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

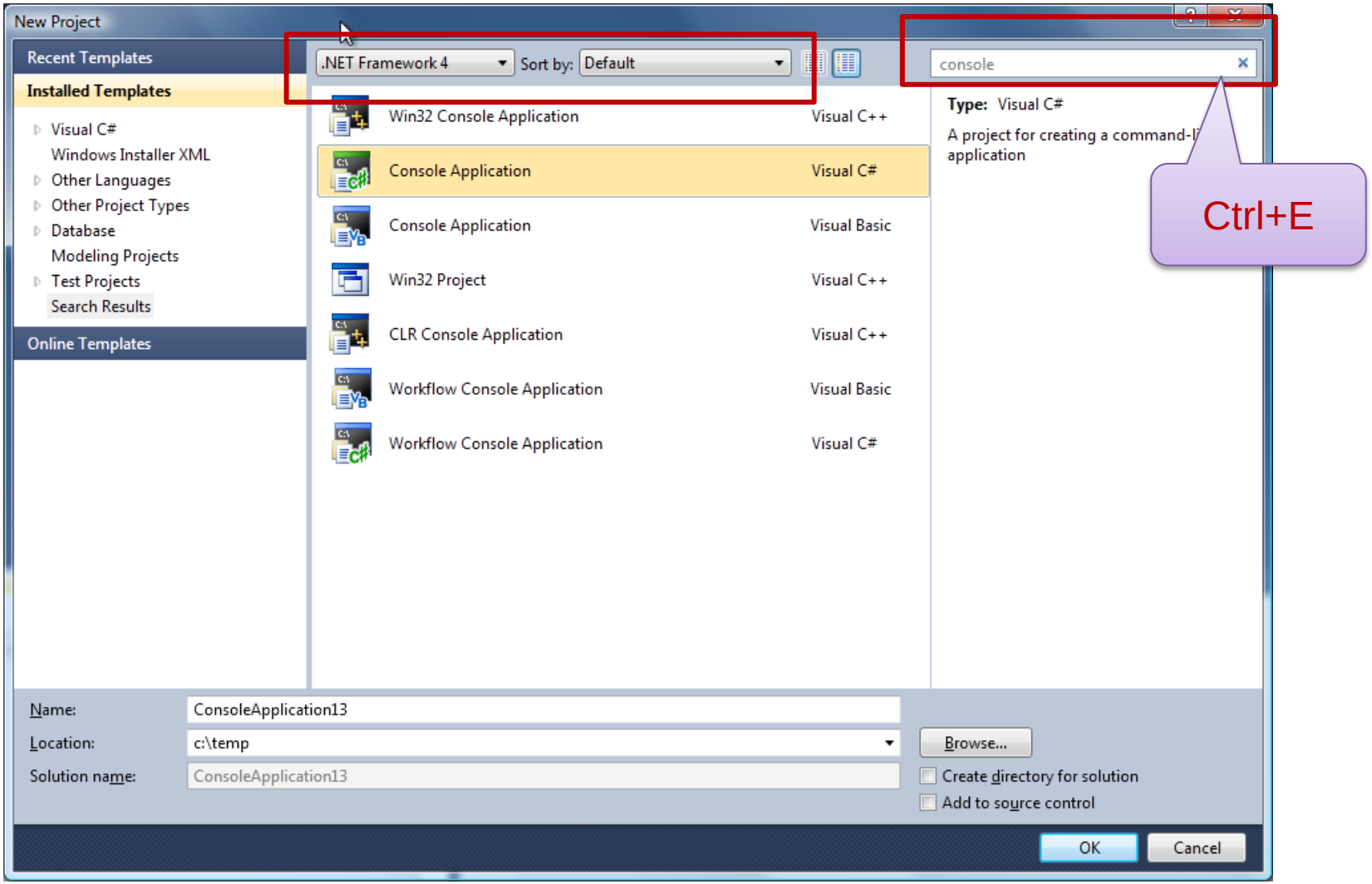
- Was ist neu in **Visual Studio 2010** für C# Entwickler?
- **Office Interop** – COM, No PIA, Optional Parameters, etc.
- **Parallel Computing** mit Tasks, PLINQ, etc.
- **dynamic** Keyword und Dynamic Language Runtime (**DLR**)
- Managed Extensibility Framework (**MEF**)

