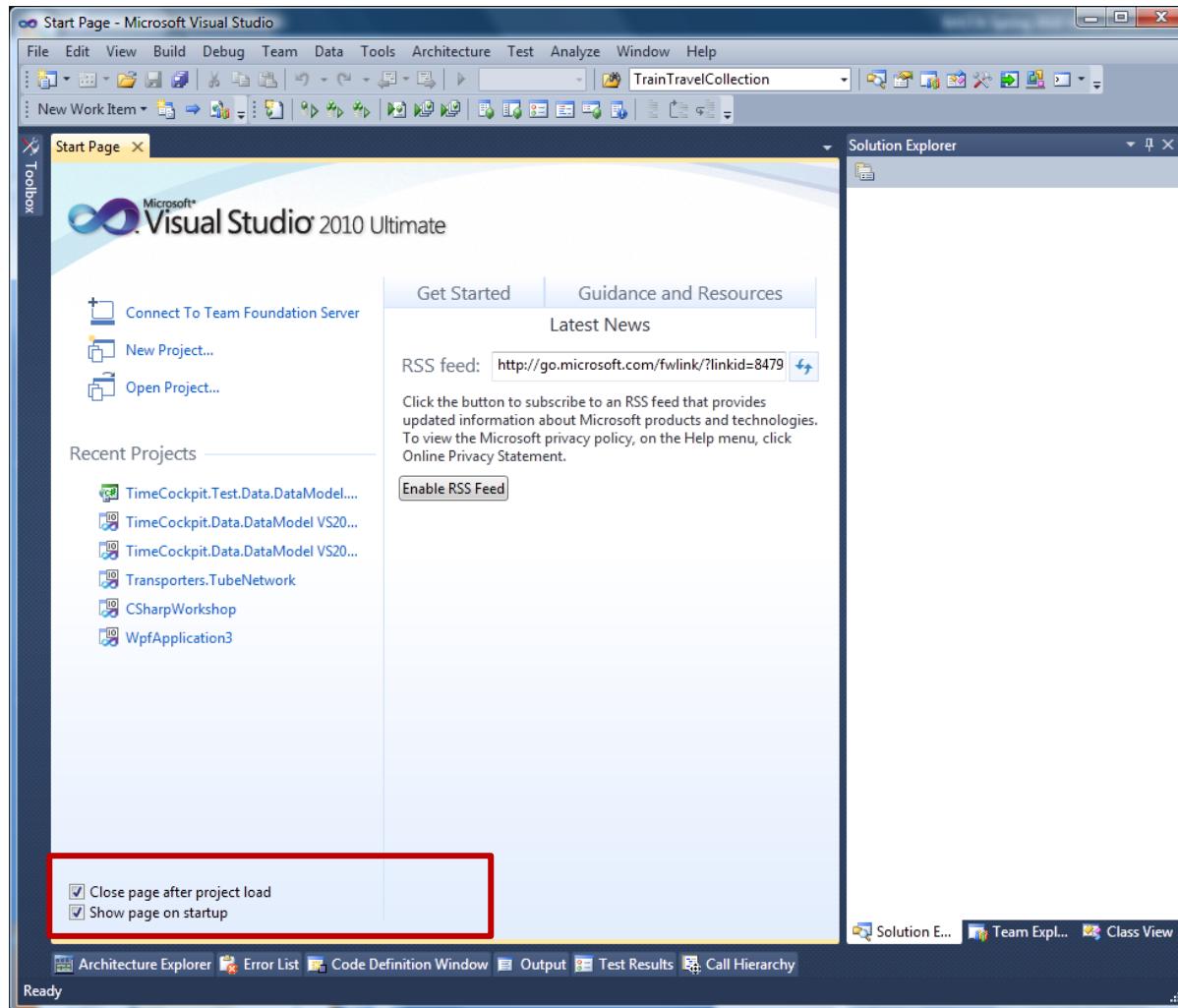
C# IDE

SOLUTIONS UND PROJEKTE

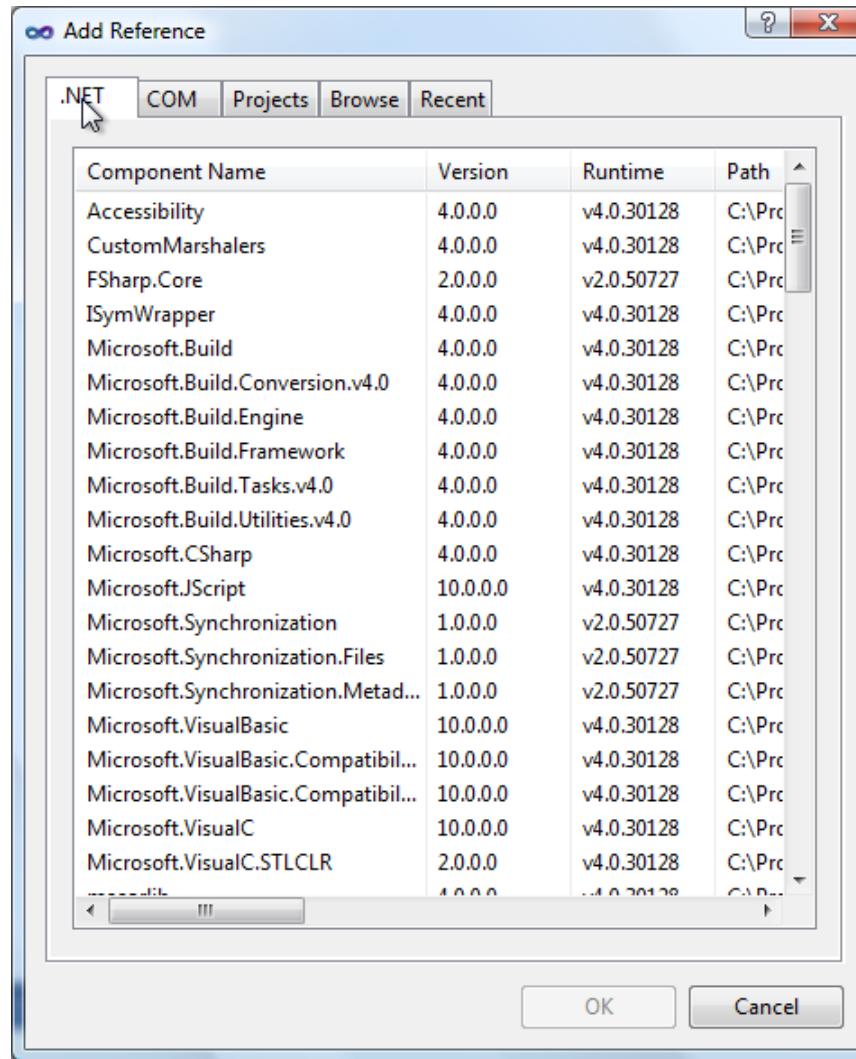
Verbesserter Project/New Dialog



Verbesserte Startpage



Async Add Reference ☺



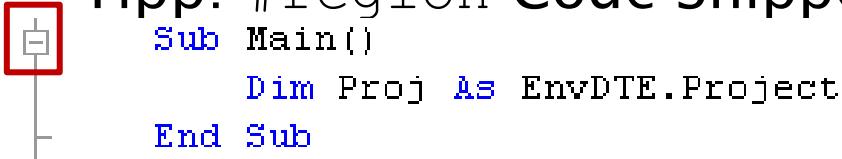
VISUAL STUDIO EDITOR

Code Selection, Copy/Move

- Tipp: Column Mode
 - Alt+Maus oder **Shift+Alt+Cursor**
- Cut, Copy, Paste
 - **Ctrl+X, Ctrl+C, Ctrl+V** 
 - Tipp: Clipboard ring (**Ctrl+Shift+V**)
 - Zugriff auf die letzten 20 kopierten Texte
 - Tipp: Ohne Markierung ganze Zeile ausschneiden/kopieren

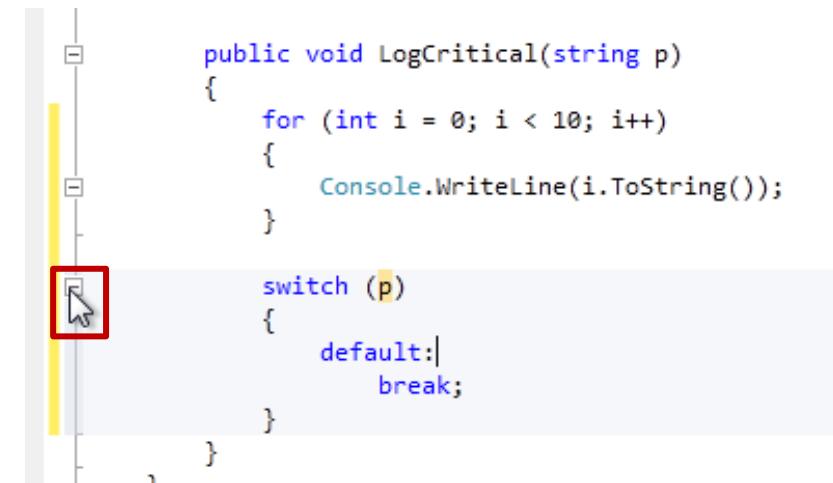
Outlining

- Toggle Outlining (**Ctrl+M, M**)
- Collapse to Definitions (**Ctrl+M, O**)
 - Tipp: #region Code Snippet



```
#region Code Snippet
Sub Main()
    Dim Proj As EnvDTE.Project
End Sub
```

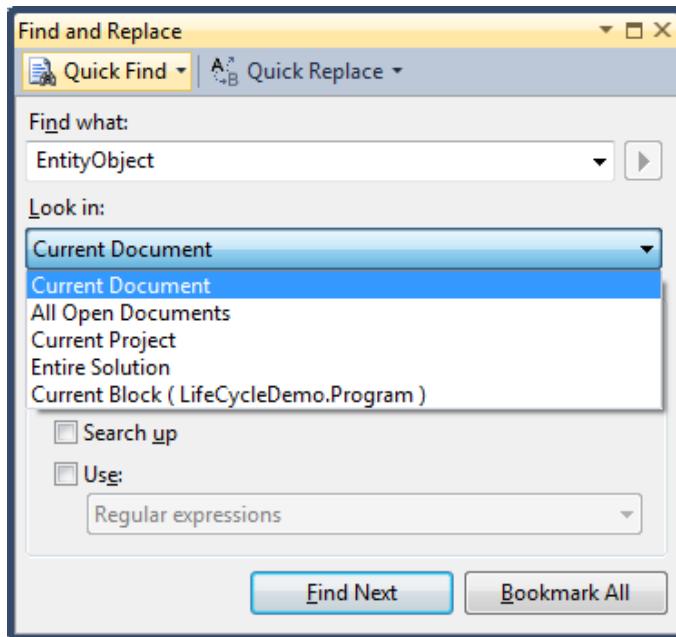
- **Neu:** Ad Hoc Blocks
 - Markieren des gewünschten Codeblocks
 - *Hide Selection* (**Ctrl+M, H**)
 - → Ad Hoc Block erzeugt



Sonstige Editor-Tipps

- Zooming
 - Zoom in Textfenster mit **Ctrl+Mousewheel**
 - Nicht in Fenstern mit Icons

Suchen und Ersetzen (1/3)

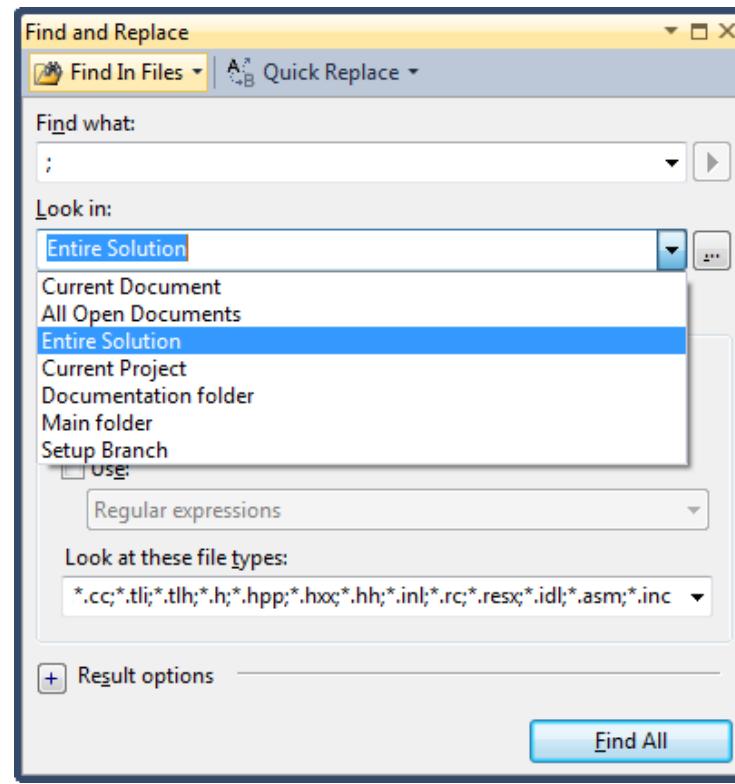


Quick Find

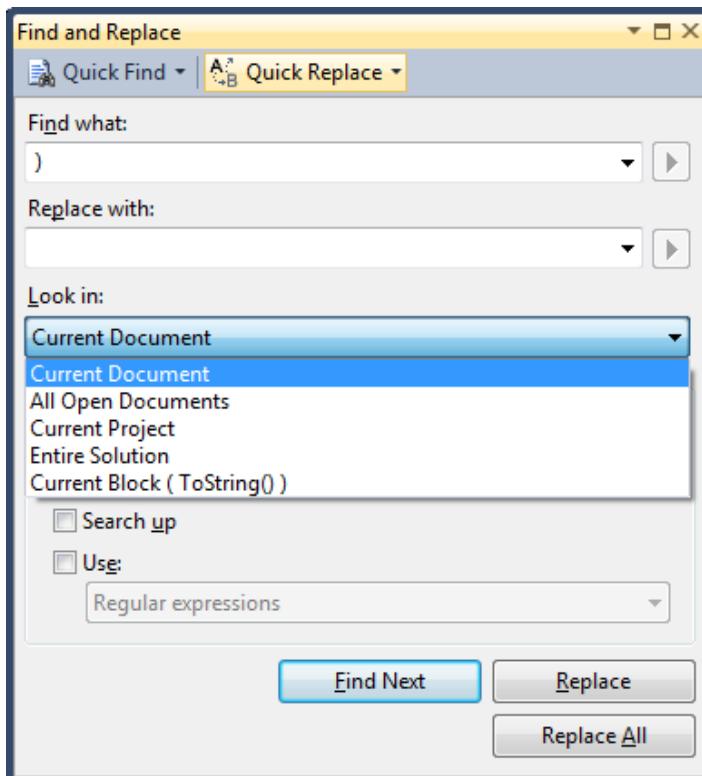
Ctrl+F



Find in Files
Ctrl+Shift+F

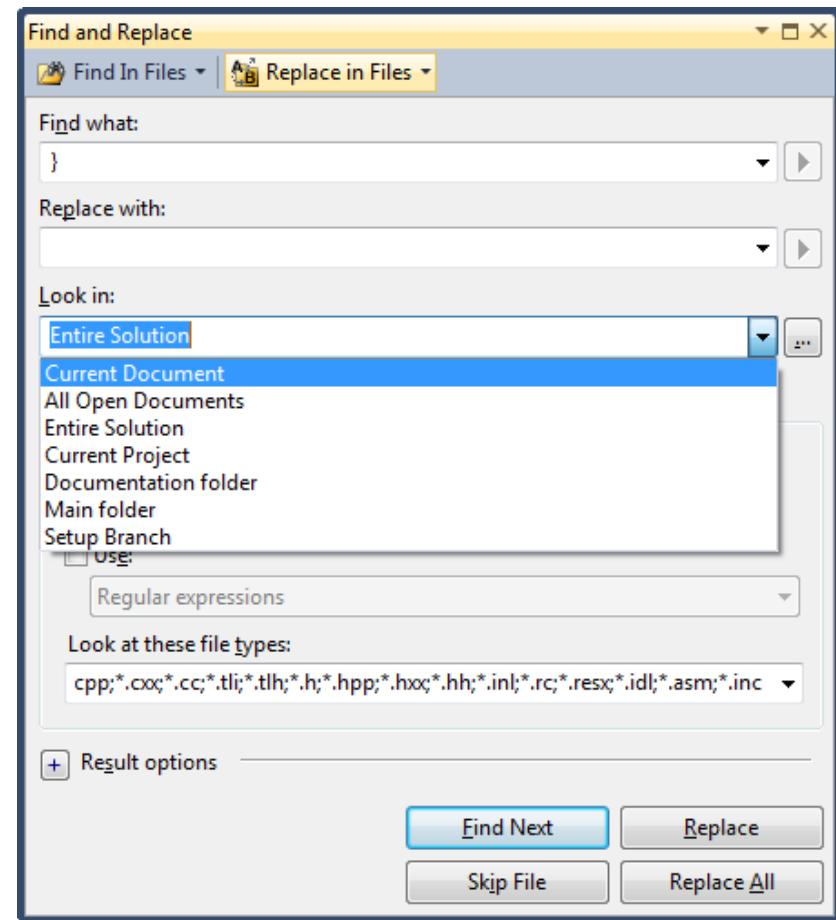


Suchen und Ersetzen (2/3)



Quick Replace

Ctrl+H

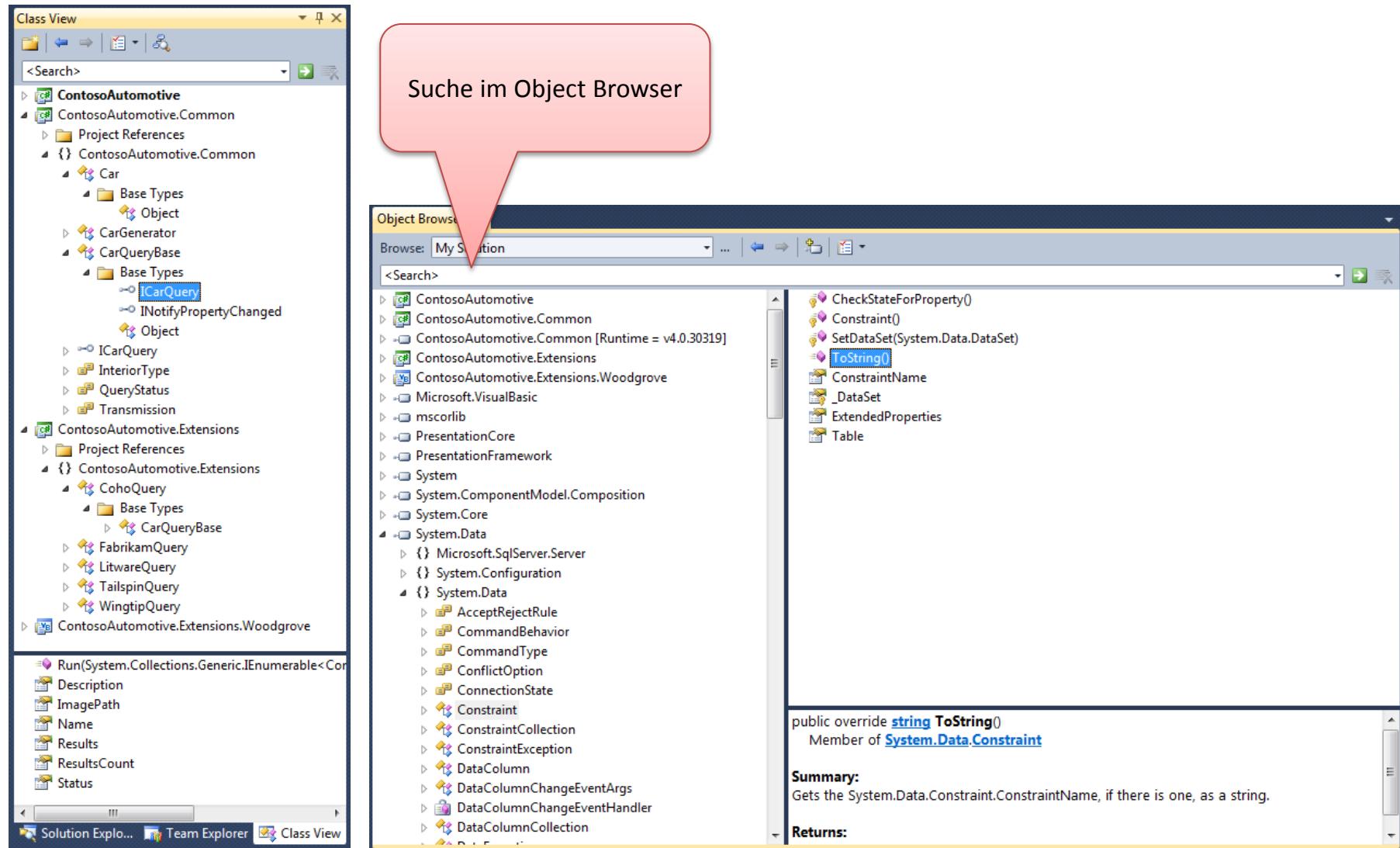


Replace in Files
Ctrl+Shift+H

Navigate To (1/3)

- Verbesserte Suchmöglichkeit
 - IMHO besser als *Object Browser* (**Ctrl+W, J**)
 - Sucht auch nach Dateinamen ☺ (z.B. *DBQ* findet *DbClientQuery.cs*)
 - CamelCaseSuche (z.B. *MAN* findet *MarkAsNew*)
- Edit, Navigate To (**Ctrl+,**)
- Tipps
 - Alles kleingeschrieben → case insensitive
 - Groß- und Kleinbuchstaben → case sensitive
 - Leertaste = And-Verknüpfung

Class View und Object Browser



Suche im Object Browser

The screenshot shows the Visual Studio interface with two main windows: Class View and Object Browser.

Class View: Shows the project structure of "ContosoAutomotive". It includes a tree view of namespaces like ContosoAutomotive, ContosoAutomotive.Common, and ContosoAutomotive.Extensions, along with their respective classes and interfaces. At the bottom, there is a list of properties for a selected item.

Object Browser: Shows a list of objects and their members. The "Browse" dropdown is set to "My Solution". The list includes various types from the solution and the .NET framework, such as ContosoAutomotive, ContosoAutomotive.Common, ContosoAutomotive.Extensions, and System.Data.Constraint. A specific member, `ToString()`, is selected, and its details are shown in the bottom pane.