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

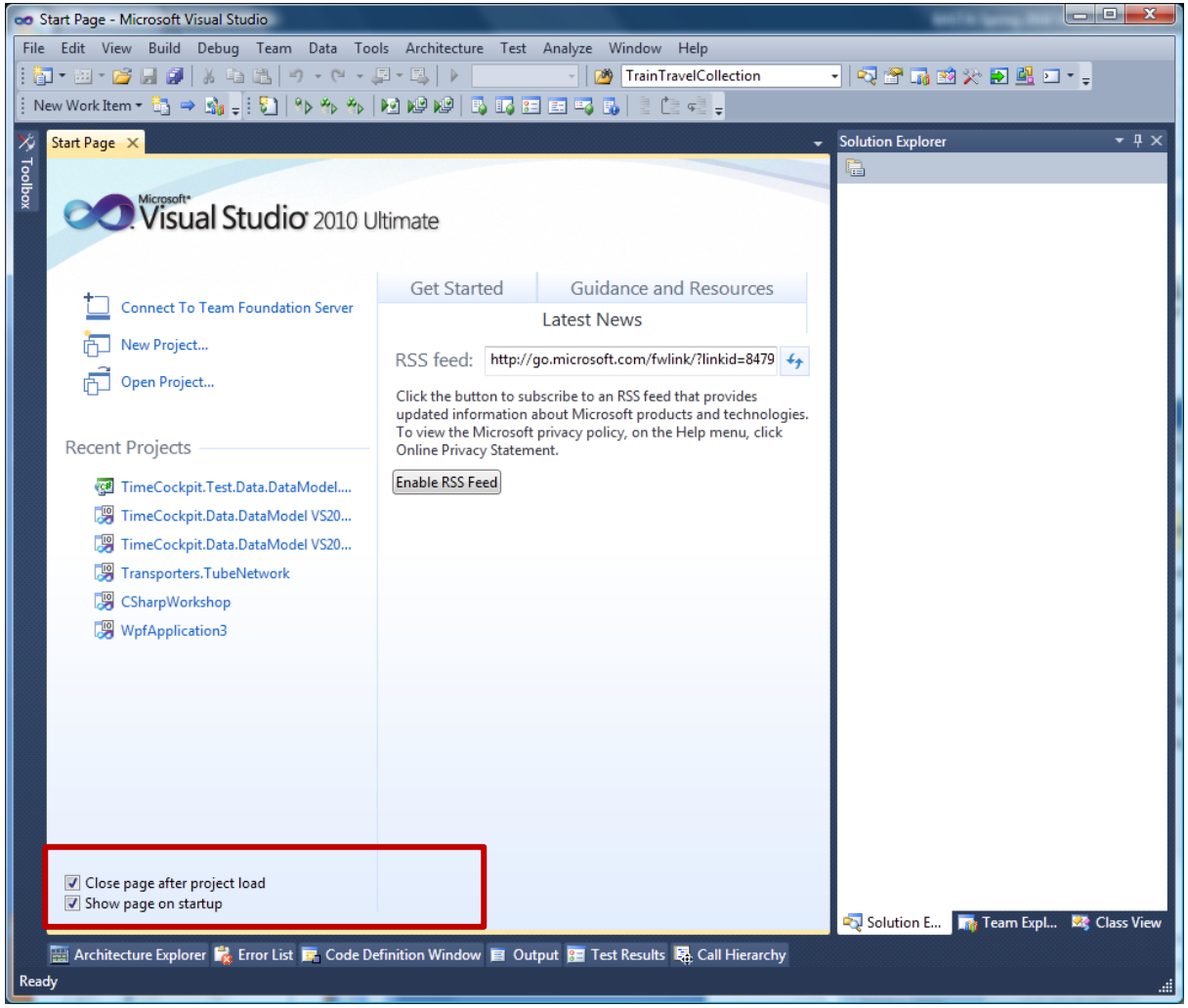
VISUAL STUDIO 2010
C# IDE

SOLUTIONS UND PROJEKTE

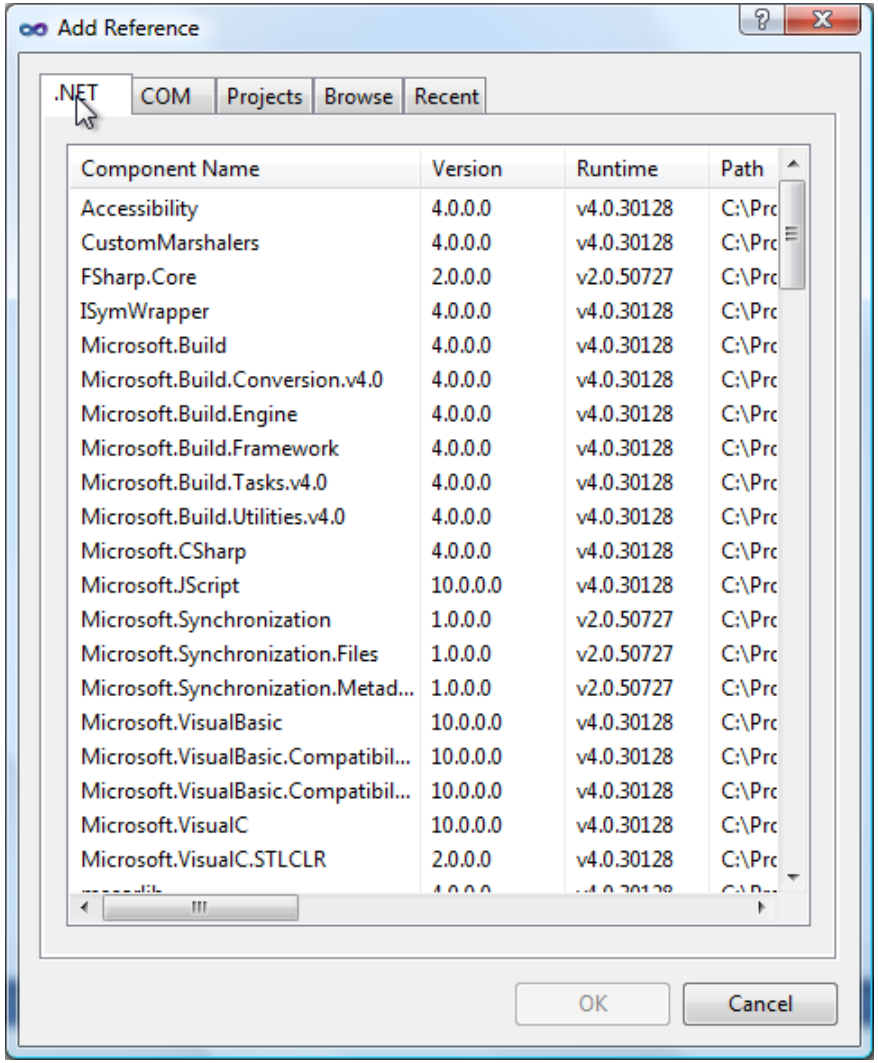
Verbesserter Project/New Dialog



Verbesserte Startpage



Async Add Reference ☺



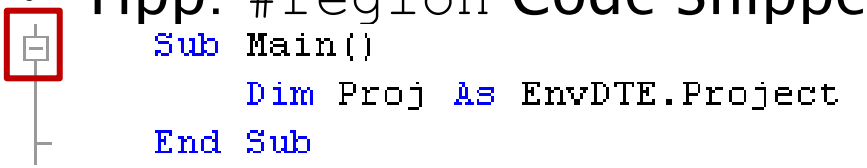
VISUAL STUDIO EDITOR

Code Selection, Copy/Move

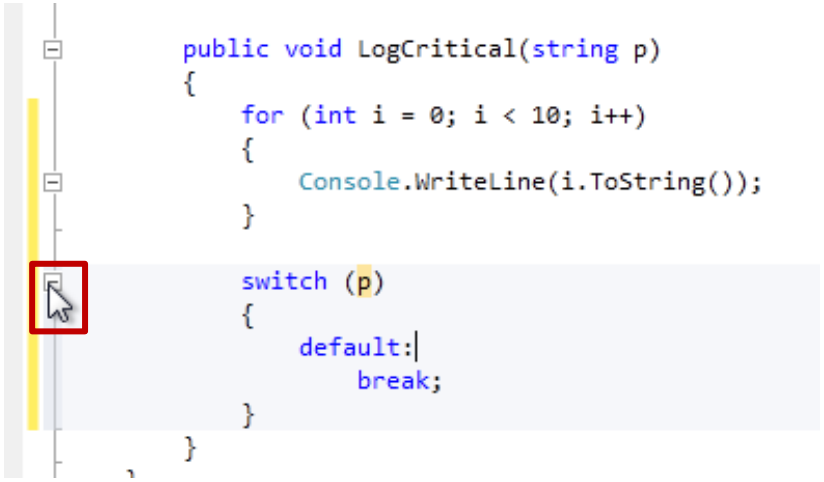
- Tipp: Column Mode
  - Alt+Maus oder Shift+Alt+Cursor
- Cut, Copy, Paste
  - Ctrl+X, Ctrl+C, Ctrl+V
  - Tipp: Clipboard ring (Ctrl+Shift+V)
    - Zugriff auf die letzten 20 kopierten Texte
  - Tipp: Ohne Markierung ganze Zeile ausschneiden/kopieren

Outlining