Bottom Pane (Details for `ToString()`):

```
public override string ToString()
Member of System.Data.Constraint

Summary:
Gets the System.Data.Constraint.ConstraintName, if there is one, as a string.

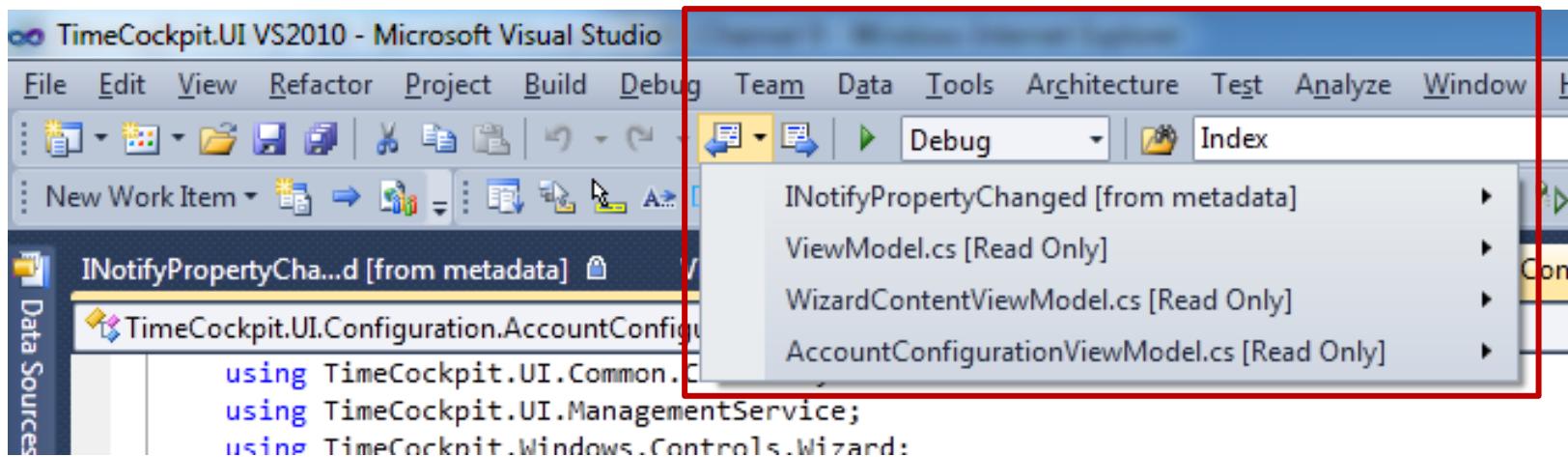
Returns:
```

Navigate To (2/3)

- Wann ist *Find Symbol* (**Alt+F12**) besser?
 - Search Scope kann festgelegt werden
 - Findet auch Verwendung, nicht nur Definition
 - Kann Komponenten ohne Sourcecode durchsuchen (z.B. Suche nach *File.Open*)
- Wann ist *Find* besser?
 - *Quick Find* (**Ctrl+F**) vs. *Find In Files* (**Ctrl+Shift+F**)
 - Tipp: Quick Replace (**Ctrl+H**) vs. Replace In Files (**Ctrl+Shift+H**)
 - Regular Expressions

Navigate To (3/3)

- Tipp: **F8**, um in Listen zum nächsten Element zu kommen (*go to next location*)
 - Build Errors
 - Find Results
 - Etc.
- Tipp: **Ctrl+Minus**, um zu zuletzt angesehenen Sourcecodezeile zurück zu springen (*navigate backward*)



Call Hierarchy (1/2)

- Zeigt...
 - ...Aufrufe von/in ausgewähltem Member
 - ...Implementierungen eines Interface
 - ...Implementierungen eines virtuellen oder abstrakten Members
- „*Find all references (Ctrl+K, R) on steroids*“
 - Kontextmenü auf Member, View Call History
 - **Ctrl+K, T**

Call Hierarchy (1/2)

- Verbesserungen gegenüber *Find all references*
 - Mehrstufig (nicht mehr ein *Find all references* nach dem anderen)
 - Scope kann eingeschränkt werden
 - Deferred execution
 - Richtigere Ergebnisse (vgl. *OnPropertyChanged*-Beispiel)
- Einschränkungen
 - Verwendung außerhalb von C# Code (z.B. XAML)

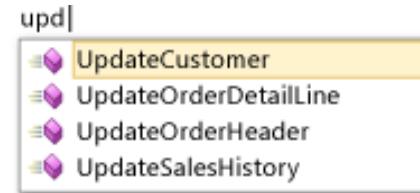
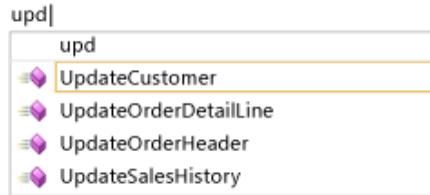
Code Definition Window

- *View, Code Definition Window* (**Ctrl+W, D**)
- Zeigt die Definition eines Symbols auf Grundlage von
 - Sourcecode oder
 - binären referenzierten Assemblies
- Reagiert auf
 - Cursorposition
 - Aktuelle Auswahl in *Class View, Object Browser* oder *Call Browser*

CODE GENERIEREN

IntelliSense Mode

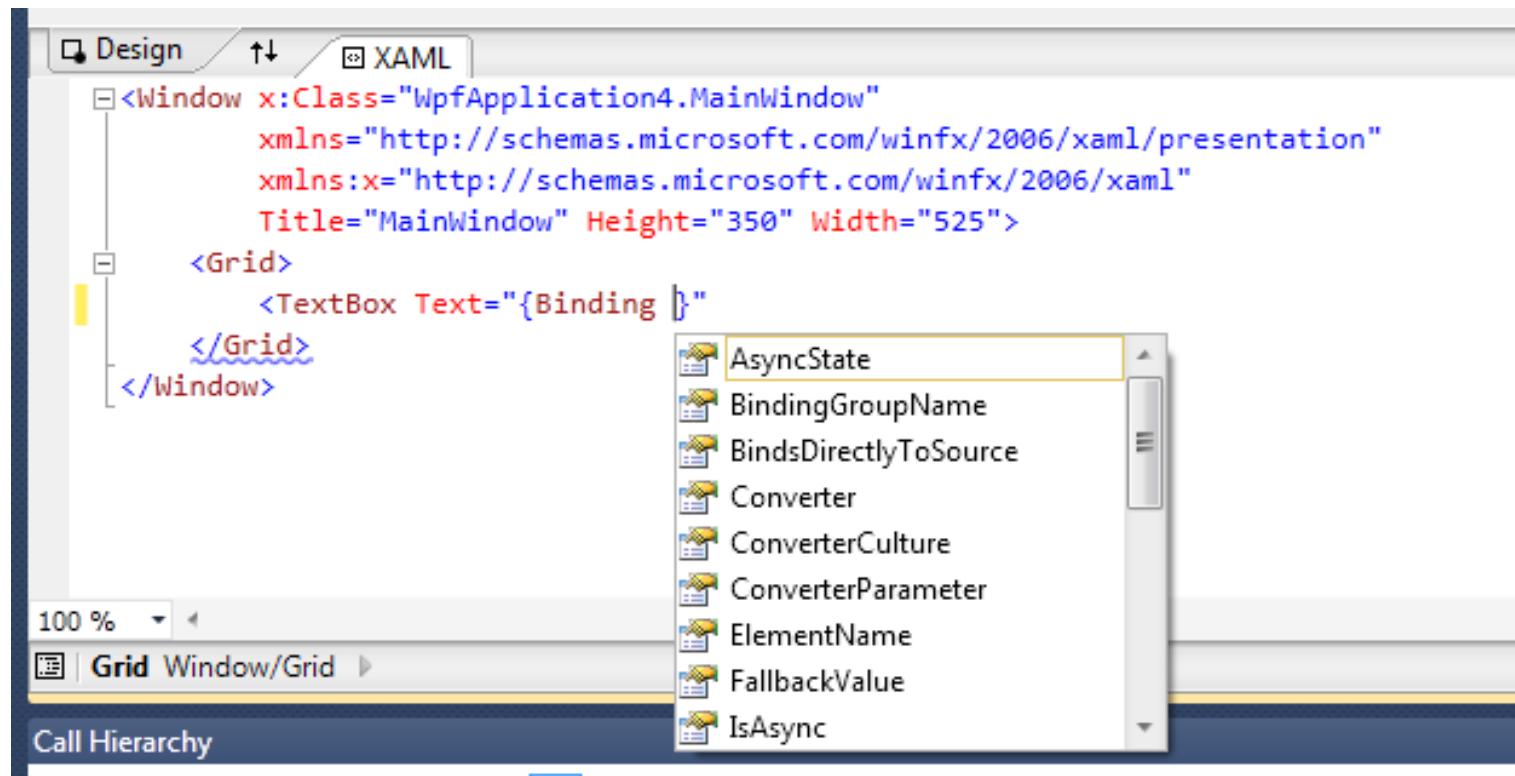
- Modi
 - Completion Mode (wie bisher)
 - Suggestion Mode (für TDD; siehe *Generate From Usage*)



- Umschalten mit **Ctrl+Alt+Space**
- BTW – Wie startet man die Member List manuell?
Ctrl+J
- BTW – Parameterinformationen blendet man mit
Ctrl+Shift+Space ein

IntelliSense in XAML...

- ...ist endlich da ☺ ☺ ☺



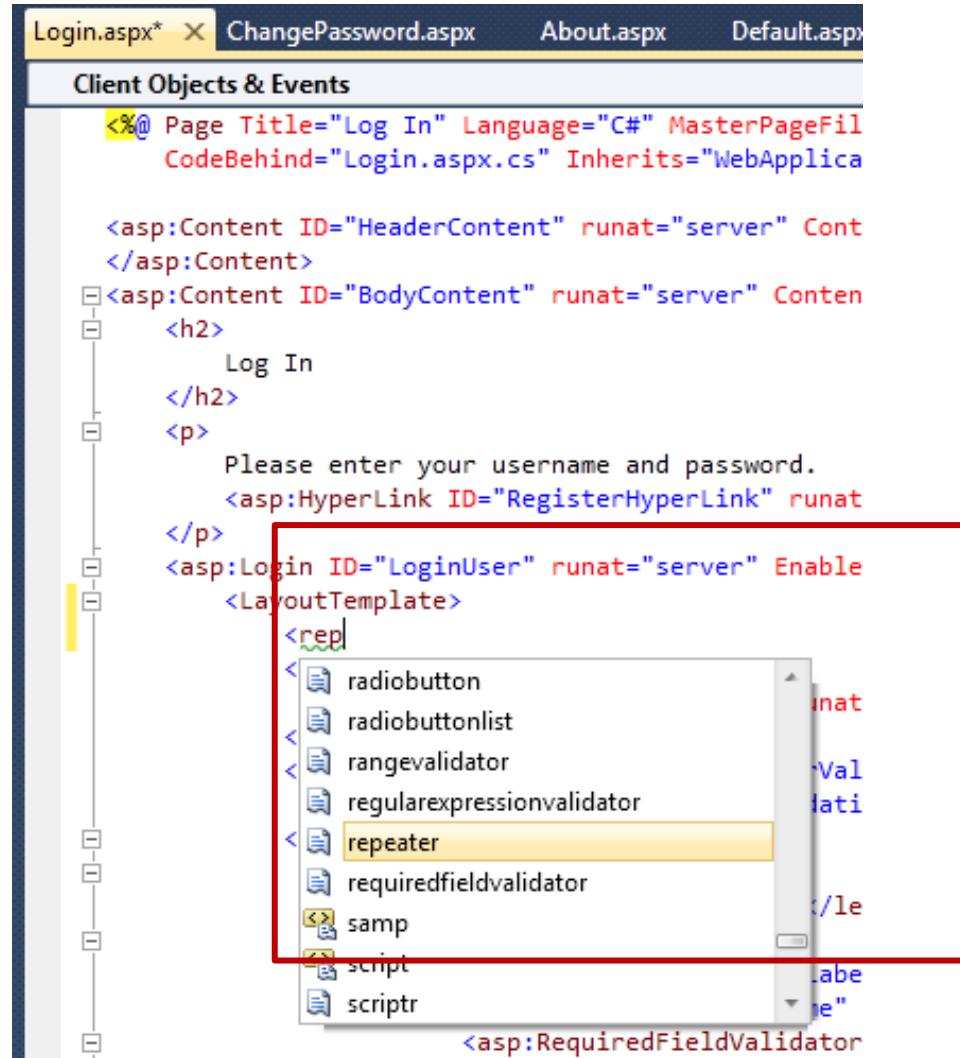
Generate From Usage (1/2)

- Hilfreich bei TDD
- Erreichbar über...
 - ...Maus (Smart Tag = Pain)
 - ...**Ctrl+.** (=Pain Killer)
- Generiert Typ, Field, Property oder Methode
 - Tipp: *Generate New Type* wenn Code in einem anderen Projekt generiert werden soll (typisch bei Testprojekten)

Generate From Usage (2/2)

- `using` hinzufügen
 - Referenz muss im Projekt enthalten sein
 - Problem: Extension Methods
- Abstrakte Basisklassen implementieren
- Interfaces implementieren

Neu: Code Snippets in ASP.NET



The screenshot shows the Visual Studio IDE with the 'Login.aspx' file open. The code editor displays the following ASPX page source:

```
<%@ Page Title="Log In" Language="C#" MasterPageFile="~/MasterPages/MasterPage.master" CodeBehind="Login.aspx.cs" Inherits="WebApplication1.Login" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeaderContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="BodyContent">
<h2>Log In</h2>
<p>Please enter your username and password.</p>
<asp:HyperLink ID="RegisterHyperLink" runat="server" Text="Create New Account?>"></asp:HyperLink>
<asp:Login ID="LoginUser" runat="server" EnablePasswordRetrieval="true" EnablePasswordReset="true">
<LayoutTemplate>
<repeater>
```

A code snippet completion dropdown is displayed, showing suggestions for controls under the '<repeater>' tag. The 'repeater' item is highlighted with a yellow background, indicating it is the selected suggestion. Other items in the list include 'radiobutton', 'radiobuttonlist', 'rangevalidator', 'regularexpressionvalidator', 'requiredfieldvalidator', 'sample', 'script', and 'scriptptr'. A red box highlights the completion dropdown.

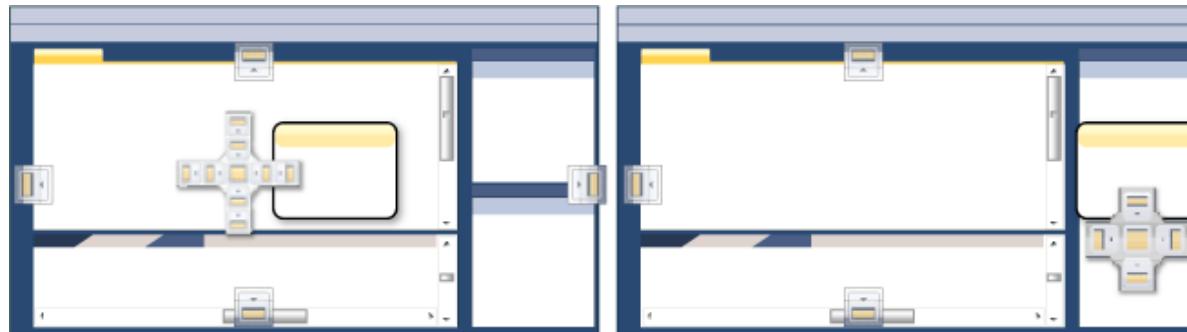
FENSTER- UND ANSICHTSVERWALTUNG

Docking (1/2)

- Document Windows
 - Im Document Frame
 - Neu: Auch außerhalb der IDE-Grenzen
(auch auf eigenem Monitor)
- Tipp: **Ctrl+Doubleclick** auf Fenstertitel, um das Fenster zur letzten Position zurückzubringen

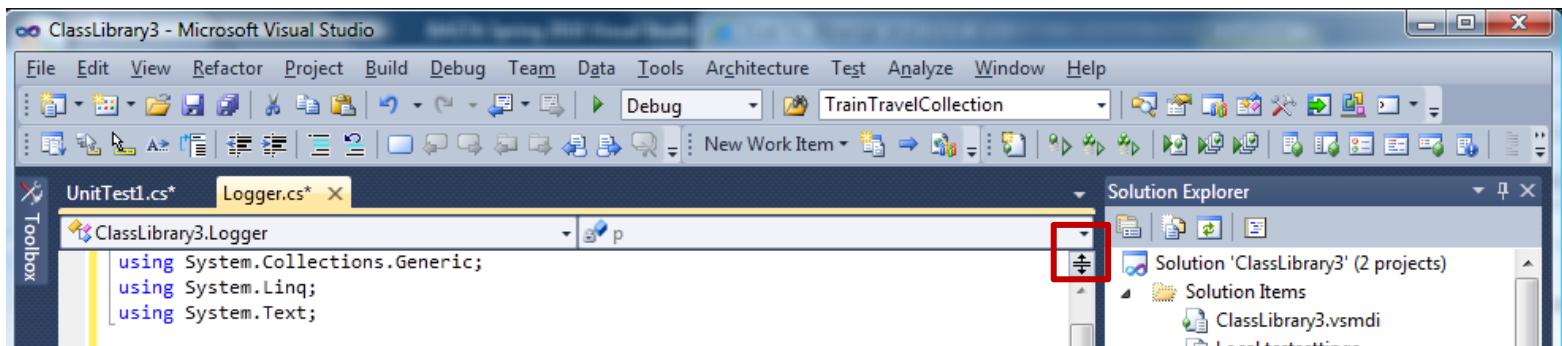
Docking (2/2)

- Tools
 - Wie bisher angedockt am IDE-Rand
 - Neu: Auch im Document Frame
 - Neu: Auch außerhalb der IDE-Grenzen
(auch auf eigenem Monitor)

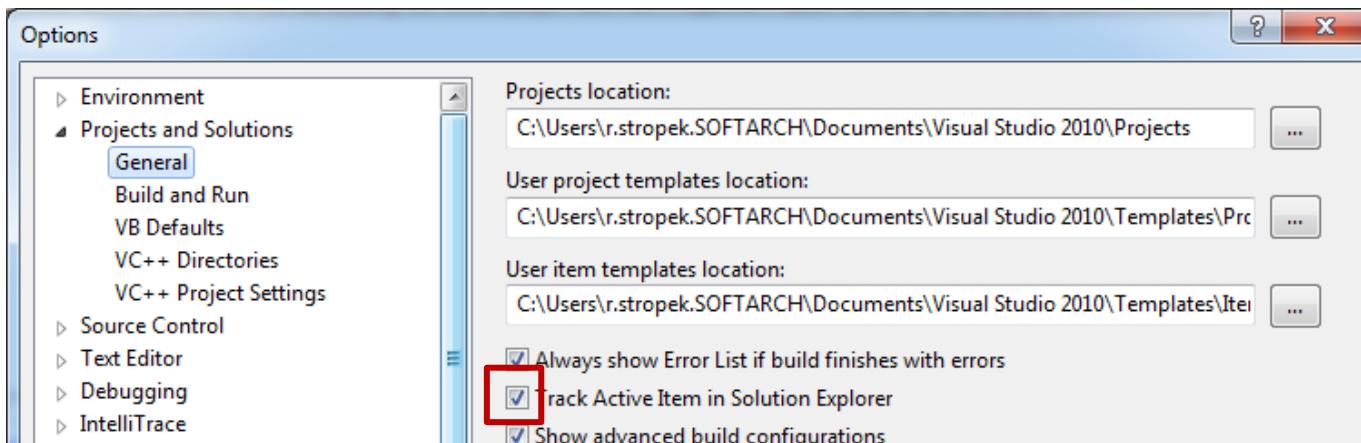


BTW – Kennen Sie den?

- *Go to open file (Ctrl+Alt+Down)*
- *Split Window*



- *Track Active Item in Solution Explorer*



Selection

- Wie in früheren Version Boxed Selection mit **Alt+Click&Drag**
- Neu in VS2010
 - Multi-Line Insert
 - Paste
 - Zero-Length Boxes (multi-line insertion point)

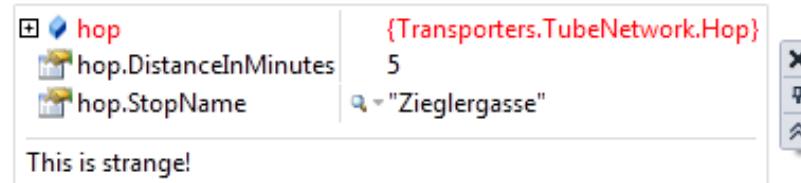
DEBUGGING

Data Tips (1/2)

- Wie bisher im Debugger für Variablen im aktuellen Scope
 - Tipp: Data Tip transparent machen mit **Ctrl**
- Neu:
 - *Pin to source*: Data Tip ist mit Position im Sourcecode verknüpft und scrollt mit
 - Kommentare bei *pinned data tips*

```
};

var currentTime = startTime;
foreach (var hop in route)
{
    travel.Stops.Add(new StationStop()
    {
        Station = stations.FirstOrDefault(s => s.Name == hop.StopName),
        StopTime = currentTime + TimeSpan.FromMinutes(hop.DistanceInMinutes)
    });
}
```

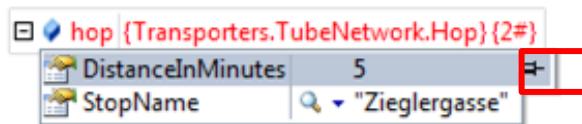


[]	hop	{Transporters.TubeNetwork.Hop}
[]	hop.DistanceInMinutes	5
[]	hop.StopName	"Zieglergasse"

This is strange!

Data Tips (2/2)

- Pinning auch für Subexpressions möglich



- Tipp: Object-IDs

```

var currentTime = startTime;
foreach (var hop in route)
{
    travel.Stops.Add(new StationStop()
    {
        Station = stations.FirstOrDefault(s => s.Name == hop.StopName),
        StopTime = currentTime = currentTime + TimeSpan.FromMinutes(hop.DistanceInMinutes)
    });
}

this.TrainTravels.Add(travel);

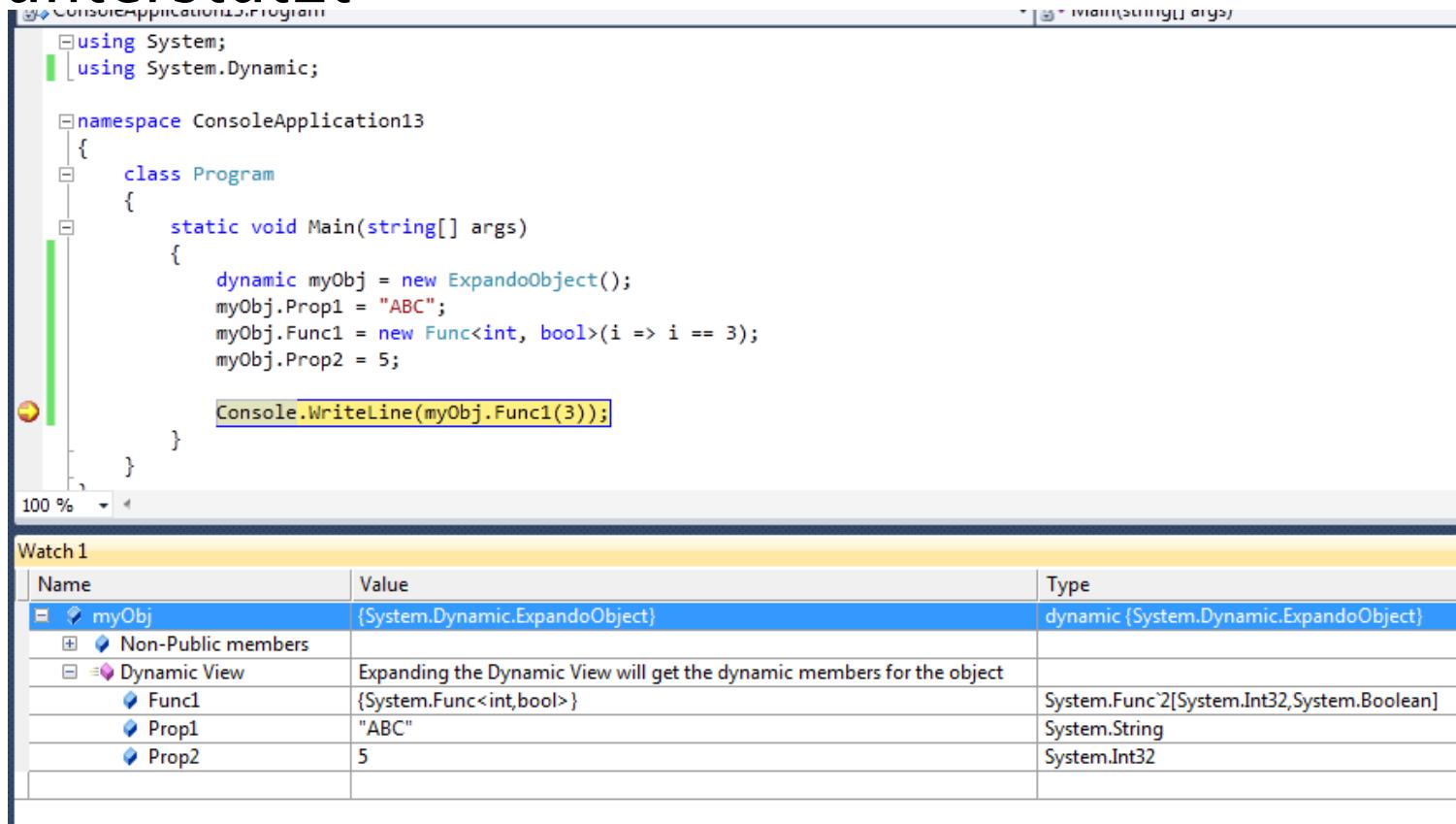
```

Watch 1

Name	Value	Type
route	{Transporters.TubeNetwork.Hop[4]}	System.Collections.Generic.IEnumerable<Transporters.TubeNetwork.Hop>
[Transporters.TubeNetwork.Hop[0]]	{Transporters.TubeNetwork.Hop}	Transporters.TubeNetwork.Hop
[Transporters.TubeNetwork.Hop[1]]	{Transporters.TubeNetwork.Hop{1#}}	Transporters.TubeNetwork.Hop
[Transporters.TubeNetwork.Hop[2]]	{Transporters.TubeNetwork.Hop}	Transporters.TubeNetwork.Hop
[Transporters.TubeNetwork.Hop[3]]	{Transporters.TubeNetwork.Hop}	Transporters.TubeNetwork.Hop

Unterstützung für DLR

- dynamic Datentyp wird im Debugger speziell unterstützt



The screenshot shows a Visual Studio IDE window. The code editor displays a C# program named 'ConsoleApplication13'. The Main method contains the following code:

```
using System;
using System.Dynamic;

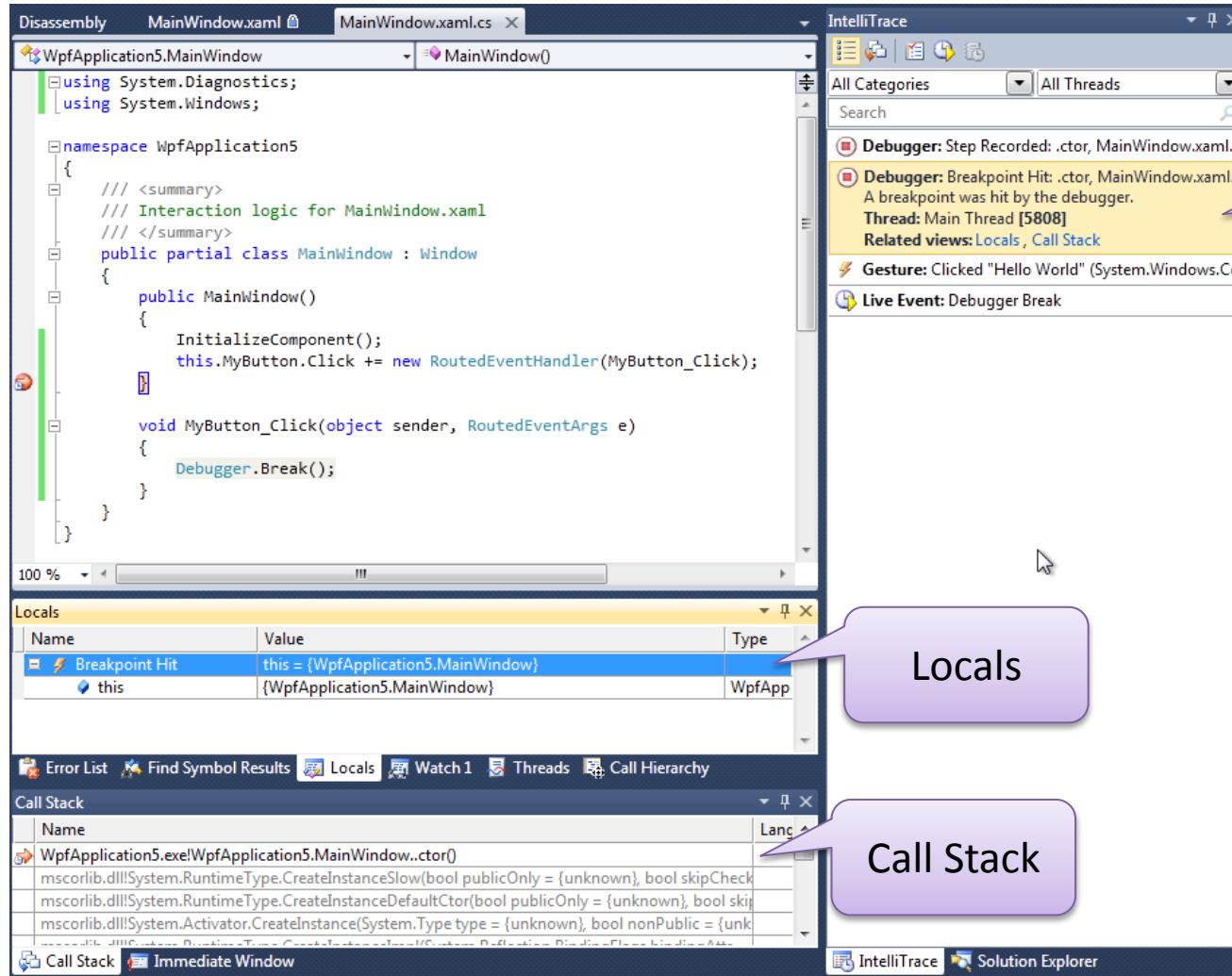
namespace ConsoleApplication13
{
    class Program
    {
        static void Main(string[] args)
        {
            dynamic myObj = new ExpandoObject();
            myObj.Prop1 = "ABC";
            myObj.Func1 = new Func<int, bool>(i => i == 3);
            myObj.Prop2 = 5;

            Console.WriteLine(myObj.Func1(3));
        }
    }
}
```

The line `Console.WriteLine(myObj.Func1(3));` is highlighted with a blue selection bar. Below the code editor is the 'Watch 1' window, which lists the variables in the current scope:

Name	Value	Type
myObj	{System.Dynamic.ExpandoObject}	dynamic {System.Dynamic.ExpandoObject}
+ Non-Public members		
+ Dynamic View	Expanding the Dynamic View will get the dynamic members for the object	
Func1	{System.Func<int,bool>}	System.Func<2[System.Int32,System.Boolean]
Prop1	"ABC"	System.String
Prop2	5	System.Int32

IntelliTrace (1/2)



The screenshot shows the Visual Studio interface during a debugging session. The top navigation bar includes Disassembly, MainWindow.xaml, and MainWindow.xaml.cs tabs. The code editor displays C# code for a MainWindow class.

IntelliTrace Window:

- Debugger: Step Recorded: .ctor, MainWindow.xaml.cs
- Debugger: Breakpoint Hit: .ctor, MainWindow.xaml.cs
 - A breakpoint was hit by the debugger.
 - Thread: Main Thread [5808]
 - Related views: Locals, Call Stack
- Gesture: Clicked "Hello World" (System.Windows.Controls.Button)
- Live Event: Debugger Break

Locals Window:

Name	Value	Type
Breakpoint Hit	this = {WpfApplication5.MainWindow}	WpfApp
this	{WpfApplication5.MainWindow}	WpfApp

Call Stack Window:

Name
WpfApplication5.exe!WpfApplication5.MainWindow..ctor()
mscorlib.dll!System.RuntimeType.CreateInstanceSlow(bool publicOnly = {unknown}, bool skipCheck = {unknown})
mscorlib.dll!System.RuntimeType.CreateInstanceDefaultCtor(bool publicOnly = {unknown}, bool skipCheck = {unknown})
mscorlib.dll!System.Activator.CreateInstance(System.Type type = {unknown}, bool nonPublic = {unknown})

Annotations:

- A purple callout points to the IntelliTrace window with the text "Events".
- A purple callout points to the Locals window with the text "Locals".
- A purple callout points to the Call Stack window with the text "Call Stack".

IntelliTrace (2/2)

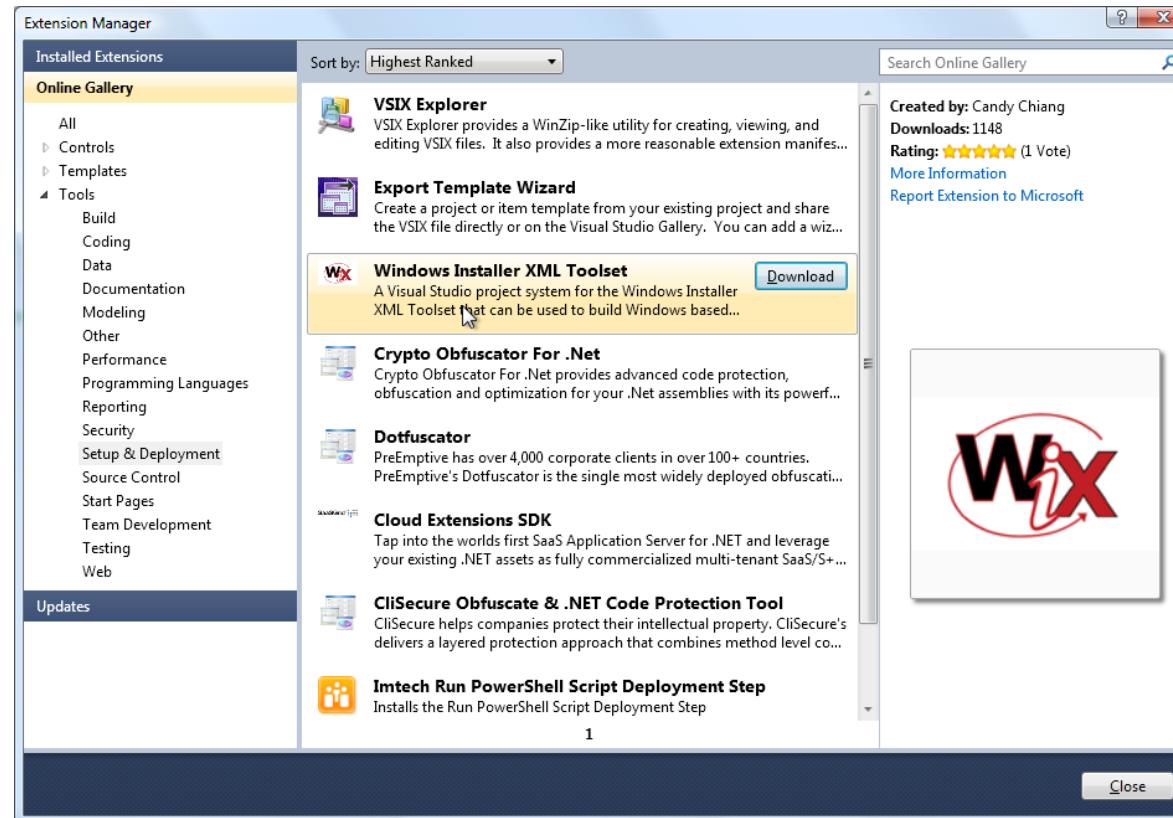
- Aufzeichnen von *Events*
 - Definierbar in *Tools / Options / IntelliTrace*
- Optional auch *Call Informations*
 - Verbraucht mehr Ressourcen
 - Ein/Ausschalten in *Tools / Options / IntelliTrace*

TOOLS

Extension Manager (1/2)

Tools, Extension Manager

(<http://visualstudiogallery.msdn.microsoft.com/en-us/>)

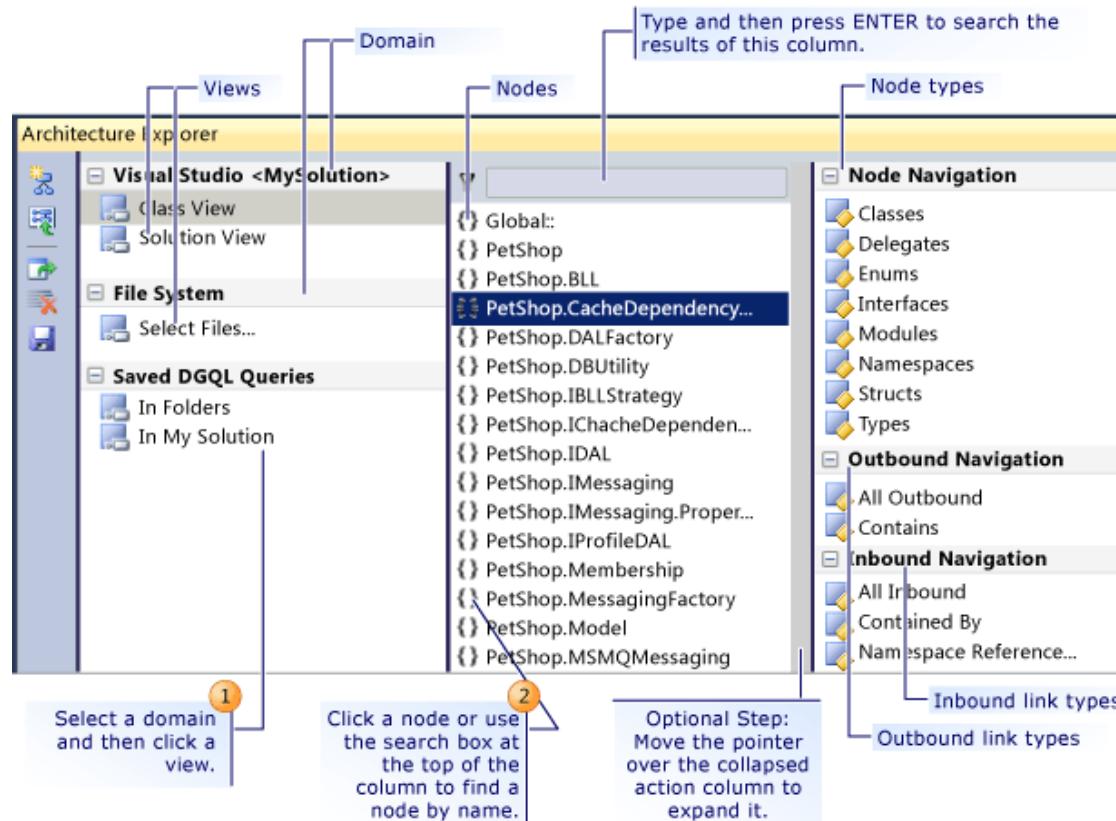


The screenshot shows the Visual Studio Extension Manager window. The left sidebar has 'Online Gallery' selected under 'Tools'. The main area lists several extensions:

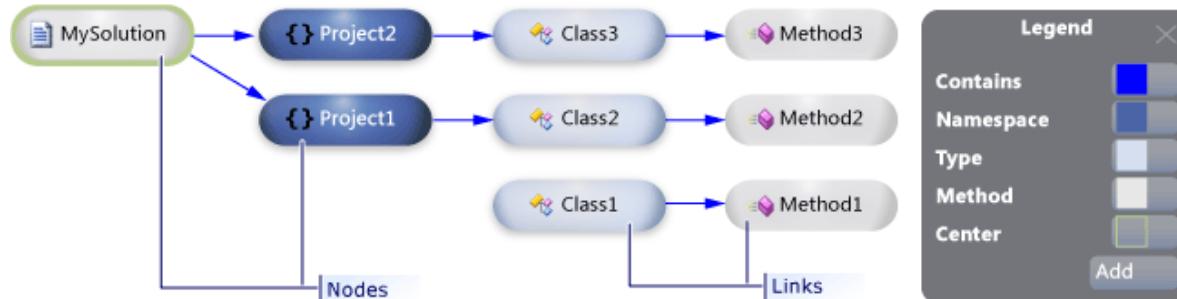
- VSIX Explorer**: A WinZip-like utility for creating, viewing, and editing VSIX files.
- Export Template Wizard**: Creates a project or item template from your existing project and shares it directly or on the Visual Studio Gallery.
- Windows Installer XML Toolset**: A Visual Studio project system for the Windows Installer XML Toolset.
- Crypto Obfuscator For .Net**: Provides advanced code protection, obfuscation, and optimization for .Net assemblies.
- Dotfuscator**: PreEmptive has over 4,000 corporate clients in over 100+ countries. Dotfuscator is the single most widely deployed obfuscator.
- Cloud Extensions SDK**: Tap into the world's first SaaS Application Server for .NET and leverage your existing .NET assets as fully commercialized multi-tenant SaaS/S+...
- ClSecure Obfuscate & .NET Code Protection Tool**: ClSecure helps companies protect their intellectual property. ClSecure's delivers a layered protection approach that combines method level co...
- Imtech Run PowerShell Script Deployment Step**: Installs the Run PowerShell Script Deployment Step.

A large preview window on the right shows the Wix logo.

Architecture Explorer



Dependency Graphs



Read more about help, find the right tools

RESOURCES

Tool Reference

- [Sandcastle](#)
 - Documentation Compiler for Managed Class Libraries
- [GhostDoc](#)
 - Generates documentation based on naming conventions
- [StyleCop](#)
 - Analyzes C# source code to enforce a set of style and consistency rules
- [Sandcastle Help File Builder](#)
 - Provides graphical and command line based tools to build a help file in an automated fashion

COM, No PIA, Optional Parameters, etc.

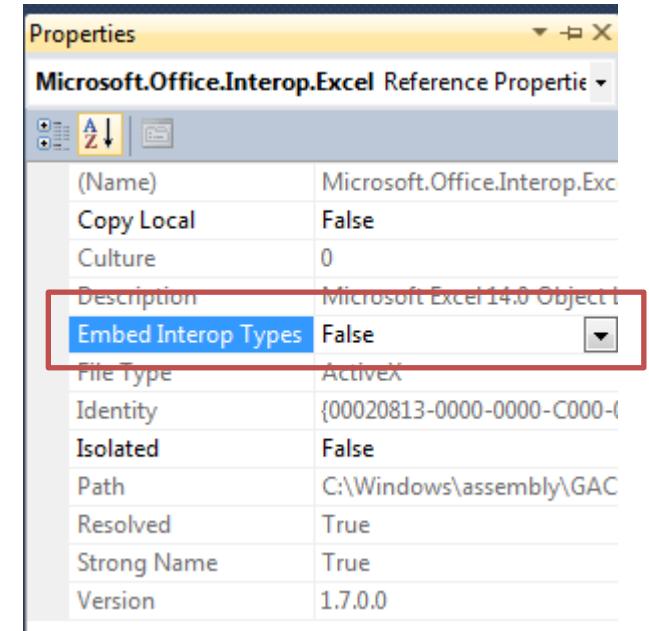
OFFICE INTEGRATION

Why Has Office Integration Been Hard?

- Primary Interop Assemblies
 - Generated with Tlbimp.exe
 - Assembly with runtime metadata
- Everyone can generate his own PIA
 - Problem: Unique set of types
 - Not compatible between developers (i.e. creators)
- Solution: COM creator provides PIA together with COM component
- Big versioning crap...

Solution: Embedded Interop Types

- False
 - Include PIA for each version of Office
- True
 - Compiler embeds type information from interop assembly (only used parts)
 - Runs with different versions of Microsoft Office ☺

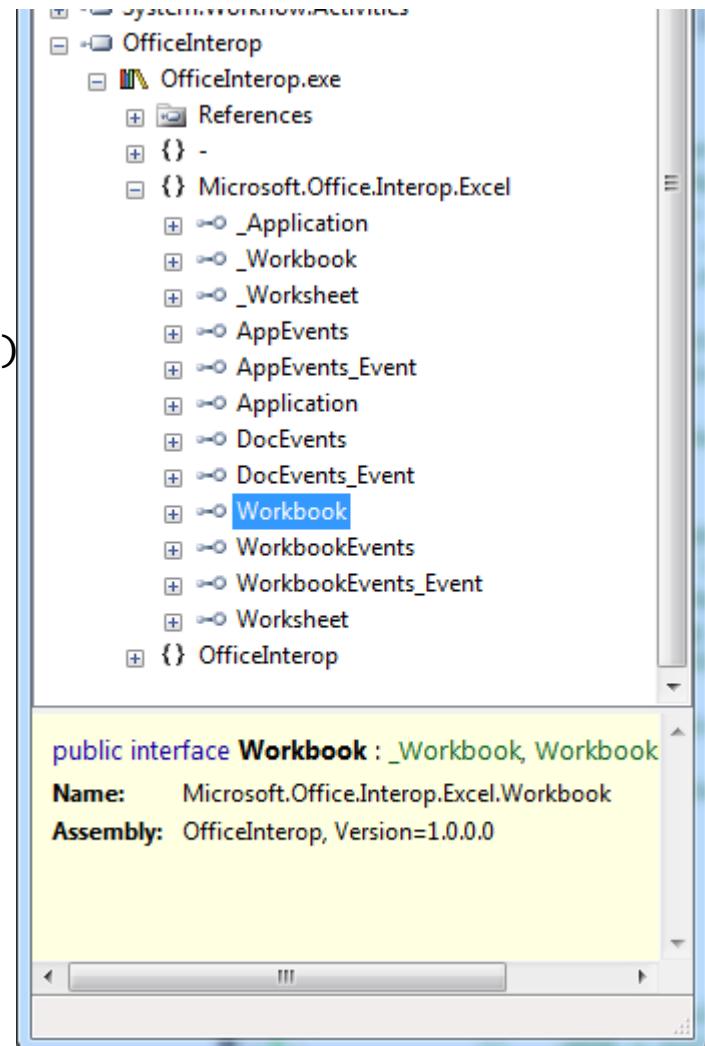


Embedded Interop Types

```
static void Main(string[] args)
{
    int[] values = {4, 6, 18, 2,
                    1, 76, 0, 3, 11};

    CreateWorkbook(values,
                    @"C:\SampleFolder\SampleWorkbook.xls")
}

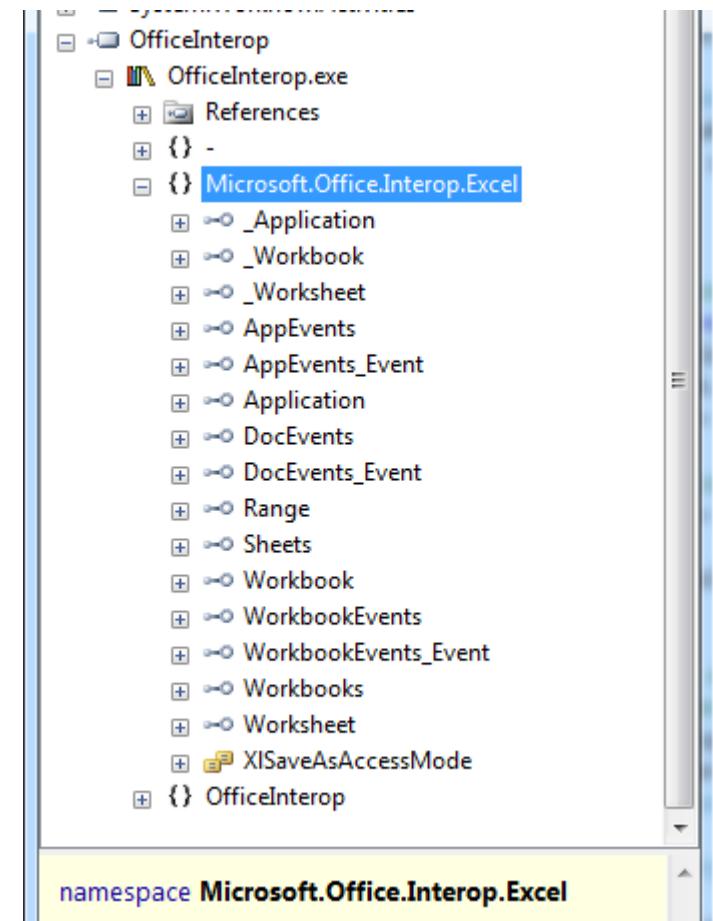
static void CreateWorkbook(int[] values,
                          string filePath)
{
    Excel.Application excelApp = null;
    Excel.Workbook wkbk;
    Excel.Worksheet sheet;
}
```



Embedded Interop Types

```
static void CreateWorkbook(int[] values, string filePath)
{
    Excel.Application excelApp = null;
    Excel.Workbook wkbk;
    Excel.Worksheet sheet;

    try
    {
        excelApp = new Excel.Application();
        wkbk = excelApp.Workbooks.Add();
        sheet = wkbk.Sheets.Add() as
            Excel.Worksheet;
        [...]
        wkbk.SaveAs(filePath);
    }
    catch
    {
    }
    finally
    {
        [...]
    }
}
```



Why Has Office Integration Been Hard?