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



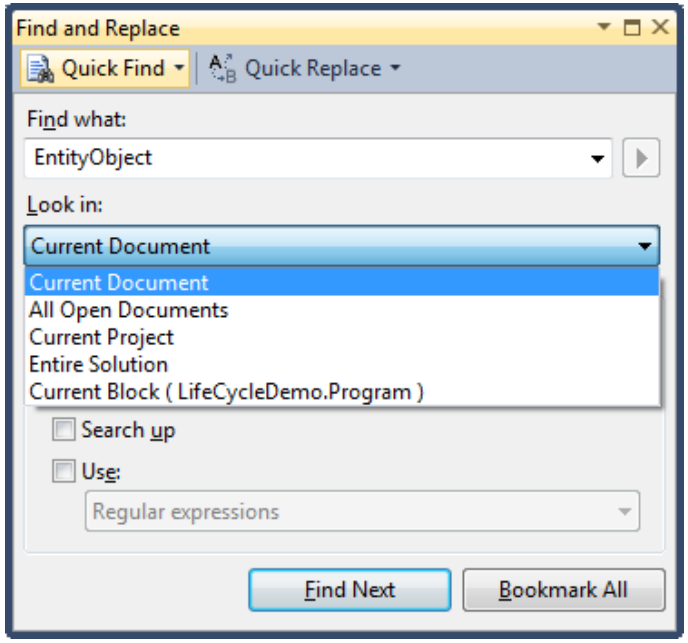
- **Neu:** Ad Hoc Blocks
  - Markieren des gewünschten Codeblocks
  - *Hide Selection* (Ctrl+M, H)
  - → Ad Hoc Block erzeugt



Sonstige Editor-Tipps

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

Suchen und Ersetzen (1/3)