- Pre C# 4
 - No optional parameters
 - No named parameters
- Hard to interact with COM libraries
- C# 4
 - Optional parameters
 - Named parameters

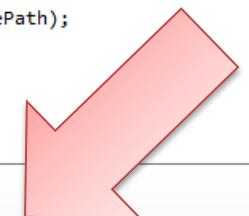
```
Excel.Application excelApp = null;
Excel.Workbook wkbk;
Excel.Worksheet sheet;

try
{
    // Start Excel and create a workbook and worksheet.
    excelApp = new Excel.Application();
    wkbk = excelApp.Workbooks.Add();
    sheet = wkbk.Sheets.Add() as Excel.Worksheet;
    sheet.Name = "Sample Worksheet";

    // Write a column of values.
    for (int i = 1; i < values.Length; i++)
    {
        sheet.Cells[i, 1] = values[i];
    }

    // Suppress any alerts and save the file. Create the directory
    // if it does not exist. Overwrite the file if it exists.
    excelApp.DisplayAlerts = false;
    string filePath = Path.GetDirectoryName(filePath);
    if (!Directory.Exists(filePath))
    {
        Directory.CreateDirectory(filePath);
    }
    wkbk.SaveAs(filePath, |

} catch [object] _Workbook.SaveAs([object Filename = Type.Missing],
{ [object FileFormat = Type.Missing],
} [object Password = Type.Missing],
} [object WriteResPassword = Type.Missing],
} [object ReadOnlyRecommended = Type.Missing],
} [object CreateBackup = Type.Missing],
} [XlSaveAsAccessMode AccessMode = XlSaveAsAccessMode.xlNoChange],
} [object ConflictResolution = Type.Missing],
} [object AddToMru = Type.Missing],
} [object TextCodepage = Type.Missing],
} [object TextVisualLayout = Type.Missing],
} [object Local = Type.Missing])
```



Be Careful With Default Values

```
using System;  
  
namespace OptionalParameters  
{  
    class Program  
    {  
        public static void DoSomething(int x = 17)  
        {  
            Console.WriteLine(x);  
        }  
  
        static void Main()  
        {  
            DoSomething();  
        }  
    }  
}
```

- Versioning problem
 Default value in calling code
- Not CLS Compliant
- Member overloading sometimes better

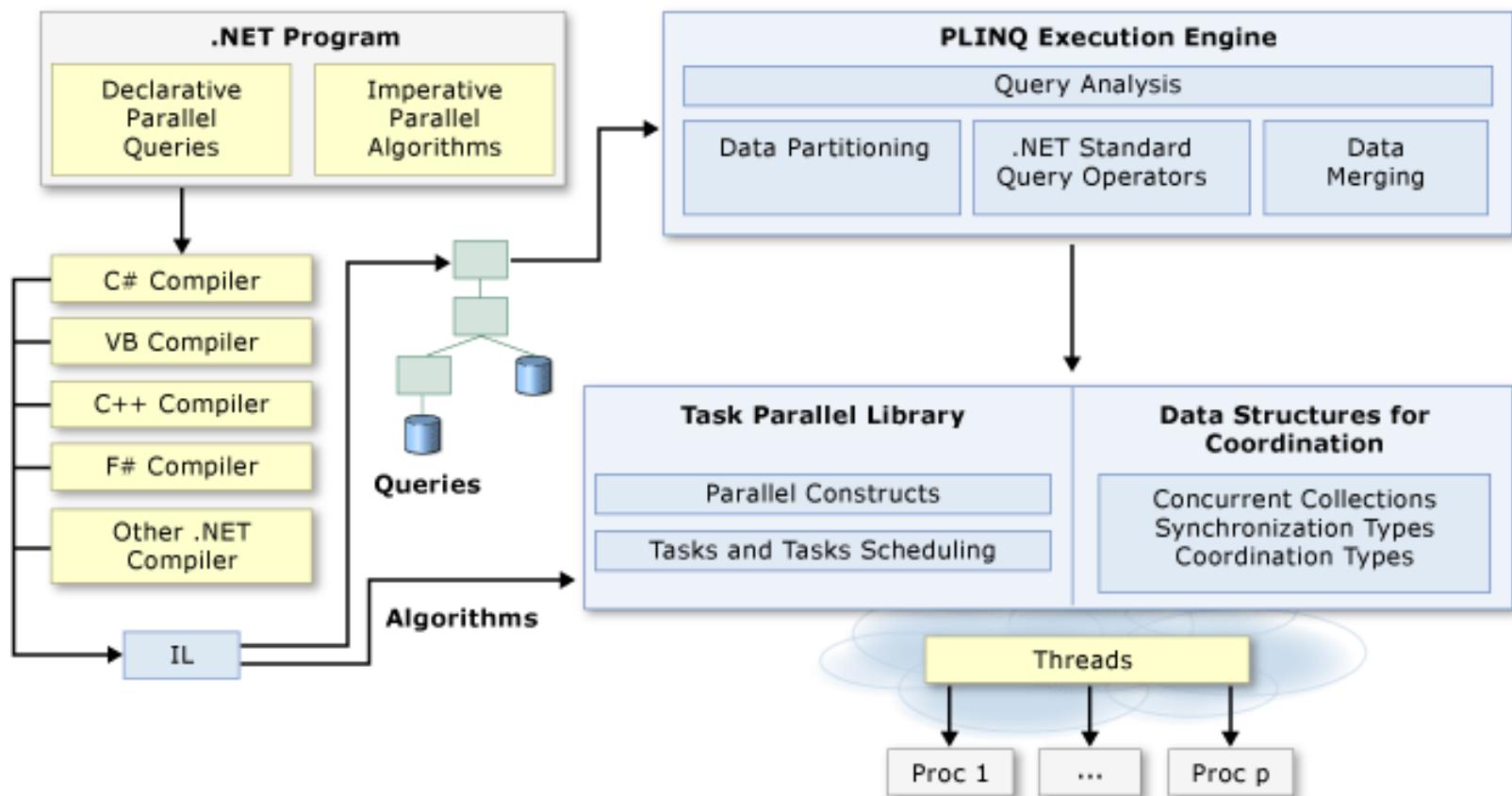


```
.method [...] static void Main() cil managed  
{  
    .entrypoint  
    .maxstack 8  
    ldc.i4.s 0x11  
    call void OptionalParameters.Program  
          ::DoSomething(int32)  
    ret  
}
```

Ready for the Many/Multicore revolution...

PARALLELE PROGRAMMIERUNG

What's New In C#/.NET 4



Was läuft hier falsch? (Code)

```
public static void MyParallelFor(  
    int inclusiveLowerBound, int exclusiveUpperBound, Action<int> body)  
{  
    int size = exclusiveUpperBound - inclusiveLowerBound;  
    int numProcs = Environment.ProcessorCount;  
    int range = size / numProcs;  
  
    int remaining = numProcs;  
    using (ManualResetEvent mre = new ManualResetEvent(false))  
    {  
        for (int p = 0; p < numProcs; p++)  
        {  
            int start = p * range + inclusiveLowerBound;  
            int end = (p == numProcs - 1) ? exclusiveUpperBound : start + range;  
            ThreadPool.QueueUserWorkItem(delegate {  
                for (int i = start; i < end; i++) body(i);  
                if (Interlocked.Decrement(ref remaining) == 0) mre.Set();  
            });  
        }  
        mre.WaitOne();  
    }  
}
```

Generell: Warum muss das jedes Mal neu erfunden werden??



Anteil des Synchronisierungsaufwands
bei kurzen Aufgaben sehr hoch

Multithreading

Pre .NET 4

- System.Threading Namespace
- Thread Klasse
- ThreadPool Klasse

.NET 4

- System.Threading.Tasks Namespace
- Task und Task<TResult> Klassen
- TaskFactory Klasse
- Parallel Klasse

Kurzer Überblick über Tasks

- **Starten**
 - Parallel.Invoke(...)
 - Task.Factory.StartNew(...)
- **Warten**
 - myTask.Wait()
 - Task.WaitAll
 - Task.WaitAny
 - Task.Factory.ContinueWhenAll(...)
 - Task.Factory.ContinueWhenAny(...)
- **Verknüpfen**
 - Task.Factory.StartNew(...,
TaskCreationOptions.AttachedToParent);
- **Abbrechen**
 - Cancellation Tokens

Nicht in Silverlight ☹



Schleifen - Parallel.For

```
var source = new double[Program.Size];
var destination = new double[Program.Size];

Console.WriteLine(MeasuringTools.Measure(() => {
    for (int i = 0; i < Program.Size; i++) {
        source[i] = (double)i;
    }

    for (int i = 0; i < Program.Size; i++) {
        destination[i] = Math.Pow(source[i], 2);
    }
}));

Console.WriteLine(MeasuringTools.Measure(() => {
    Parallel.For(0, Program.Size, (i) => source[i] = (double)i);
    Parallel.For(0, Program.Size,
        (i) => destination[i] = Math.Pow(source[i], 2));
}));
```

Schleifen - Parallel.For

- Unterstützung für Exception Handling
- Break und Stop Operationen
 - Stop: Keine weiteren Iterationen
 - Break: Keine Iterationen nach dem aktuellen Index mehr
 - Siehe dazu auch ParallelLoopResult
- Int32 und Int64 Laufvariablen
- Konfigurationsmöglichkeiten (z.B. Anzahl an Threads)
- Schachtelbar
 - Geteilte Threading-Ressourcen
- Effizientes Load Balancing
- U.v.m.

Nicht selbst entwickeln!

Schleifen - Parallel.ForEach

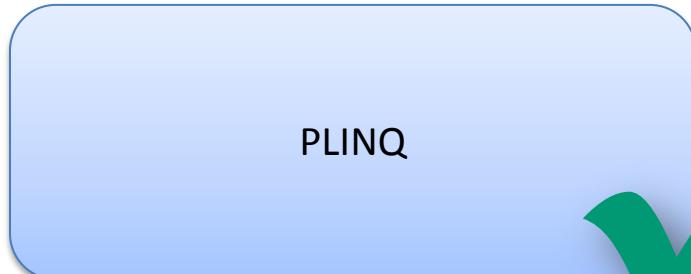
```
Console.WriteLine(
    "Serieller Durchlauf mit foreach: {0}",
    MeasuringTools.Measure(() =>
{
    double sumOfSquares = 0;
    foreach (var square in Enumerable.Range(0, Program.Size).Select(
        i => Math.Pow(i, 2)))
    {
        sumOfSquares += square;
    }
}));
```

```
Console.WriteLine(
    "Paralleler Durchlauf mit foreach: {0}",
    MeasuringTools.Measure(() =>
{
    double sumOfSquares = 0;
    Parallel.ForEach(Enumerable.Range(0, Program.Size)
        .Select(i => Math.Pow(i, 2)), square => sumOfSquares += square);
}));
```

Hoher Aufwand für abgesicherten
Zugriff auf MoveNext/Current
→ Parallele Version oft langsamer

Schleifen - Parallel.ForEach

```
Console.WriteLine(
    "Paralleler Durchlauf mit PLINQ: {0}",
    MeasuringTools.Measure(() =>
{
    double sumOfSquares = 0;
    sumOfSquares = ParallelEnumerable
        .Range(0, Program.Size)
        .AsOrdered()
        .Select(i => Math.Pow(i, 2))
        .Sum();
}));
```



PLINQ



Von LINQ zu PLINQ

LINQ

```
var result = source  
    .Where(...)  
    .Select(...)
```

PLINQ

```
var result = source  
    .AsParallel()  
    .Where(...)  
    .Select(...)
```

Aus `IEnumerable` wird
`ParallelQuery`

Tipp: `AsOrdered()` erhält die
Sortierreihenfolge

Performancetipps für PLINQ

- Allokieren von Speicher in parallelem Lambdaausdruck vermeiden
 - Sonst kann Speicher + GC zum Engpass werden
 - Wenn am Server: [Server GC](#)
- [False Sharing](#) vermeiden
- Bei zu kurzen Delegates ist Koordinationsaufwand für Parallelisierung oft höher als Performancegewinn
 - → Expensive Delegates
 - Generell: Auf richtige Granularität der Delegates achten
- AsParallel() kann an jeder Stelle im LINQ Query stehen
 - → Teilweise serielle, teilweise parallele Ausführung möglich
- Über Environment.ProcessorCount kann Anzahl an Kernen ermittelt werden
- **Messen, Messen, Messen!**

Was läuft hier falsch? (Code)

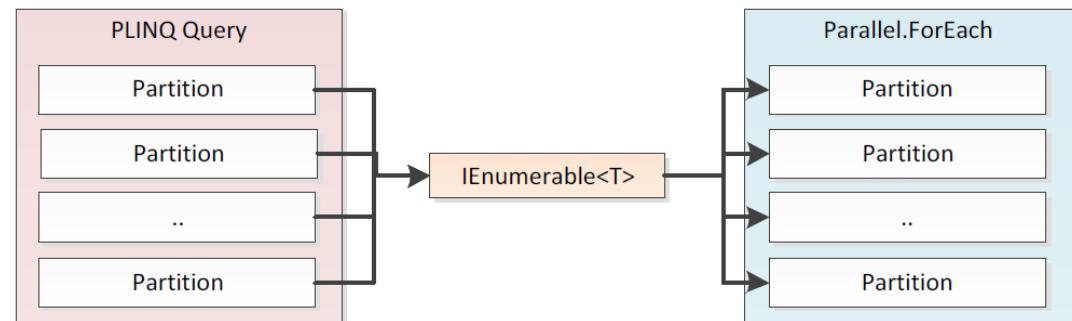
```

var result = new List<double>();
Console.WriteLine(
    "Paralleler Durchlauf mit Parallel.ForEach: {0}",
    MeasuringTools.Measure(() =>
{
    Parallel.ForEach(
        source.AsParallel(),
        i =>
    {
        if (i % 2 == 0)
        {
            lock (result)
            {
                result.Add(i);
            }
        }
    });
}));

```



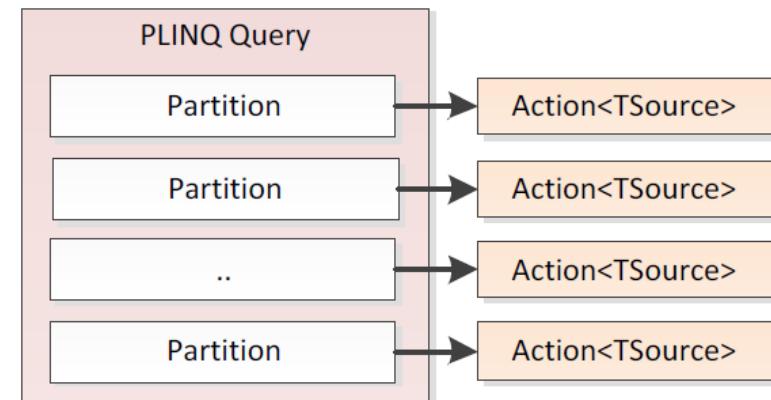
Parallel.ForEach verwendet
IEnumerable<T> → unnötige
Merge-Schritte



Was läuft hier falsch? (Code)

```
Console.WriteLine(  
    "Paralleler Durchlauf mit Parallel.ForEach: {0}",  
MeasuringTools.Measure(() =>  
{  
    source.AsParallel().ForEach(  
        i =>  
        {  
            if (i % 2 == 0)  
            {  
                lock (result)  
                {  
                    result.Add(i);  
                }  
            }  
        }  
    );  
}));
```

Lock-free Collection wäre
überlegenswert!



Was läuft hier falsch? (Code)

```
Console.WriteLine(
    "Serielles Lesen: {0}",
MeasuringTools.Measure(() =>
{
    foreach (var url in urls)
    {
        var request = WebRequest.Create(url);
        using (var response = request.GetResponse())
        {
            using (var stream = response.GetResponseStream())
            {
                var content = new byte[1024];
                while (stream.Read(content, 0, 1024) != 0) ;
            }
        }
    }
}));
```



Optimal für Parallelisierung selbst
bei einem Core (IO-Bound Waits)

Was läuft hier falsch? (Code)

```
Console.WriteLine(
    "Paralleles Lesen: {0}",
MeasuringTools.Measure(() =>
{
    Parallel.ForEach(urls, url =>
    {
        var request = WebRequest.Create(url);
        using (var response = request.GetResponse())
        {
            using (var stream = response.GetResponseStream())
            {
                var content = new byte[1024];
                while (stream.Read(content, 0, 1024) != 0) ;
            }
        }
    });
});
```



Anzahl Threads = Anzahl Cores;
könnte mehr sein, da IO-Bound
waits

```
Parallel.ForEach(
    urls,
    new ParallelOptions() { MaxDegreeOfParallelism = urls.Length },
    url => { ... });
```

Was läuft hier falsch? (Code)

```
Console.WriteLine(
    "Paralleles Lesen: {0}",
MeasuringTools.Measure(() =>
{
    urls.AsParallel().WithDegreeOfParallelism(urls.Length)
        .Select(url => WebRequest.Create(url))
        .Select(request => request.GetResponse())
        .Select(response => new {
            Response = response,
            Stream = response.GetResponseStream() })
        .ForAll(stream =>
    {
        var content = new byte[1024];
        while (stream.Stream.Read(content, 0, 1024) != 0) ;
        stream.Stream.Dispose();
        stream.Response.Close();
    });
}));
```



OK für Client, tödlich für Server!
Wenn Anzahl gleichzeitiger User wichtig ist sind
andere Lösungen vorzuziehen.

Was läuft hier falsch? (Code)

```
Console.WriteLine(
    "Paralleles Lesen mit TaskFactory: {0}",
    MeasuringTools.Measure(() =>
{
    var tasks = new Task[url.Length];
    for (int i = 0; i < url.Length; i++)
    {
        tasks[i] = Task.Factory.StartNew(() => Readurl(url[i]));
    }

    Task.WaitAll(tasks);
})
);

...
private static void Readurl(object url)
{
    ...
}
```



Delegate verwendet Wert von i aus
dem Main Thread →
IndexOutOfRangeException

Was läuft hier falsch? (Code)

```
// Variante 1
...
var tasks = new Task[url.Length];
for (int i = 0; i < url.Length; i++)
{
    var tmp = i;
    tasks[i] = Task.Factory.StartNew(() => Readurl(urls[tmp]));
}
...
...
```

Durch lokale Variable wird delegate unabhängig;
mehr zum Thema unter dem Schlagwort *Closures*

```
// Variante 2
var tasks = new Task[url.Length];
for (int i = 0; i < url.Length; i++)
{
    tasks[i] = Task.Factory.StartNew(Readurl, urls[i]);
}
```

State object wird an delegate übergeben



Producer/Consumer

Was läuft hier falsch? (Code)

```
var buffer = new Queue<long>();
var cancelTokenSource = new CancellationTokenSource();
var done = false;

var producer = Task.Factory.StartNew((cancelTokenObj) => {
    var counter = 10000000;
    var cancelToken = (CancellationToken)cancelTokenObj;
    try {
        while (!cancelToken.IsCancellationRequested && counter-- > 0) {
            // Here we get some data (e.g. reading it from a file)
            var value = DateTime.Now.Ticks;
            // Write it to buffer with values that have to be processed
            buffer.Enqueue(value);
        }
    } finally {
        done = true;
    }
}, cancelTokenSource.Token);
```



buffer wird nicht gelockt

Producer/Consumer

Was läuft hier falsch? (Code)

```
var consumer = Task.Factory.StartNew((cancelTokenObj) =>
{
    var cancelToken = (CancellationToken)cancelTokenObj;
    while (!cancelToken.IsCancellationRequested && !done)
    {
        // Get the next value to process
        lock (buffer)
        {
            var value = buffer.Dequeue();
        }

        // Here we do some expensive processing
        Thread.SpinWait(1000);
    }
}, cancelTokenSource.Token);
```

Prüfung ob leer fehlt

Consumer ist viel langsamer als
Producer → Producer überschwemmt
Consumer mit Daten

Collections für parallele Programmierung

- System.Collections.Concurrent für Thread-Safe Collections
 - BlockingCollection<T>
Blocking und Bounding-Funktionen
 - ConcurrentDictionary<T>
 - ConcurrentQueue<T>
 - ConcurrentStack<T>
 - ConcurrentBag<T>
- Optimal zur Umsetzung von Pipelines
 - Datei wird gelesen, gepackt, verschlüsselt, geschrieben

Producer/Consumer

Was läuft hier falsch? (Code)

```
var buffer = new BlockingCollection<long>(10);
var cancelTokenSource = new CancellationTokenSource();

var producer = Task.Factory.StartNew((cancelTokenObj) => {
    var counter = 10000000;
    var cancelToken = (CancellationToken)cancelTokenObj;
    try {
        while (!cancelToken.IsCancellationRequested && counter-- > 0) {
            // Here we get some data (e.g. reading it from a file)
            var value = DateTime.Now.Ticks;
            // Write it to the buffer with values that have to be processed
            buffer.Add(value);
        }
    }
    finally {
        buffer.CompleteAdding();
    }
}, cancelTokenSource.Token);
```



Producer/Consumer

Was läuft hier falsch? (Code)

```
var consumer = Task.Factory.StartNew((cancelTokenObj) =>
{
    var cancelToken = (CancellationToken)cancelTokenObj;
    foreach (var value in buffer.GetConsumingEnumerable())
    {
        if ( cancelToken.IsCancellationRequested )
        {
            break;
        }

        // Here we do some expensive procesing
        Thread.SpinWait(1000);
    }
}, cancelTokenSource.Token);
```

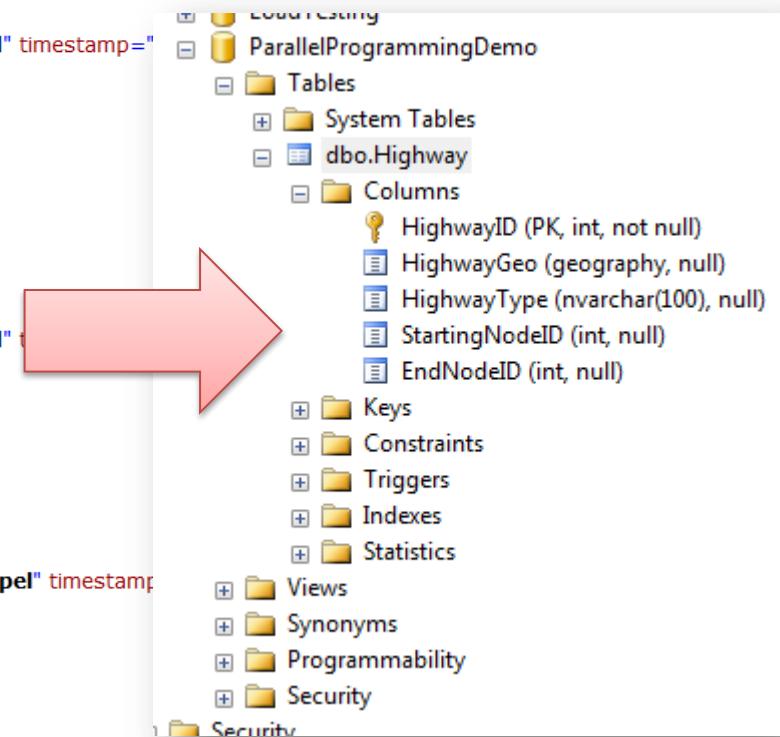


Mission Impossible?

```

<?xml version="1.0" encoding="UTF-8" ?>
- <osm version="0.6" generator="pbf2osm">
  <node id="172539" lat="52.5651847" lon="13.3354546" version="9" changeset="5702878" user="Woschl" uid="121042" timestamp="2010-09-06T21:0
  <node id="172540" lat="52.5647252" lon="13.3364064" version="7" changeset="5702878" user="Woschl" uid="121042" timestamp="2010-09-06T21:0
  <node id="172541" lat="52.5655270" lon="13.3362226" version="2" changeset="728814" user="bahnpirat" uid="13203" timestamp="2009-03-03T14:1
  <node id="172542" lat="52.5660003" lon="13.3375554" version="3" changeset="728814" user="bahnpirat" uid="13203" timestamp="2009-03-03T14:1
  <node id="172543" lat="52.5663124" lon="13.3394369" version="4" changeset="3410834" user="toaster" uid="10549" timestamp="2009-12-20T01:32
  <node id="172544" lat="52.5666165" lon="13.3432402" version="5" changeset="3410834" user="toaster" uid="10549" timestamp="2009-12-20T01:32
  - <node id="172545" lat="52.5670070" lon="13.3466339" version="5" changeset="5701736" user="Woschl" uid="121042" timestamp="2010-09-06T19:1
    <tag k="highway" v="traffic_signals" />
  </node>
- <way id="30770007" version="2" changeset="2121805" uid="6669" user="Elwood" timestamp="2010-09-06T21:0
  <nd ref="172539" />
  <nd ref="172540" />
  <nd ref="172541" />
  <nd ref="172542" />
  <tag k="access" v="permissive" />
  <tag k="highway" v="residential" />
  <tag k="maxspeed" v="5" />
  <tag k="name" v="Wolkenburgweg" />
  <tag k="postal_code" v="14169" />
</way>
- <way id="30770008" version="3" changeset="2121805" uid="6669" user="Elwood" timestamp="2010-09-06T21:0
  <nd ref="172542" />
  <nd ref="172543" />
  <tag k="access" v="permissive" />
  <tag k="highway" v="residential" />
  <tag k="maxspeed" v="5" />
  <tag k="name" v="Lohrbergweg" />
  <tag k="postal_code" v="14169" />
</way>
- <way id="30770010" version="1" changeset="99086" uid="72235" user="Basstoelpel" timestamp="2010-09-06T21:0
  <nd ref="172544" />
  <nd ref="172545" />
  <tag k="highway" v="footway" />
</way>
</osm>