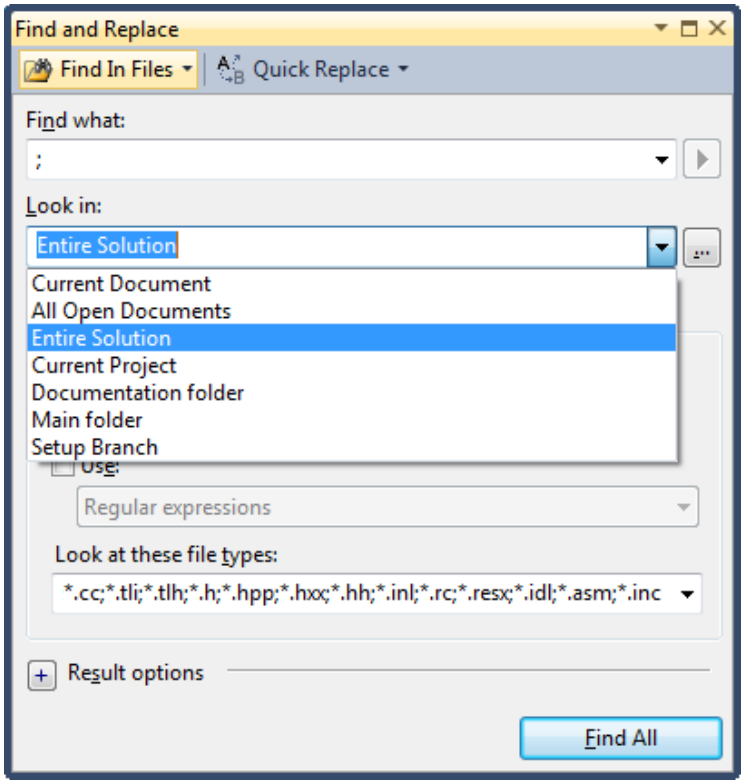


Quick Find

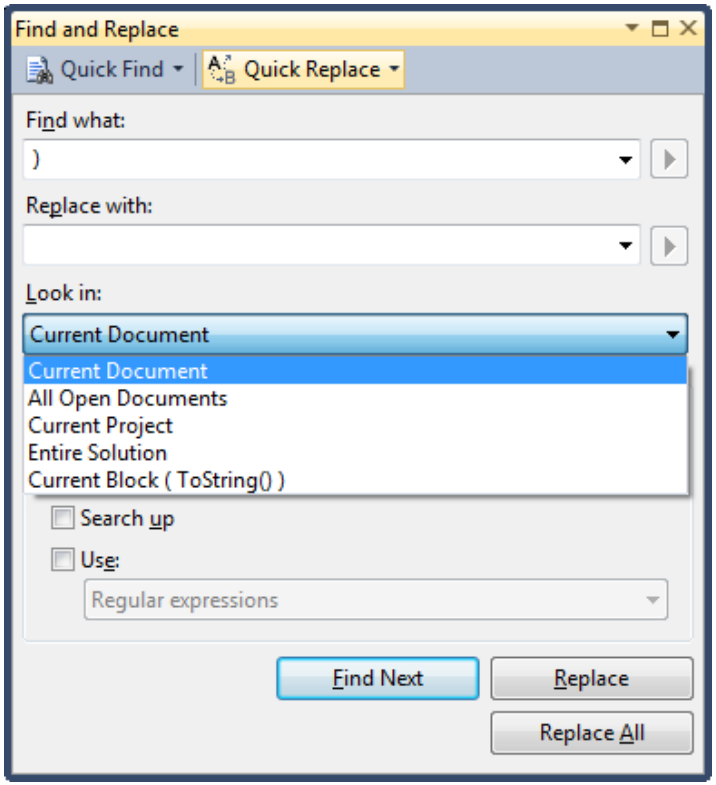
Ctrl+F