```

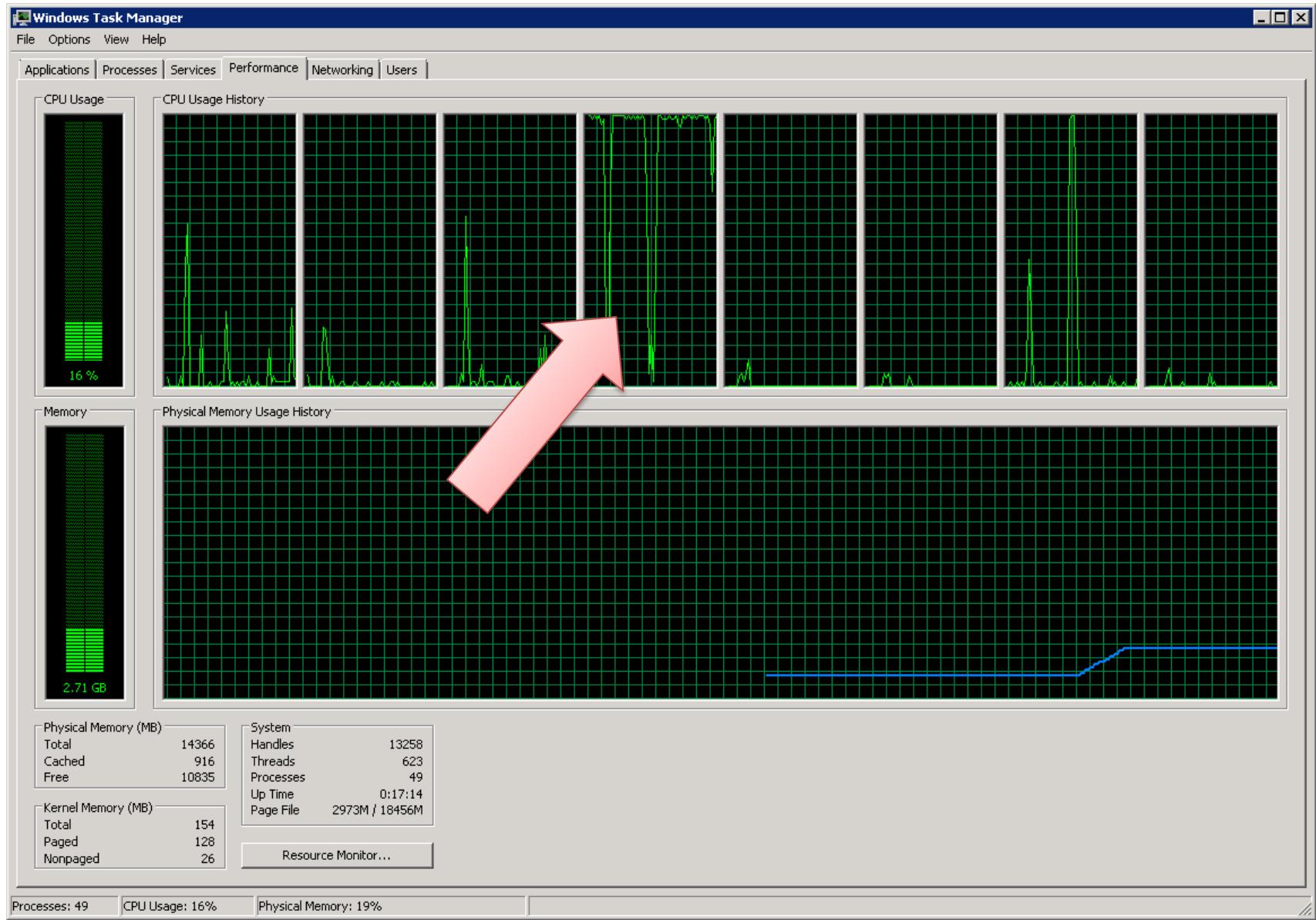


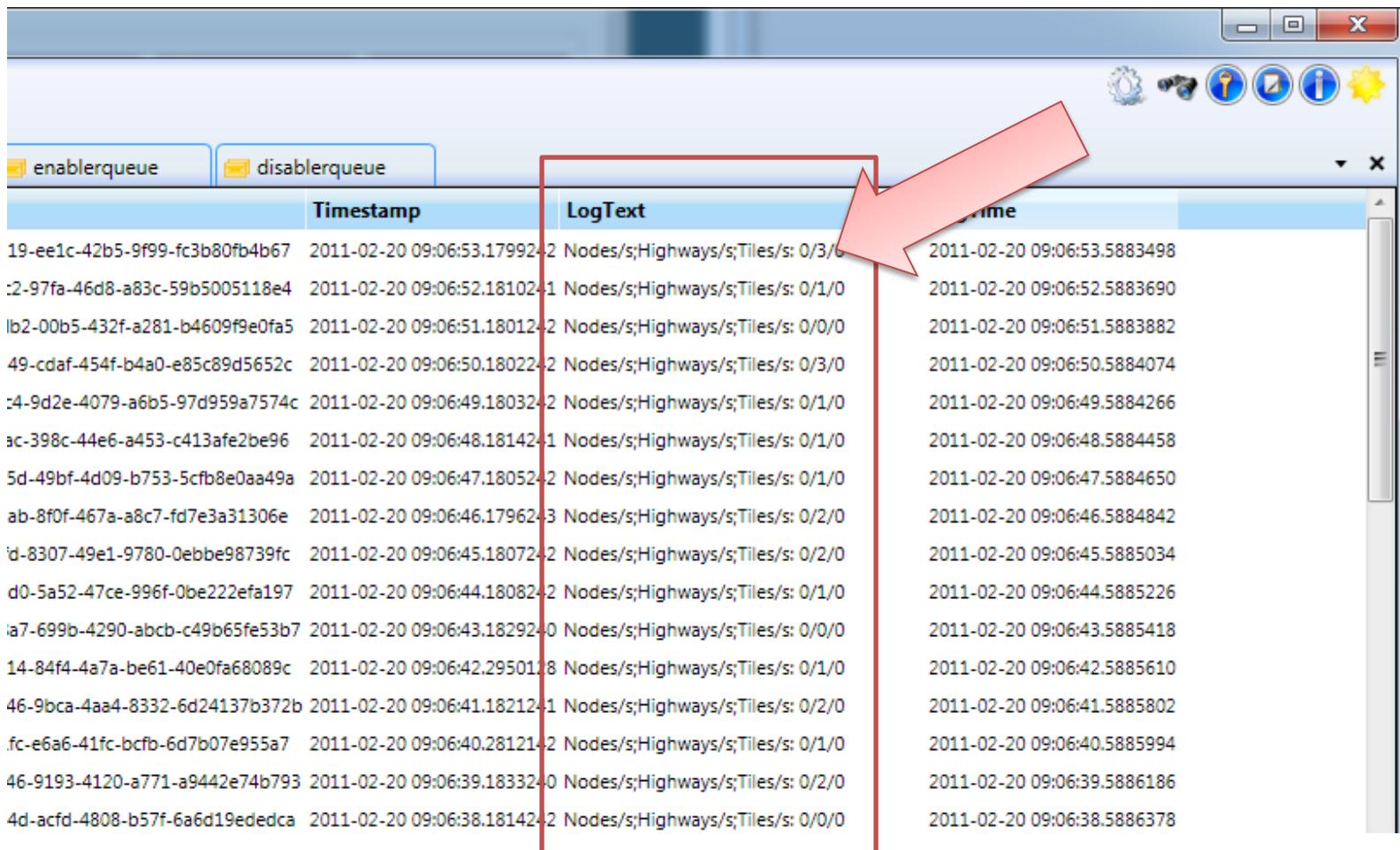
Mission Impossible

- Import large XML files with geodata into SQL Server
 - Smallest 200MB
- Download via http
- LARGE Server
 - 8 cores, ~15GB Ram, ~2TB Disc
 - Running in the cloud (Azure)

Solution 1: The One-Liner

```
XElement tmpNode;
XDocument doc;
foreach (var row in
(doc = XDocument.Load(@"https://loadtesting.blob.core.windows.net/osm/berlin.xml"))
.Descendants("way")
.Where(w => w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").Count() > 0)
.Select(w =>
    new
    {
        WayId = w.Attribute("id").Value,
        WayType = w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").First().Attribute("v").Value,
        Linestring = "LINESTRING(" + w.Descendants("nd")
            .Aggregate< XElement, string>(string.Empty, (agg, node) =>
                agg
                + (agg.Length != 0 ? "," : string.Empty)
                + (tmpNode = doc.Root.Descendants("node")
                    .Where(n => n.Attribute("id").Value == node.Attribute("ref").Value).First())
                    .Attribute("lat").Value
                + " " + tmpNode.Attribute("lon").Value) + ")",
        StartingNodeId = w.Descendants("nd").First().Attribute("ref").Value,
        EndNodeId = w.Descendants("nd").Last().Attribute("ref").Value
    })
{
    Write row to database
}
```



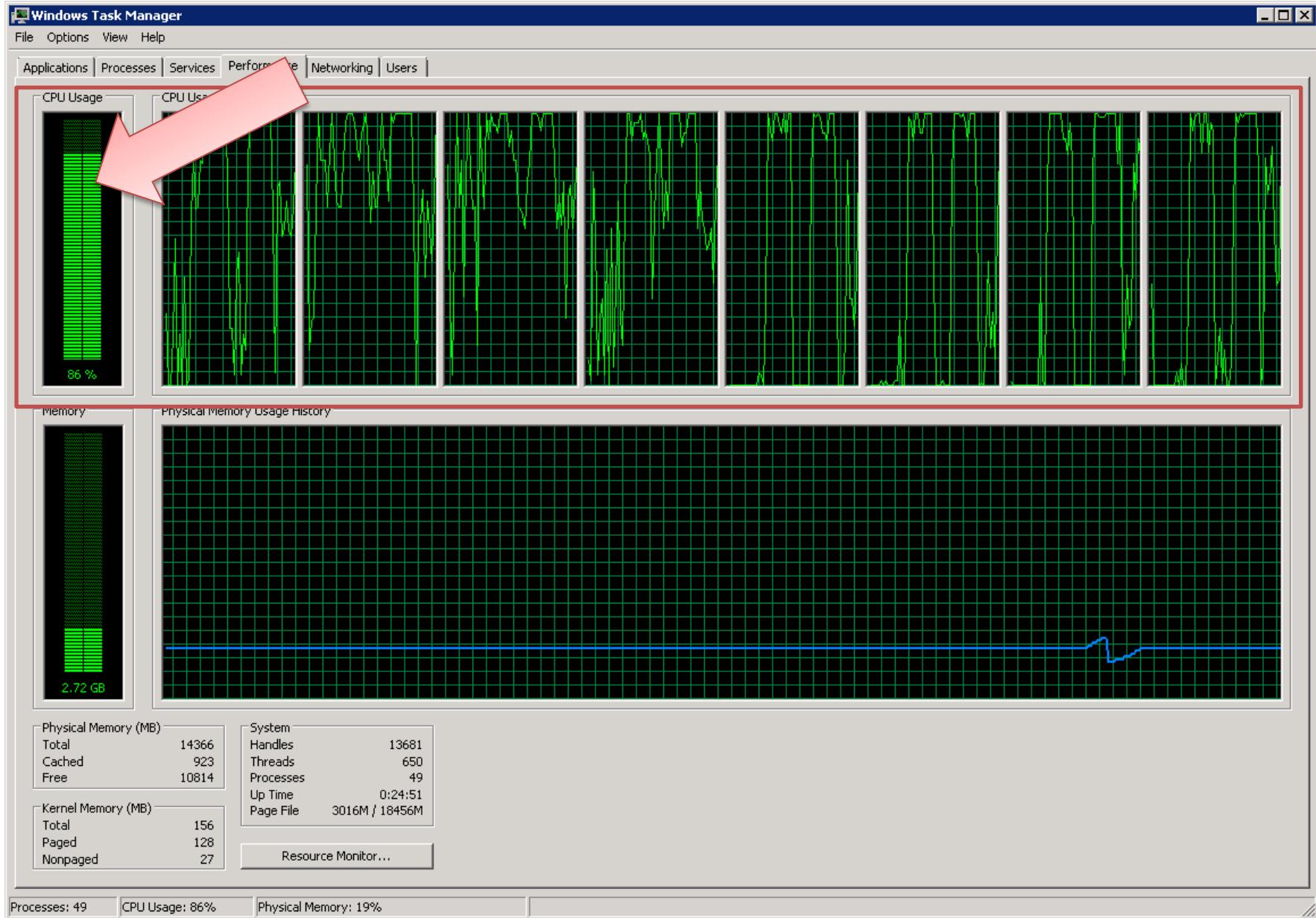


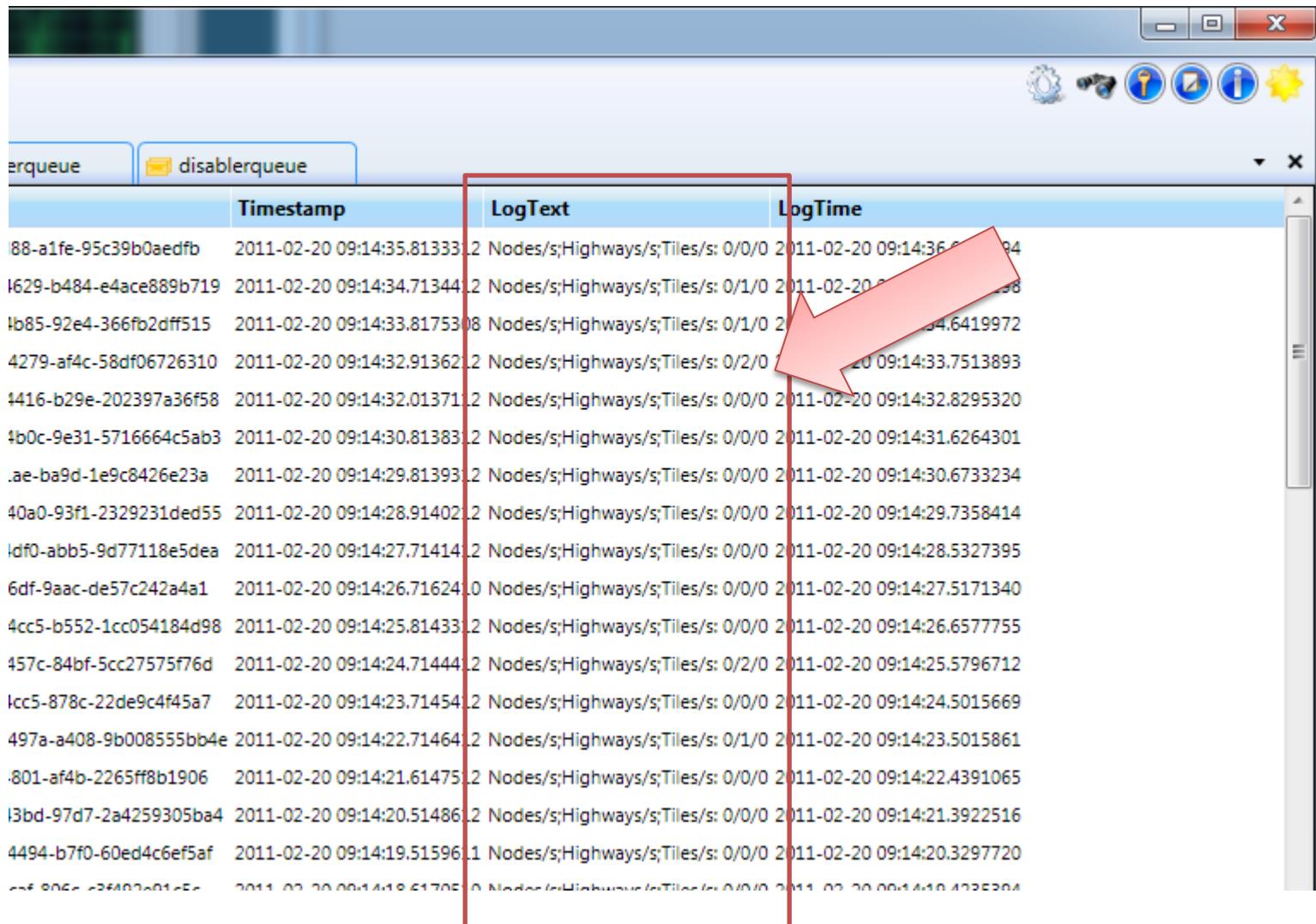
The screenshot shows a Windows application window with a title bar containing icons for minimize, maximize, and close. Below the title bar is a toolbar with several icons. The main area of the window contains two tabs: "enablerqueue" (selected) and "disablerqueue". Under each tab, there is a table with columns: "Timestamp" and "LogText". The "LogText" column is highlighted with a red border and has a red arrow pointing to its header. The data in the table is as follows:

Timestamp	LogText
19-ee1c-42b5-9f99-fc3b80fb4b67	2011-02-20 09:06:53.17992-2 Nodes/s;Highways/s;Tiles/s: 0/3/0
:2-97fa-46d8-a83c-59b5005118e4	2011-02-20 09:06:52.18102-1 Nodes/s;Highways/s;Tiles/s: 0/1/0
lb2-00b5-432f-a281-b4609f9e0fa5	2011-02-20 09:06:51.18012-2 Nodes/s;Highways/s;Tiles/s: 0/0/0
49-cdaf-454f-b4a0-e85c89d5652c	2011-02-20 09:06:50.18022-2 Nodes/s;Highways/s;Tiles/s: 0/3/0
:4-9d2e-4079-a6b5-97d959a7574c	2011-02-20 09:06:49.18032-2 Nodes/s;Highways/s;Tiles/s: 0/1/0
ac-398c-44e6-a453-c413afe2be96	2011-02-20 09:06:48.18142-1 Nodes/s;Highways/s;Tiles/s: 0/1/0
5d-49bf-4d09-b753-5cfb8e0aa49a	2011-02-20 09:06:47.18052-2 Nodes/s;Highways/s;Tiles/s: 0/1/0
ab-8f0f-467a-a8c7-fd7e3a31306e	2011-02-20 09:06:46.17962-3 Nodes/s;Highways/s;Tiles/s: 0/2/0
'd-8307-49e1-9780-0ebbe98739fc	2011-02-20 09:06:45.18072-2 Nodes/s;Highways/s;Tiles/s: 0/2/0
d0-5a52-47ce-996f-0be222efa197	2011-02-20 09:06:44.18082-2 Nodes/s;Highways/s;Tiles/s: 0/1/0
ia7-699b-4290-abcb-c49b65fe53b7	2011-02-20 09:06:43.18292-0 Nodes/s;Highways/s;Tiles/s: 0/0/0
14-84f4-4a7a-be61-40e0fa68089c	2011-02-20 09:06:42.29501-8 Nodes/s;Highways/s;Tiles/s: 0/1/0
46-9bca-4aa4-8332-6d24137b372b	2011-02-20 09:06:41.18212-1 Nodes/s;Highways/s;Tiles/s: 0/2/0
.fc-e6a6-41fc-bcfc-6d7b07e955a7	2011-02-20 09:06:40.28121-2 Nodes/s;Highways/s;Tiles/s: 0/1/0
46-9193-4120-a771-a9442e74b793	2011-02-20 09:06:39.18332-0 Nodes/s;Highways/s;Tiles/s: 0/2/0
4d-acfd-4808-b57f-6a6d19ed6dca	2011-02-20 09:06:38.18142-2 Nodes/s;Highways/s;Tiles/s: 0/0/0
	2011-02-20 09:06:53.5883498
	2011-02-20 09:06:52.5883690
	2011-02-20 09:06:51.5883882
	2011-02-20 09:06:50.5884074
	2011-02-20 09:06:49.5884266
	2011-02-20 09:06:48.5884458
	2011-02-20 09:06:47.5884650
	2011-02-20 09:06:46.5884842
	2011-02-20 09:06:45.5885034
	2011-02-20 09:06:44.5885226
	2011-02-20 09:06:43.5885418
	2011-02-20 09:06:42.5885610
	2011-02-20 09:06:41.5885802
	2011-02-20 09:06:40.5885994
	2011-02-20 09:06:39.5886186
	2011-02-20 09:06:38.5886378

Solution 2: Making it Parallel

```
XElement tmpNode;
XDocument doc;
(doc = XDocument.Load(@"https://loadtesting.blob.core.windows.net/osm/berlin.xml"))
    .Descendants("way")
    .AsParallel()
    .Where(w => w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").Count() > 0)
    .Select(w =>
        new
        {
            WayId = w.Attribute("id").Value,
            WayType = w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").First().Attribute("v").Value,
            Linestring = "LINESTRING(" + w.Descendants("nd")
                .Aggregate< XElement, string>(string.Empty, (agg, node) =>
                    agg
                    + (agg.Length != 0 ? "," : string.Empty)
                    + (tmpNode = doc.Root.Descendants("node")
                        .AsParallel()
                        .Where(n => n.Attribute("id").Value == node.Attribute("ref").Value).First())
                        .Attribute("lat").Value
                    + " " + tmpNode.Attribute("lon").Value) + ")",
            StartingNodeId = w.Descendants("nd").First().Attribute("ref").Value,
            EndNodeId = w.Descendants("nd").Last().Attribute("ref").Value
        }
    )
    .ForAll(row =>
{
    Write row to database
});
```





	Timestamp	LogText	LogTime
88-a1fe-95c39b0aedfb	2011-02-20 09:14:35.813331	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:36.000000
f629-b484-e4ace889b719	2011-02-20 09:14:34.713441	2 Nodes/s;Highways/s;Tiles/s: 0/1/0	2011-02-20 09:14:34.713441
fb85-92e4-366fb2dff515	2011-02-20 09:14:33.8175308	2 Nodes/s;Highways/s;Tiles/s: 0/1/0	2011-02-20 09:14:34.6419972
4279-af4c-58df06726310	2011-02-20 09:14:32.913621	2 Nodes/s;Highways/s;Tiles/s: 0/2/0	2011-02-20 09:14:33.7513893
f416-b29e-202397a36f58	2011-02-20 09:14:32.013711	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:32.8295320
fb0c-9e31-5716664c5ab3	2011-02-20 09:14:30.813831	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:31.6264301
.ae-ba9d-1e9c8426e23a	2011-02-20 09:14:29.813931	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:30.6733234
40a0-93f1-2329231ded55	2011-02-20 09:14:28.914021	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:29.7358414
fdf0-abb5-9d77118e5dea	2011-02-20 09:14:27.714141	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:28.5327395
6df-9aac-de57c242a4a1	2011-02-20 09:14:26.716241	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:27.5171340
4cc5-b552-1cc054184d98	2011-02-20 09:14:25.814331	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:26.6577755
457c-84bf-5cc27575f76d	2011-02-20 09:14:24.714441	2 Nodes/s;Highways/s;Tiles/s: 0/2/0	2011-02-20 09:14:25.5796712
fcc5-878c-22de9c4f45a7	2011-02-20 09:14:23.714541	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:24.5015669
497a-a408-9b008555bb4e	2011-02-20 09:14:22.714641	2 Nodes/s;Highways/s;Tiles/s: 0/1/0	2011-02-20 09:14:23.5015861
.801-af4b-2265ff8b1906	2011-02-20 09:14:21.614751	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:22.4391065
f3bd-97d7-2a4259305ba4	2011-02-20 09:14:20.514861	2 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:21.3922516
4494-b7f0-60ed4c6ef5af	2011-02-20 09:14:19.515961	1 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:20.3297720
c9f-80c4-428402601254	2011-02-20 09:14:18.517051	0 Nodes/s;Highways/s;Tiles/s: 0/0/0	2011-02-20 09:14:18.4225204

Solution 3: Enhanced LINQ

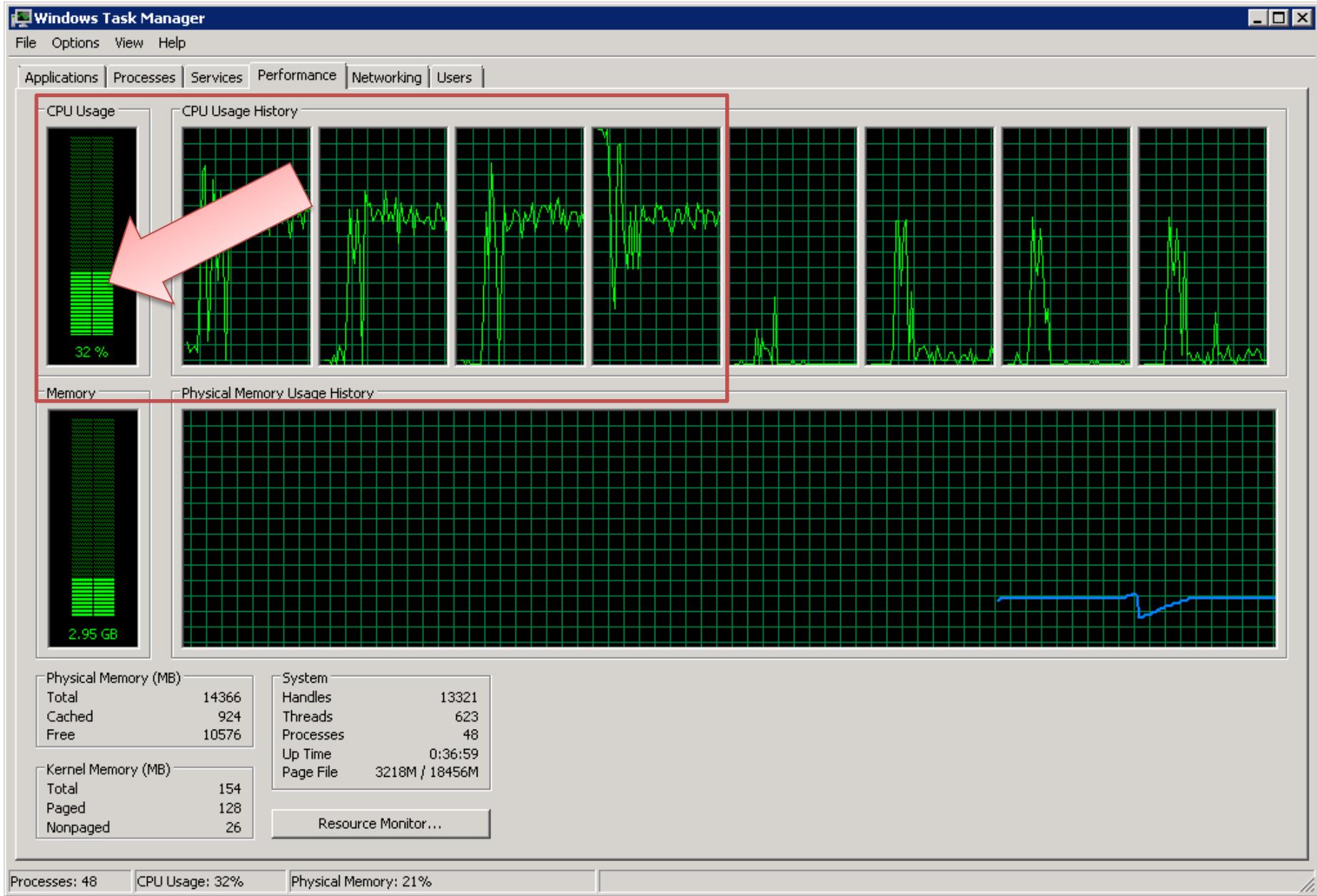
```

var nodes = new ConcurrentDictionary<string, string>();
var doc = XDocument.Load(@"https://loadtesting.blob.core.windows.net/osm/berlin.xml");

doc.Root.Descendants("node")
    .Select(n => new Tuple<string, string>(n.Attribute("id").Value, string.Format("{0} {1}", n.Attribute("lat").Value, n.Attribute("lon").Value)))
    .AsParallel()
    .ForAll(n =>
    {
        nodes.AddOrUpdate(n.Item1, n.Item2, (id, p) => p);
        lock (this.statisticsLockObject)
        {
            this.nodesPerSecond++;
        }
    });
}

using (var context = new GeoWriterContext())
{
    doc.Descendants("way")
        .AsParallel()
        .Where(w => w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").Count() > 0)
        .Select(w =>
        {
            new
            {
                WayId = w.Attribute("id").Value,
                WayType = w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").First().Attribute("v").Value,
                Linestring = "LINESTRING(" + w.Descendants("nd")
                    .Aggregate< XElement, string>(string.Empty, (agg, node) =>
                    {
                        agg += (agg.Length != 0 ? "," : string.Empty)
                            + nodes[node.Attribute("ref").Value] + ")";
                    })
                    .StartingNodeId = w.Descendants("nd").First().Attribute("ref").Value,
                    EndNodeId = w.Descendants("nd").Last().Attribute("ref").Value
            });
        });
    .ForAll(row =>
    {
        Write row to database
    });
}

```



b4a03 2011-02-20 11:06:43.6175922 Nodes/s;Highways/s;Tiles/s: 0/204/0
382a1 2011-02-20 11:06:42.7176822 Nodes/s;Highways/s;Tiles/s: 0/202/0
912c0 2011-02-20 11:06:42.5686971 Exception Message: 24117: The LineString input is not valid because it doe
l2117 2011-02-20 11:06:42.5177022 Could not write way 4402591: LINESTRING(52.3869551 13.1543742)
lc1503 2011-02-20 11:06:41.6177922 Nodes/s;Highways/s;Tiles/s: 0/203/0
2a5d70 2011-02-20 11:06:40.6178922 Nodes/s;Highways/s;Tiles/s: 0/198/0
i5ad6 2011-02-20 11:06:39.6179922 Nodes/s;Highways/s;Tiles/s: 0/75/0
f6c62 2011-02-20 11:06:39.2170323 Nodes/s;Highways/s;Tiles/s: 0/17/0
26bf5 2011-02-20 11:06:38.2181322 Nodes/s;Highways/s;Tiles/s: 52/0/0
fb1bf 2011-02-20 11:06:38.0181522 Nodes/s;Highways/s;Tiles/s: 0/0/0
f9b93f 2011-02-20 11:06:37.2192321 Nodes/s;Highways/s;Tiles/s: 970248/0/0
5ed8 2011-02-20 11:06:37.6181922 Nodes/s;Highways/s;Tiles/s: 0/0/0
b831b6 2011-02-20 11:06:37.8181722 Nodes/s;Highways/s;Tiles/s: 0/0/0
lb817e 2011-02-20 11:06:37.4202120 Nodes/s;Highways/s;Tiles/s: 0/0/0
id628 2011-02-20 11:06:31.6181922 Nodes/s;Highways/s;Tiles/s: 0/0/0
38d5fa 2011-02-20 11:06:30.6181922 Nodes/s;Highways/s;Tiles/s: 0/0/0
74ae 2011-02-20 11:06:29.6209920 Nodes/s;Highways/s;Tiles/s: 0/0/0
i6d89a 2011-02-20 11:06:28.6190922 Nodes/s;Highways/s;Tiles/s: 0/0/0
8f1dc 2011-02-20 11:06:27.6191922 Nodes/s;Highways/s;Tiles/s: 0/0/0
35166 2011-02-20 11:06:26.6192922 Nodes/s;Highways/s;Tiles/s: 0/0/0
716b34 2011-02-20 11:06:25.6193922 Nodes/s;Highways/s;Tiles/s: 0/0/0
3000eb5 2011-02-20 11:06:24.5195022 Nodes/s;Highways/s;Tiles/s: 0/0/0
a60a8 2011-02-20 11:06:23.6195922 Nodes/s;Highways/s;Tiles/s: 0/0/0
afea0 2011-02-20 11:06:22.6206921 Nodes/s;Highways/s;Tiles/s: 0/0/0
87c156 2011-02-20 11:06:21.6197922 Nodes/s;Highways/s;Tiles/s: 0/0/0
a01788 2011-02-20 11:06:20.5209021 Nodes/s;Highways/s;Tiles/s: 0/0/0
36a3b 2011-02-20 11:06:19.5200022 Nodes/s;Highways/s;Tiles/s: 0/0/0
d2b0 2011-02-20 11:06:18.0211521 Nodes/s;Highways/s;Tiles/s: 0/0/0
882d6 2011-02-20 11:06:18.2201322 Nodes/s;Highways/s;Tiles/s: 0/0/0
i17652 2011-02-20 11:06:18.6920850 Nodes/s;Highways/s;Tiles/s: 0/0/0
i56517 2011-02-20 11:06:15.5204022 Nodes/s;Highways/s;Tiles/s: 0/0/0
3b7e 2011-02-20 11:06:14.5205022 Nodes/s;Highways/s;Tiles/s: 0/0/0
2991 2011-02-20 11:06:13.5206022 Nodes/s;Highways/s;Tiles/s: 0/0/0
lc9bf6 2011-02-20 11:06:12.5197023 Nodes/s;Highways/s;Tiles/s: 0/0/0
3424f 2011-02-20 11:06:11.6207922 Nodes/s;Highways/s;Tiles/s: 0/0/0
7ac02 2011-02-20 11:06:10.5209022 Launched dynamically loaded component async.

Total: 5,835 Min.

Download

Solution 4: XmlReader

```
var nodes = new Dictionary<string, string>();
using (var context = new GeoWriterContext())
{
    using (var reader = XmlReader.Create(@"https://loadtesting.blob.core.windows.net/osm/berlin.xml"))
    {
        var isInWay = false;
        var highwayType = string.Empty;
        string motorwayId = string.Empty, startingNodeId = string.Empty, endNodeId = string.Empty;
        var motorwayNodes = new List<string>();

        while (reader.Read())
        {
            if (reader.NodeType == XmlNodeType.Element)
            {
                switch (reader.Name)
                {
                    case "node":
                        nodes.Add(reader.GetAttribute("id"),
                            string.Format("{0} {1}", reader.GetAttribute("lat"), reader.GetAttribute("lon")));
                        lock (statisticsLockObject)
                        {
                            nodesPerSecond++;
                        }
                        break;

                    case "way":
                        motorwayId = reader.GetAttribute("id");
                        isInWay = true;
                        break;

                    case "nd":
                        var refNodeId = reader.GetAttribute("ref");
                        if (isInWay && nodes.ContainsKey(refNodeId))
                        {
                            endNodeId = refNodeId;
                            if (startingNodeId.Length == 0)
                            {
                                startingNodeId = refNodeId;
                            }

                            motorwayNodes.Add(nodes[refNodeId]);
                        }
                        break;

                    case "tag":
                        if (reader.GetAttribute("k") == "highway")
                        {
                            highwayType = reader.GetAttribute("v");
                        }
                        break;

                    default:
                        break;
                }
            }
            else if (reader.NodeType == XmlNodeType.EndElement)
            {
                if (isInWay)
                {
                    if (highwayType != null)
                    {
                        if (motorwayNodes.Count > 0)
                        {
                            context.AddWay(motorwayId, startingNodeId, endNodeId, highwayType);
                        }
                    }
                }
            }
        }
    }
}
```

Solution 4: XmlReader

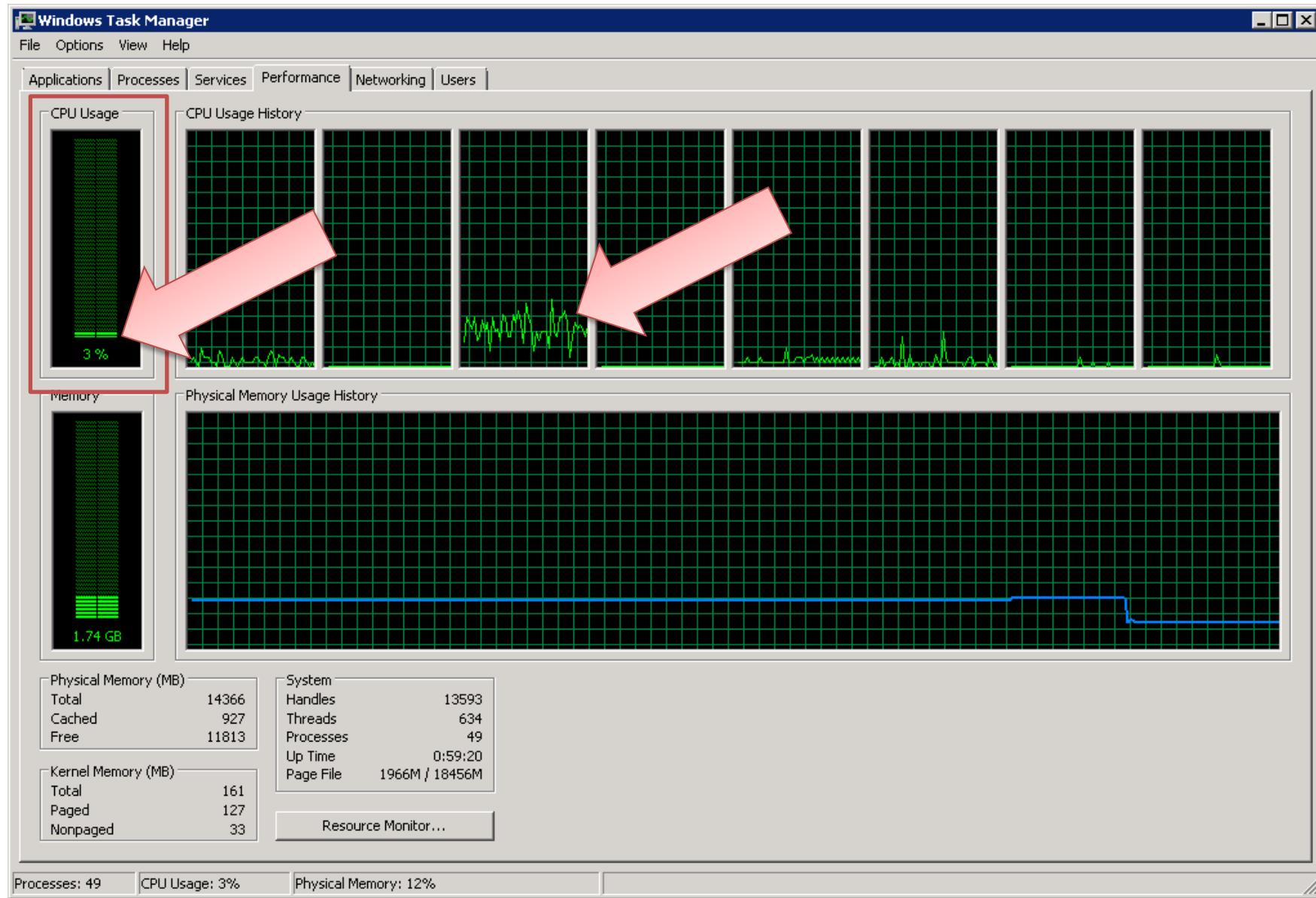
```
j
else if (reader.NodeType == XmlNodeType.EndElement)
{
    if (reader.Name == "way")
    {
        if (isInWay && !string.IsNullOrEmpty(highwayType) && motorwayNodes.Count > 1)
        {
            bool isFirstNode = true;
            var lineStringBuilder = new StringBuilder("LINESTRING(");
            foreach (var node in motorwayNodes)
            {
                if (!isFirstNode)
                {
                    lineStringBuilder.Append(',');
                }
                else
                {
                    isFirstNode = false;
                }

                lineStringBuilder.Append(node);
            }

            lineStringBuilder.Append(')');
        }

        Write row to database
    }

    motorwayNodes.Clear();
    isInWay = false;
    highwayType = string.Empty;
    startingNodeId = endNodeId = string.Empty;
}
}
```



4auc8a1	2011-02-20 11:27:04.975511	Nodes/s;Highways/s;Tiles/s: 0/189/0	2011-02-20 11:27:05.545065 /
1d91b818	2011-02-20 11:27:03.9756311	Nodes/s;Highways/s;Tiles/s: 0/188/0	2011-02-20 11:27:04.5430913
d14b357	2011-02-20 11:27:02.9757311	Nodes/s;Highways/s;Tiles/s: 0/188/0	2011-02-20 11:27:03.5431169
i93d1ce1	2011-02-20 11:27:01.9758311	Nodes/s;Highways/s;Tiles/s: 0/198/0	2011-02-20 11:27:02.5431488
4978f31	2011-02-20 11:27:00.9759311	Nodes/s;Highways/s;Tiles/s: 0/190/0	2011-02-20 11:27:01.5431808
le70106	2011-02-20 11:26:59.9760311	Nodes/s;Highways/s;Tiles/s: 0/188/0	2011-02-20 11:27:00.5432128
a2259a	2011-02-20 11:26:58.9761311	Nodes/s;Highways/s;Tiles/s: 0/186/0	2011-02-20 11:26:59.5432448
167078d	2011-02-20 11:26:57.9762311	Nodes/s;Highways/s;Tiles/s: 0/189/0	2011-02-20 11:26:58.5432768
35774a8	2011-02-20 11:26:56.9763311	Nodes/s;Highways/s;Tiles/s: 0/186/0	2011-02-20 11:26:57.5433088
98d6245	2011-02-20 11:26:55.9764311	Nodes/s;Highways/s;Tiles/s: 0/173/0	2011-02-20 11:26:56.5433408
382dce55	2011-02-20 11:26:54.9765311	Nodes/s;Highways/s;Tiles/s: 0/184/0	2011-02-20 11:26:55.5433728
:ae2a013	2011-02-20 11:26:53.9766311	Nodes/s;Highways/s;Tiles/s: 0/188/0	2011-02-20 11:26:54.5434048
af77d3c	2011-02-20 11:26:52.9767311	Nodes/s;Highways/s;Tiles/s: 0/180/0	2011-02-20 11:26:53.5434368
1568976	2011-02-20 11:26:51.9768311	Nodes/s;Highways/s;Tiles/s: 0/114/0	2011-02-20 11:26:52.5434688
28c34b7	2011-02-20 11:26:50.9779310	Nodes/s;Highways/s;Tiles/s: 135175/0/0	2011-02-20 11:26:51.5435008
fe776fa	2011-02-20 11:26:49.9770311	Nodes/s;Highways/s;Tiles/s: 138644/0/0	2011-02-20 11:26:5
1905ce4	2011-02-20 11:26:48.9781310	Nodes/s;Highways/s;Tiles/s: 137719/0/0	2011-02-20 11:26:4
99707c8	2011-02-20 11:26:47.9792309	Nodes/s;Highways/s;Tiles/s: 142014/0/0	2011-02-20 11:26:4
814ad5	2011-02-20 11:26:46.9783310	Nodes/s;Highways/s;Tiles/s: 132040/0/0	2011-02-20 11:26:4
77b81fd	2011-02-20 11:26:46.0774211	Nodes/s;Highways/s;Tiles/s: 128271/0/0	2011-02-20 11:26:4
3e6846f	2011-02-20 11:26:44.9775311	Nodes/s;Highways/s;Tiles/s: 134859/0/0	2011-02-20 11:26:45.5124309
?df98732	2011-02-20 11:26:44.1776111	Nodes/s;Highways/s;Tiles/s: 21578/0/0	2011-02-20 11:26:44.5593303
ff9a857	2011-02-20 11:26:42.6917597	Start dynamically loaded component async.	2011-02-20 11:26:43.1843655
2c3d9d	2011-02-20 11:26:42.9777311	Dynamically loaded component async.	2011-02-20 11:26:43.3874853

Total: 6,24 Min.
7,1% slower

Solution 5: Producer/Consumer

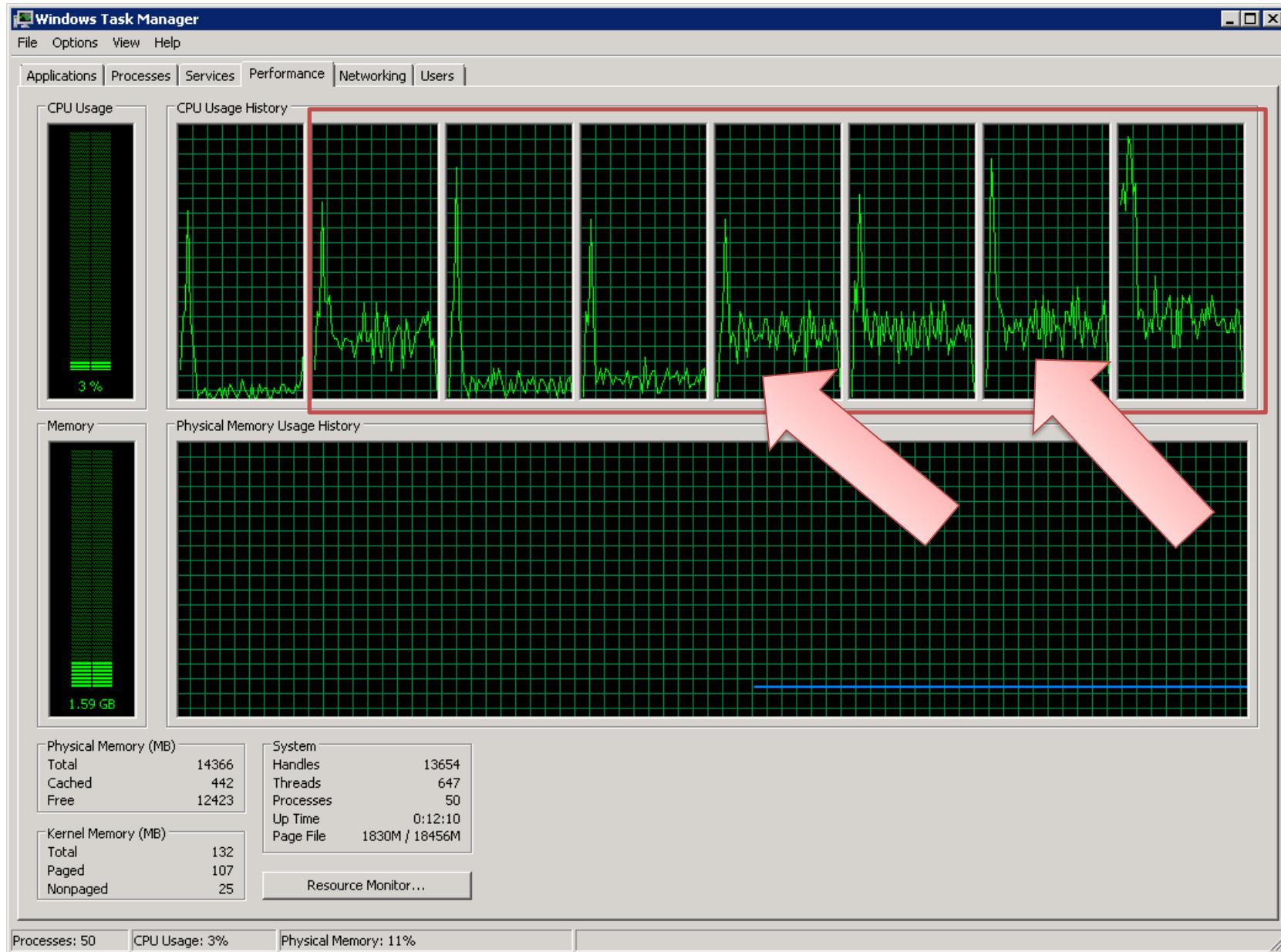
```

var queue = new BlockingCollection<dynamic>(100000);
int workerThreads, completionPortThreads;
ThreadPool.GetMaxThreads(out workerThreads, out completionPortThreads);
ThreadPool.SetMaxThreads(
    Math.Max(workerThreads, Environment.ProcessorCount * 3),
    Math.Max(workerThreads, Environment.ProcessorCount * 3));

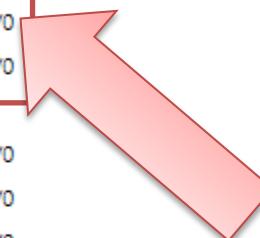
var consumer = Enumerable.Range(0, Environment.ProcessorCount * 3).Select(i =>
    Task.Factory.StartNew(() =>
{
    using (var context = new GeoWriterContext())
    {
        foreach (var item in queue.GetConsumingEnumerable())
        {
            #region Write row to database
            try
            {
                context.InsertHighway(
                    item.WayId,
                    SqlGeography.STLineFromText(new SqlChars(new SqlString(item.Linestring.ToString())), 4326),
                    item.WayType,
                    item.StartingNodeId,
                    item.EndNodeId);

                lock (this.statisticsLockObject)
                {
                    this.highwaysPerSecond++;
                }
            }
            catch (Exception ex)
            {
                logWriter(string.Format("Could not write way {0}: {1}", item
                    exceptionWriter(ex));
            }
            #endregion
        }
    }
})).ToArray();
}

queue.Add(new
{
    WayId = Int32.Parse(motorwayId),
    Linestring = lineStringBuilder.ToString(),
    WayType = highwayType,
    StartingNodeId = Int32.Parse(startingNodeId),
    EndNodeId = Int32.Parse(endNodeId)
});
```