Find in Files

Ctrl+Shift+F

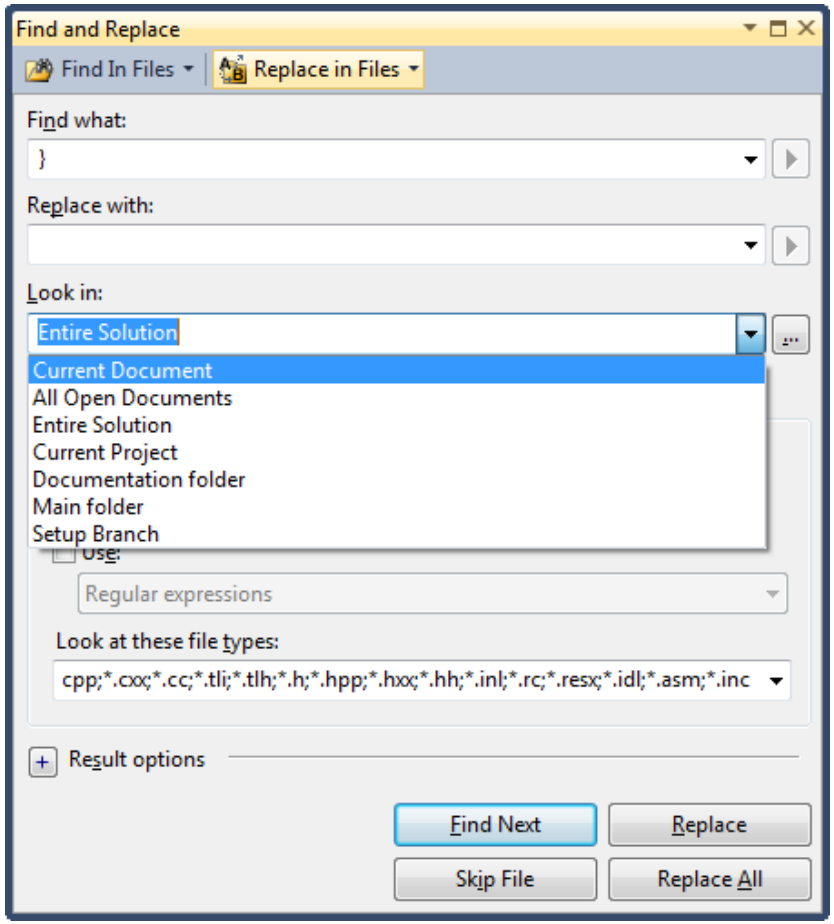


Suchen und Ersetzen (2/3)