0ca099d51	2011-02-20 12:08:12 //244/04 Nodes/s;Highways/s;Tiles/s: 0/1446/0
f5659eaa4	2011-02-20 12:08:36.7244704 Nodes/s;Highways/s;Tiles/s: 0/1481/0
4eb0c2b01e	2011-02-20 12:08:35.7244704 Nodes/s;Highways/s;Tiles/s: 0/1415/0
4e23b1c3cf	2011-02-20 12:08:34.7234704 Nodes/s;Highways/s;Tiles/s: 0/957/0
'b86b91df6	2011-02-20 12:08:34.0244704 Nodes/s;Highways/s;Tiles/s: 0/1438/0
:72b65ddd70	2011-02-20 12:08:33.0244704 Nodes/s;Highways/s;Tiles/s: 0/1461/0
!f0fd1df8a2	2011-02-20 12:08:32.0244704 Nodes/s;Highways/s;Tiles/s: 0/1441/0
'7da4bf3c8	2011-02-20 12:08:31.2244704 Nodes/s;Highways/s;Tiles/s: 0/0/0
5adf312cc4	2011-02-20 12:08:31.0244704 Nodes/s;Highways/s;Tiles/s: 0/2877/0
98327acbc9	2011-02-20 12:08:29.0234704 Nodes/s;Highways/s;Tiles/s: 0/1436/0
'3edbfb132e	2011-02-20 12:08:28.0244704 Nodes/s;Highways/s;Tiles/s: 0/1429/0
!9f771d7fc	2011-02-20 12:08:27.0244704 Nodes/s;Highways/s;Tiles/s: 0/6703/0
:f98e92857	2011-02-20 12:08:26.0244704 Nodes/s;Highways/s;Tiles/s: 0/11975/0
7e6673df77	2011-02-20 12:08:25.0244704 Nodes/s;Highways/s;Tiles/s: 0/13035/0
f00cb15703	2011-02-20 12:08:24.2244704 Nodes/s;Highways/s;Tiles/s: 0/0/0
>ecba32e846	2011-02-20 12:08:24.0244704 Nodes/s;Highways/s;Tiles/s: 0/29902/0
a91fe3115f	2011-02-20 12:08:22.1244704 Nodes/s;Highways/s;Tiles/s: 16966/7066/0
741b3ee94	2011-02-20 12:08:21.7254704 Nodes/s;Highways/s;Tiles/s: 60/0/0
:c3a1b090d	2011-02-20 12:08:21.5234704 Nodes/s;Highways/s;Tiles/s: 252723/0/0
181742793	2011-02-20 12:08:19.4244704 Nodes/s;Highways/s;Tiles/s: 3/0/0
Db1e6b454a	2011-02-20 12:08:19.2244704 Nodes/s;Highways/s;Tiles/s: 49/0/0
!df08735d9	2011-02-20 12:08:19.0244704 Nodes/s;Highways/s;Tiles/s: 227380/0/0
:2eca38e15a	2011-02-20 12:08:16.5244704 Nodes/s;Highways/s;Tiles/s: 82399/0/0
6545a3873e	2011-02-20 12:08:14.8554704 Nodes/s;Highways/s;Tiles/s: 0/0/0
d7b4d2bafa	2011-02-20 12:08:15.5274704 Nodes/s;Highways/s;Tiles/s: 75340/0/0
:7dc3a8ed64	2011-02-20 12:08:14.6534704 Nodes/s;Highways/s;Tiles/s: 133362/0/0
a75188ec7	2011-02-20 12:08:13.0244704 Nodes/s;Highways/s;Tiles/s: 182018/0/0
:6f72af883e	2011-02-20 12:08:10.6274704 Nodes/s;Highways/s;Tiles/s: 0/0/0



Total: 0,98 Min.
83% faster

Dynamic, DLR

C# DYNAMIC FEATURES

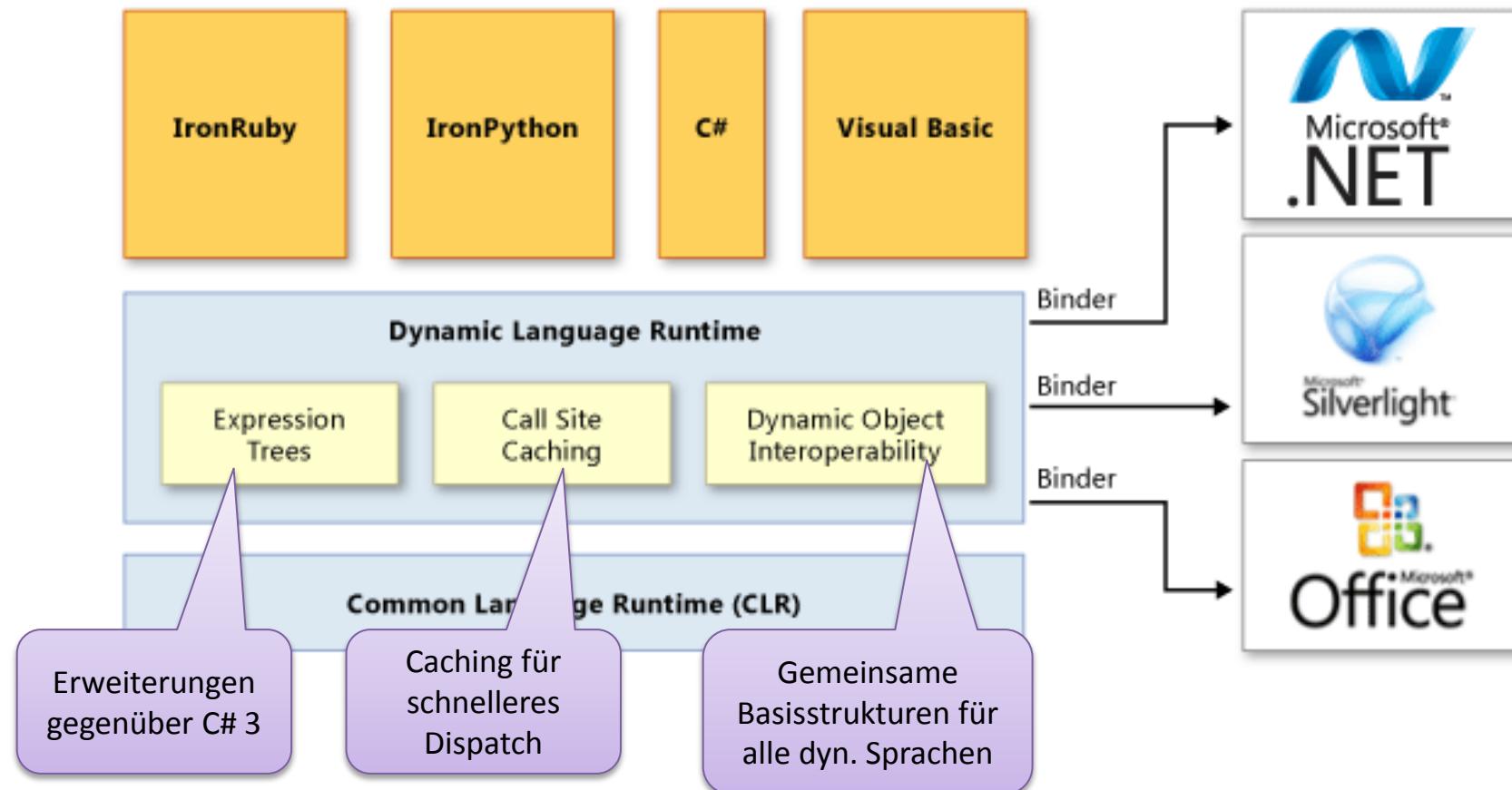
C# dynamic Keyword

- Kein compile-time type checking
- Zur Laufzeit wird dynamic zu object + Code zur Auflösung des Member Access
- Kann verwendet werden...
 - ...bei Deklarationen von Members, return values, Parameter, lokale Variablen und type constraints
 - ...bei Typkonvertierungen
 - ...überall wo Typen als Werte verwendet werden (z.B. is, as, typeof)

C# dynamic Keyword

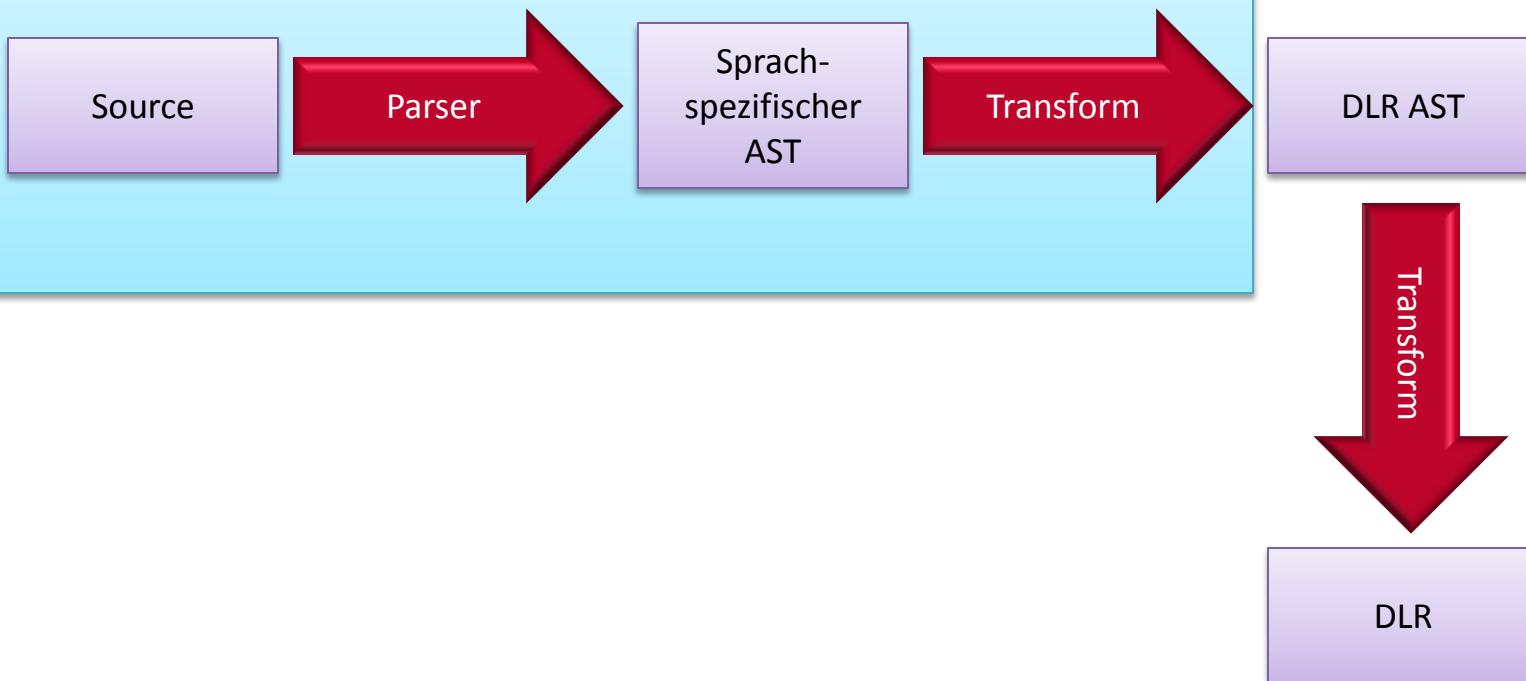
- Anwendungsbereiche
 - COM API
 - Dynamische Sprachen (z.B. IronPython)
 - HTML DOM
- Nachteile
 - Fehler treten zur Laufzeit auf
 - Etwas längere Laufzeit

Dynamic Language Runtime



AST in DLR

Sprachspezifisch



• ExpressionTrees in C#

■ Inheritance Hierarchy

2010

```
System.Object
System.Linq.Expressions.Expression
  System.Linq.Expressions.BinaryExpression
  System.Linq.Expressions.BlockExpression
  System.Linq.Expressions.ConditionalExpression
  System.Linq.Expressions.ConstantExpression
  System.Linq.Expressions.DebugInfoExpression
  System.Linq.Expressions.DefaultExpression
  System.Linq.Expressions.DynamicExpression
  System.Linq.Expressions.GotoExpression
  System.Linq.Expressions.IndexExpression
  System.Linq.Expressions.InvocationExpression
  System.Linq.Expressions.LabelExpression
  System.Linq.Expressions.LambdaExpression
  System.Linq.Expressions.ListInitExpression
  System.Linq.Expressions.LoopExpression
  System.Linq.Expressions.MemberExpression
  System.Linq.Expressions.MemberInitExpression
  System.Linq.Expressions.MethodCallExpression
  System.Linq.Expressions.NewArrayExpression
  System.Linq.Expressions.NewExpression
  System.Linq.Expressions.ParameterExpression
  System.Linq.Expressions.RuntimeVariablesExpression
  System.Linq.Expressions.SwitchExpression
  System.Linq.Expressions.TryExpression
  System.Linq.Expressions.TypeBinaryExpression
  System.Linq.Expressions.UnaryExpression
```

■ Inheritance Hierarchy

2008

```
System.Object
System.Linq.Expressions.Expression
  System.Linq.Expressions.BinaryExpression
  System.Linq.Expressions.ConditionalExpression
  System.Linq.Expressions.ConstantExpression
  System.Linq.Expressions.InvocationExpression
  System.Linq.Expressions.LambdaExpression
  System.Linq.Expressions.ListInitExpression
  System.Linq.Expressions.MemberExpression
  System.Linq.Expressions.MemberInitExpression
  System.Linq.Expressions.MethodCallExpression
  System.Linq.Expressions.NewArrayExpression
  System.Linq.Expressions.NewExpression
  System.Linq.Expressions.ParameterExpression
  System.Linq.Expressions.TypeBinaryExpression
  System.Linq.Expressions.UnaryExpression
```

DLR Basics

- `IDynamicMetaObjectProvider`
 - `GetMetaObject` → `DynamicMetaObject`
- `DynamicMetaObject`
 - Erzeugt Expression Trees für das dynamische Binden
- `DynamicObject`
 - Vereinfacht die Implementierung von `IDynamicMetaObjectProvider`
- `ExpandoObject`
 - Erlaubt dynamisches Hinzufügen und Entfernen von Members
 - Implementiert `IDictionary` und `IPropertyChanged`

Voraussetzungen für IronPython

- Herunterladen von [IronPython für .NET](#)
- Referenzen auf
 - IronPython.dll
 - IronPython.Modules.dll
 - Microsoft.Scripting.dll
 - Microsoft.Dynamic.dll

Typische Einsatzgebiete

aus Sicht von C# Entwicklern ;-)

- Typische Anwendungen
 - Automatisieren von Routinetätigkeiten (Makros)
 - Schnittstellen
 - Installation, Wartung, Updates
 - Prototyping
- Nutzen
 - Anpassungsmöglichkeiten vorort beim Kunden eventuell durch den Kunden
 - Kein VS, kein Kompilieren notwendig
 - Dynamisches Programmieren manchmal effektiver (z.B. bei Prototyping)
 - Python ist eine coole Sprache

Hosting API Grundlagen

- Microsoft.Scripting.Hosting
- ScriptRuntime
 - Möglichkeit, verschiedene Laufzeitumgebungen voneinander zu trennen (z.B. für Security)
 - Python runtime mit
`IronPython.Hosting.Python.CreateRuntime()`
- ScriptEngine
 - Enthält alle wichtigen Funktionen zum Ausführen von Python Code, zum Zugriff auf Variablen, etc.
 - Python Engine mit
`IronPython.Hosting.Python.CreateEngine()`



Under Creative Common License
<http://www.flickr.com/photos/42311564@N00/2355590508/>

Use Case 1: Scripting

Möglichkeit, im Programm Scripts auszuführen

Pythondatei ausführen

```
var engine = Python.CreateEngine();
using (var stream = new ScriptOutputStream( s => {
    this.AppendToScriptOutput(s);
    App.Current.Dispatcher.BeginInvoke(
        new Action(() => this.OnPropertyChanged("ScriptOutput")));
}, Encoding.UTF8))
{
    engine.Runtime.IO.SetOutput(stream, Encoding.UTF8);
    var scriptSource = engine.CreateScriptSourceFromFile("Samplescript01.py");
    try
    {
        scriptSource.Execute();
    }
    catch (SyntaxErrorException e)
    {
        this.AppendToScriptOutput("Syntax error (line {0}, column {1}): {2}",
            e.Line, e.Column, e.Message);
        App.Current.Dispatcher.BeginInvoke(
            new Action(() => this.OnPropertyChanged("ScriptOutput")));
    }
}
```

Wegen asynchroner Ausführung

Exkurs: ScriptOutputStream

```
public sealed class ScriptOutputStream : Stream
{
    public ScriptOutputStream(Action<string> write, Encoding encoding)
    {
        [...]
        chunks = new BlockingCollection<byte[]>();
        this.processingTask = Task.Factory.StartNew(() => {
            foreach (var chunk in chunks.GetConsumingEnumerable())
            {
                write(this.encoding.GetString(chunk));
            }
        }, TaskCreationOptions.LongRunning);
    }
    public override void Write(byte[] buffer, int offset, int count)
    {
        var chunk = new byte[count];
        Buffer.BlockCopy(buffer, offset, chunk, 0, count);
        this.chunks.Add(chunk);
    }
    public override void Close()
    {
        this.chunks.CompleteAdding();
        try { this.processingTask.Wait(); }
        finally { base.Close(); }
    }
    [...]
}
```

Consumer

Producer

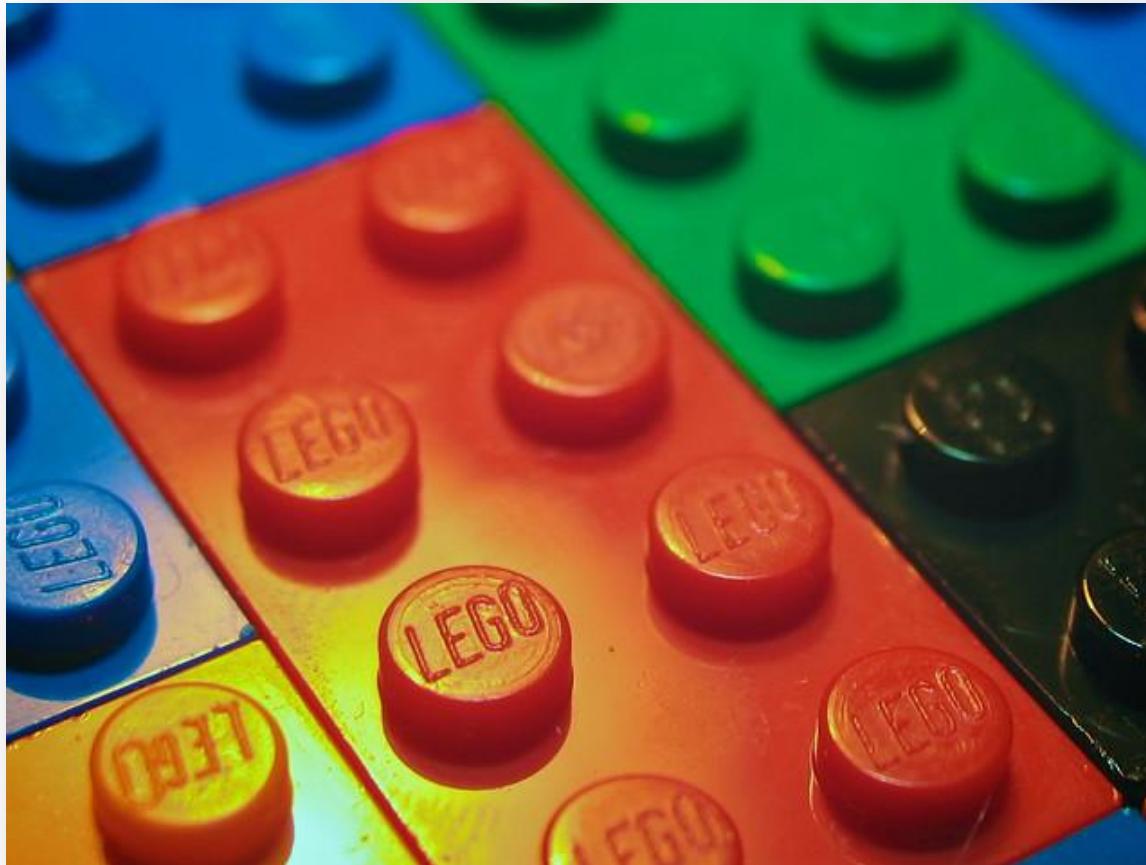
• Beispielscript in Python

```
import clr  
clr.AddReference("mscorlib")  
  
from System.Threading import Thread  
  
for i in range(0, 10):  
    print str(i+1)  
    Thread.Sleep(500)  
  
print "Done!"
```

Referenzen auf Assemblies

~using

Methode aus dem .NET Framework



Under Creative Common License
<http://www.flickr.com/photos/pixel8ed/3842982196/>

Use Case 2: Dynamisches UI

C# UI durch dynamische Elemente erweitern

• Beispielscript in Python

[...]

```
class ViewModel:  
    numberofspeakers = 0  
    def __init__(self, speakers):  
        self.numberofspeakers = speakers  
  
    def getNumberOfSpeakers():  
        vm =  
        ViewModel(Application.Current.MainWindow.DataContext.Speakers.Length)  
        stream =  
        Application.Current.GetType().Assembly.GetManifestResourceStream(  
            "IronPython.UI.Scripts.Resultwindow.xaml")  
        reader = StreamReader(stream)  
        window = XamlReader.Parse(reader.ReadToEnd())  
        reader.Close()  
        stream.Close()  
        window.DataContext = vm  
        window.FindName("closeButton").Click += Lambda s, e: window.Close()  
        window.Show()  
  
Application.Current.Dispatcher.BeginInvoke(Action(lambda:  
    getNumberOfSpeakers()))  
print "Done!"
```

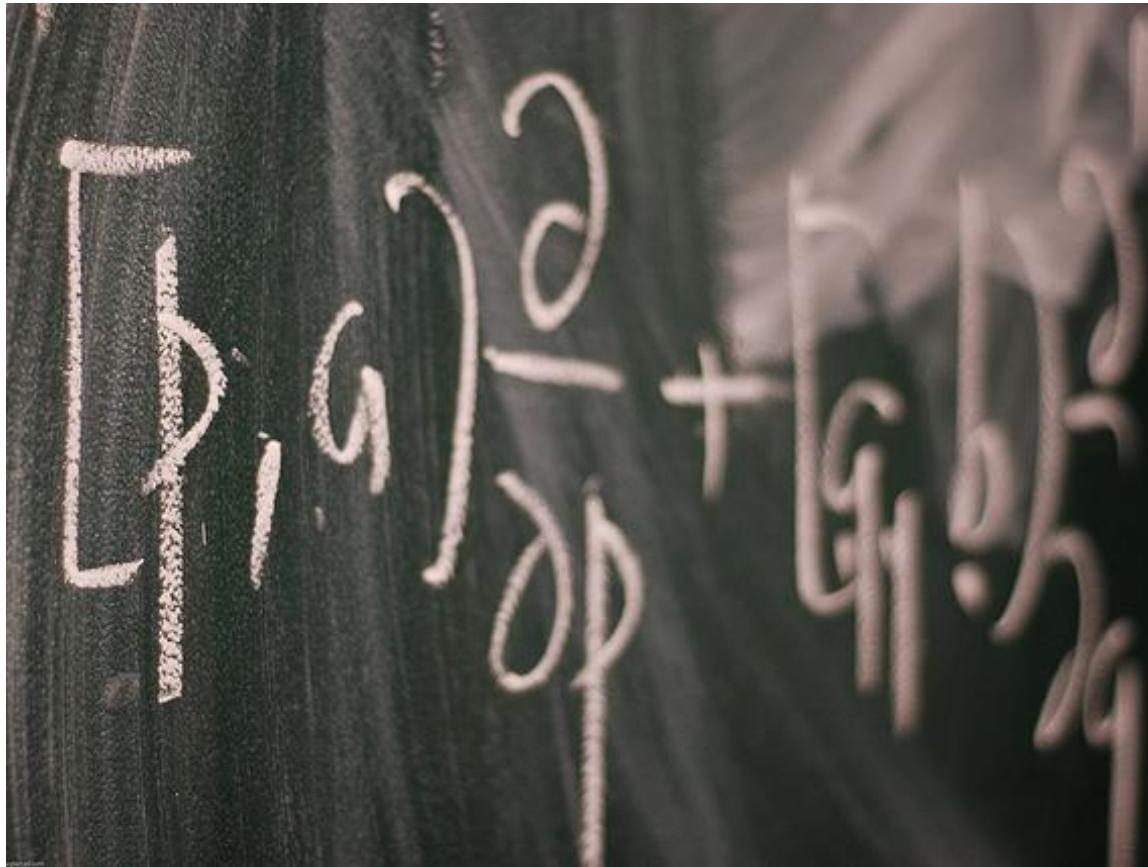
ViewModel geschrieben in Python (Python implementiert [ICustomTypeDescriptor](#))

Zugriff auf Elemente der C# Anwendung

Dynamisches Laden des XAML-Codes

Auf WPF-Event in Python reagieren

BeginInvoke wegen Hintergrundthread



Under Creative Common License
<http://www.flickr.com/photos/erikwst/2421129047/>

Use Case 3: Berechnete Spalten

Geschäftsobjekte mit Hilfe von Python um berechnete Spalten erweitern

Hilfsklasse zum Erweitern

```
public class ExtendedObject<T> : DynamicObject
{
    private Dictionary<string, Func<T, object>> calculatedProperties =
        new Dictionary<string, Func<T, object>>();

    public ExtendedObject(T underlyingObject)
    {
        this.UnderlyingObject = underlyingObject;
    }

    public T UnderlyingObject { get; private set; }

    public void AddCalculatedProperty(string propertyName, string formula)
    {
        // Proper error handling is missing!!!
        var engine = Python.CreateEngine();
        var script = engine.CreateScriptSourceFromString(formula);
        var function = script.Execute<Func<T, object>>();
        this.calculatedProperties.Add(propertyName, function);
    }

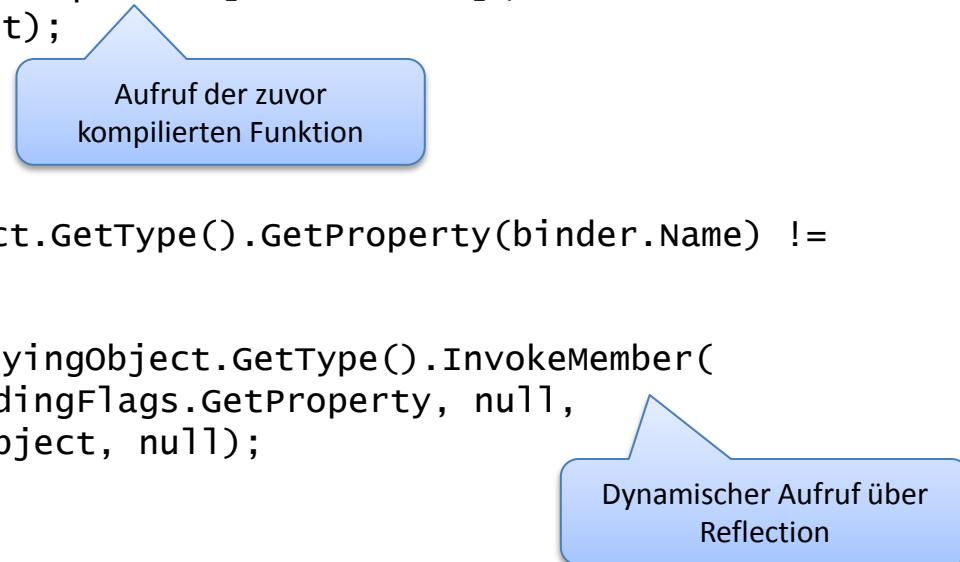
    [...]
```

DLR

Formel = Python Lambda
Expression

Hilfsklasse zum Erweitern

```
public override bool TryGetMember(GetMemberBinder binder, out object result)
{
    if (this.calculatedProperties.ContainsKey(binder.Name))
    {
        result = this.calculatedProperties[binder.Name](
            this.UnderlyingObject);
        return true;
    }
    else
    {
        if (this.UnderlyingObject.GetType().GetProperty(binder.Name) != null)
        {
            result = this.UnderlyingObject.GetType().InvokeMember(
                binder.Name, BindingFlags.GetProperty, null,
                this.UnderlyingObject, null);
            return true;
        }
    }
    return base.TryGetMember(binder, out result);
}
```



Praktische Anwendung

```
[...]  
this.Speakers = context.Speakers.Include("Sessions").ToArray()  
    .AsParallel()  
    .Select(speaker => new ExtendedObject<Speaker>(speaker)).ToArray();  
this.Speakers.AsParallel()  
    .ForAll(eo => eo.AddCalculatedProperty("FullName",  
        "lambda s: s.LastName + \", \" + s.FirstName"));  
[...]
```

Berechnete Spalte als
Python Lambda

```
<DataGrid [...]>  
    <DataGrid.Columns>  
        <DataGridTextColumn Binding="{Binding Path=FirstName}" [...] />  
        [...]  
        <DataGridTextColumn Binding="{Binding Path=FullName}" [...] />  
    </DataGrid.Columns>  
</DataGrid>
```

Ganz normales Binding in
UI (WPF)

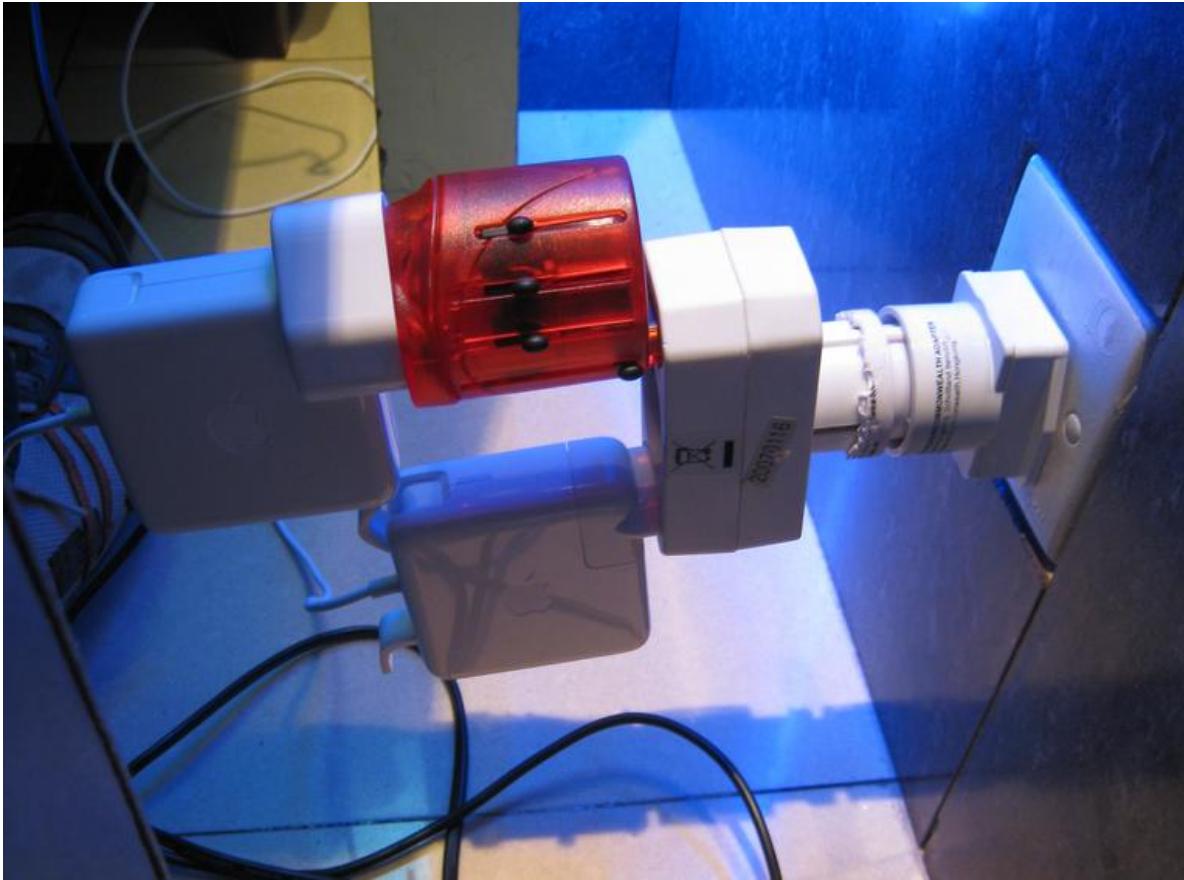
Advanced: LINQ in Python

```
this.Speakers.AsParallel().ForAll(  
    eo => eo.AddCalculatedProperty("NumberOfApprovedSessions", @"  
import clr  
clr.AddReference(""System.Core"")  
from System.Linq import Enumerable  
lambda s: Enumerable.Count(s.Sessions, lambda p: p.Approved));
```

Linq in Python Lambda

```
this.Speakers.AsParallel().ForAll(  
    eo => eo.AddCalculatedProperty("NumberOfApprovedSessions",  
    "lambda s: len([session for session in s.Sessions if  
        session.Approved])"));
```

Das gleiche mit Python list
comprehension



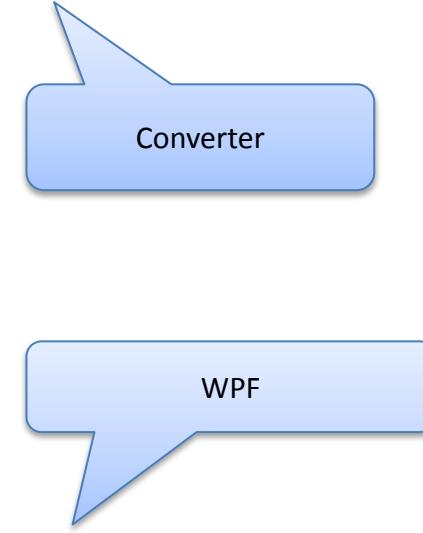
Under Creative Commons License
<http://www.flickr.com/photos/mnoach/3922903520/>

Use Case 4: Simplify ViewModel

The last converter ever written ;-)

IronPython Converter

```
public object Convert(object value, Type targetType, object parameter,  
CultureInfo culture)  
{  
    var engine = Python.CreateEngine();  
    var scope = engine.CreateScope();  
    scope.SetVariable("value", value);  
    engine.CreateScriptSourceFromString(parameter.ToString(),  
        SourceCodeKind.Expression);  
    var result = engine.Execute(parameter.ToString(), scope);  
    return result;  
}  
  
<DataGridTemplateColumn Header="Number of approved Sessions">  
    <DataGridTemplateColumn.CellTemplate>  
        <DataTemplate>  
            <Border Background="{Binding Path=NumberOfApprovedSessions,  
                Converter={StaticResource ResourceKey=IronPythonExpressionConverter},  
                ConverterParameter='"if value == 0 else "Green"'}">  
                <TextBlock Text="{Binding Path=NumberOfApprovedSessions}" />  
            </Border>  
        </DataTemplate>  
    </DataGridTemplateColumn.CellTemplate>  
</DataGridTemplateColumn>
```



The diagram illustrates the flow of data from C# code to XAML binding. A blue arrow points from the C# code block on the left to the XAML code block on the right. Two callout bubbles are present: one labeled "Converter" pointing to the "Converter" section in the XAML code, and another labeled "WPF" pointing to the "WPF" section in the XAML code.



Under Creative Common License
<http://www.flickr.com/photos/4442915@N00/4430052521/>

Use Case 5: Logik in Python

Teile der Verarbeitungsfunktionen in Python

• Python Beispielcode

```
this.ApproveSessionCommand = new GenericCommand(  
    x => this.SelectedSession != null,  
    x =>  
    {  
        var engine = Python.CreateEngine();  
        var scope = engine.CreateScope();  
        scope.SetVariable("viewModel", this);  
        engine.CreateScriptSourceFromString(@"  
viewModel.SelectedSession.Approved = True  
viewModel.SaveChanges()  
").Execute(scope);  
    }, this);
```

Variablen an Script
übergeben

Excel-Export

```
import clr
clr.AddReferenceByName(
    'Microsoft.Office.Interop.Excel, Version=11.0.0.0, Culture=neutral,
    PublicKeyToken=71e9bce111e9429c')
clr.AddReference("'"System.Core""")
from Microsoft.Office.Interop import *
from Microsoft.Office.Interop.Excel import *
from System import *

def export(speakers):
    ex = Excel.ApplicationClass()
    ex.Visible = True
    ex.DisplayAlerts = False
    workbook = ex.workbooks.Add()
    ws = workbook.worksheets[1]
    rowIndex = 1
    for speaker in speakers:
        ws.Cells[rowIndex, 1].Value2 = speaker.FirstName
        ws.Cells[rowIndex, 2].Value2 = speaker.LastName
        ws.Cells[rowIndex, 3].Value2 = speaker.FullName
        ws.Cells[rowIndex, 4].Value2 = speaker.NumberOfApprovedSessions
        rowIndex = rowIndex + 1
```

Import des Excel Interop Assemblies

Export einer Funktion

Excel-Automatisierung

• Excel-Export

```
var engine = Python.CreateEngine();
var scope = engine.CreateScope();
var scriptSource = @"[...]";
engine.CreateScriptSourceFromString(scriptSource).Execute(scope);
dynamic exportFunc = scope.GetVariable("export");
engine.Operations.Call(exportFunc, this.Speakers);
```

Aufruf der Funktion mit
ObjectOperations

Funktionsdefinition
abfragen

Weitere Ressourcen

- IronPython Dokumentation
 - <http://www.ironpython.net>
 - <http://docs.python.org>
 - Sourcecode (DLR und IronPython sind auf [codeplex](http://codeplex.com))
- Lust, IronPython in einer echten Anwendung auszuprobieren?
 - <http://www.timecockpit.com>

Why does the world need MEF?

THE PROBLEM

Original Goals

- Before MEF
 - Multiple extensibility mechanism for different Microsoft tools (e.g. Visual Studio, Trace Listeners, etc.)
 - Developers outside of MS had the same problem
- MEF: Provide standard mechanisms for hooks for 3rd party extensions
- Goal: *Open and Dynamic Applications*
 - make it easier and cheaper to build extensible applications and extensions

MEF „Hello World“

```
[Export(typeof(Shape))]  
public class Square : Shape  
{  
    // Implementation  
}
```

Export with
name or type

```
[Export(typeof(Shape))]  
public class Circle : Shape  
{  
    // Implementation  
}
```

„Attributed
Programming
Model“

```
[Export]  
public class Toolbox  
{  
    [ImportMany]  
    public Shape[] Shapes { get; set; }  
    // Additional implementation...  
}  
[...]
```

Defaults to
typeof(Toolbox)

```
var catalog = new AssemblyCatalog(typeof(Square).Assembly);  
var container = new CompositionContainer(catalog);  
Toolbox toolbox = container.GetExportedValue<Toolbox>();
```

MEF „Hello World“ (continued)

- *Parts*
 - Square, Circle and Toolbox
- *Dependencies*
 - Imports (Import-Attribute)
 - E.g. Toolbox.Shapes
- *Capabilities*
 - Exports (Export-Attribute)
 - E.g. Square, Circle

MEF „Hello World“

DEMO

Exports And Imports

- Export attribute
 - Class
 - Field
 - Property
 - Method
- Import attribute
 - Field
 - Property
 - Constructor parameter
- Export and import must have the same contract
 - Contract name and contract type
 - Contract name and type can be inferred from the decorated element

Inherited Exports

```
• [Export]  
public class NumOne  
{  
    [Import]  
    public IMyData MyData  
        { get; set; }  
}
```

Import automatically inherited

```
public class NumTwo : NumOne  
{  
}
```

Export NOT inherited
→ NumTwo has no exports

```
[InheritedExport]  
public class NumThree  
{  
    [Export]  
    Public IMyData MyData { get; set; }  
}
```

Member-level exports
are never inherited

```
public class NumFour : NumThree  
{  
}
```

Inherits export with
contract NumThree
(including all metadata)

MEF Catalogs

- Catalogs provide components
- Derived from
`System.ComponentModel.Composition.
Primitives.ComposablePartCatalog`
 - AssemblyCatalog
 - Parse all the parts present in a specified assembly
 - DirectoryCatalog
 - Parses the contents of a directory
 - TypeCatalog
 - Accepts type array or a list of managed types
 - AggregateCatalog
 - Collection of ComposablePartCatalog objects

Directory catalog sample

DEMO

How to import using MEF

IMPORT TYPES

Lazy Imports

- Imported object is not instantiated immediately
 - Imported (only) when accessed
- Sample:

```
public class MyClass
{
    [Import]
    public Lazy<IMyAddin> MyAddin
        { get; set; }
}
```

Prerequisite Imports

- Composition engine uses parameter-less constructor by default
- Use a different constructor with `ImportingConstructor` attribute
- Sample:

```
[ImportingConstructor]
public MyClass(
    [Import(typeof(IMySubAddin))] IMyAddin
    MyAddin)
{
    _theAddin = MyAddin;
}
```

Could be removed here; automatically imported

Optional Imports

- By default composition fails if an import could not be fulfilled
- Use `AllowDefault` property to specify optional imports
- Sample:

```
public class MyClass
{
    [Import(AllowDefault = true)]
    public Plugin thePlugin { get; set; }
}
```

Creation Policy

- RequiredCreationPolicy **property**
- CreationPolicy.Any
 - Shared **if importer does not explicitly request NonShared**
- CreationPolicy.Shared
 - Single shared instance of the part will be created for all requestors
- CreationPolicy.NonShared
 - New non-shared instance of the part will be created for every requestor

Part Lifecycle

DEMO

Advanced exports

METADATA AND METADATA VIEWS

Goal

- Export provides additional metadata so that importing part can decide which one to use
- Import can inspect metadata without creating exporting part
- Prerequisite: Lazy import

Metadata and metadata views (10 Minutes)

DEMO

Metadata

```
namespace MetadataSample
{
    public interface ITranslatorMetadata
    {
        string SourceLanguage { get; }

        [DefaultValue("en-US")]
        string TargetLanguage { get; }
    }
}
```

Export Metadata can
be mapped to
metadata view
interface

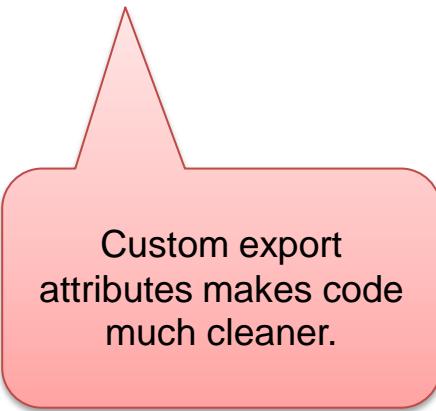
```
namespace MetadataSample
{
    [Export(typeof(ITranslator))]
    [ExportMetadata("SourceLanguage", "de-DE")]
    [ExportMetadata("TargetLanguage", "en-US")]
    public class GermanEnglishTranslator : ITranslator
    {
        public string Translate(string source)
        {
            throw new NotImplementedException();
        }
    }
}
```

Metadata (continued)

Custom Export Attributes

```
[TranslatorExport("de-DE", "en-US")]

public class GermanEnglishTranslator
    : ITranslator
{
    public string Translate(
        string source)
    {
        throw new NotImplementedException();
    }
}
```



Custom export
attributes makes code
much cleaner.

```
[Export(typeof(ITranslator))]
[ExportMetadata("SourceLanguage", "de-DE")]
[ExportMetadata("TargetLanguage", "en-US")]
public class GermanEnglishTranslator
    : ITranslator
{
    public string Translate(
        string source)
    {
        throw new NotImplementedException();
    }
}
```

Custom Export Attributes (continued)

```
[MetadataAttribute]
[AttributeUsage(AttributeTargets.Class, AllowMultiple = false)]
public class TranslatorExportAttribute
    : ExportAttribute, ITranslatorMetadata
{
    public TranslatorExportAttribute(
        string sourceLanguage, string targetLanguage)
        : base(typeof(ITranslator))
    {
        this.SourceLanguage = sourceLanguage;
        this.TargetLanguage = targetLanguage;
    }
    public string SourceLanguage { get; private set; }
    public string TargetLanguage { get; private set; }
}
```

Using MEF To Extend A WPF Application

DEMO

MEF AND SILVERLIGHT

MEF In Silverlight

- Additional catalog DeploymentCatalog
 - Load exported parts contained in XAP files
 - Provides methods for asynchronously downloading XAP files containing exported parts (`DeploymentCatalog.DownloadAsync`)
- Goal
 - Minimize initial load times
 - Application can be extended at run-time

MEF and Silverlight

DEMO

Read more about help, find the right tools

RESOURCES

Resources About MEF

- Managed Extensibility Framework on [MSDN](#)
- Managed Extensibility Framework for .NET 3.5 on [Codeplex](#)
- [Visual Studio 2010 and .NET Framework 4 Training Kit](#)



BASTA!

VIELEN DANK FÜR IHRE
MITARBEIT!