Quick Replace

Ctrl+H



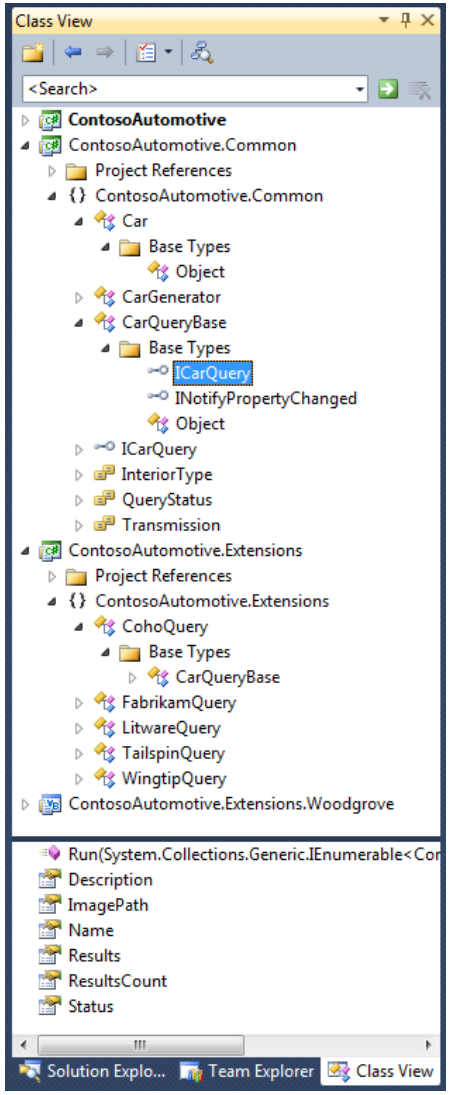
Replace in Files

Ctrl+Shift+H

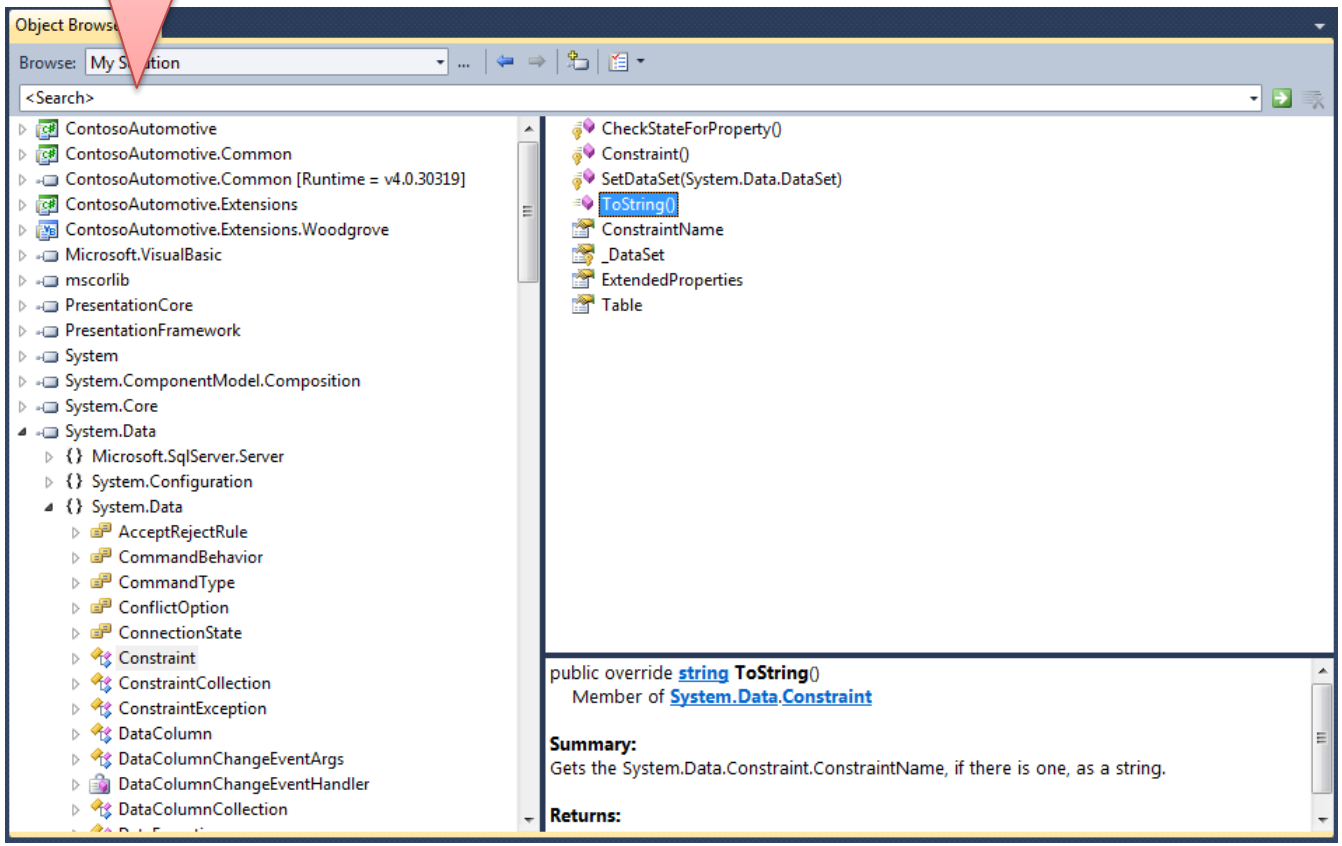
Navigate To (1/3)

- 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

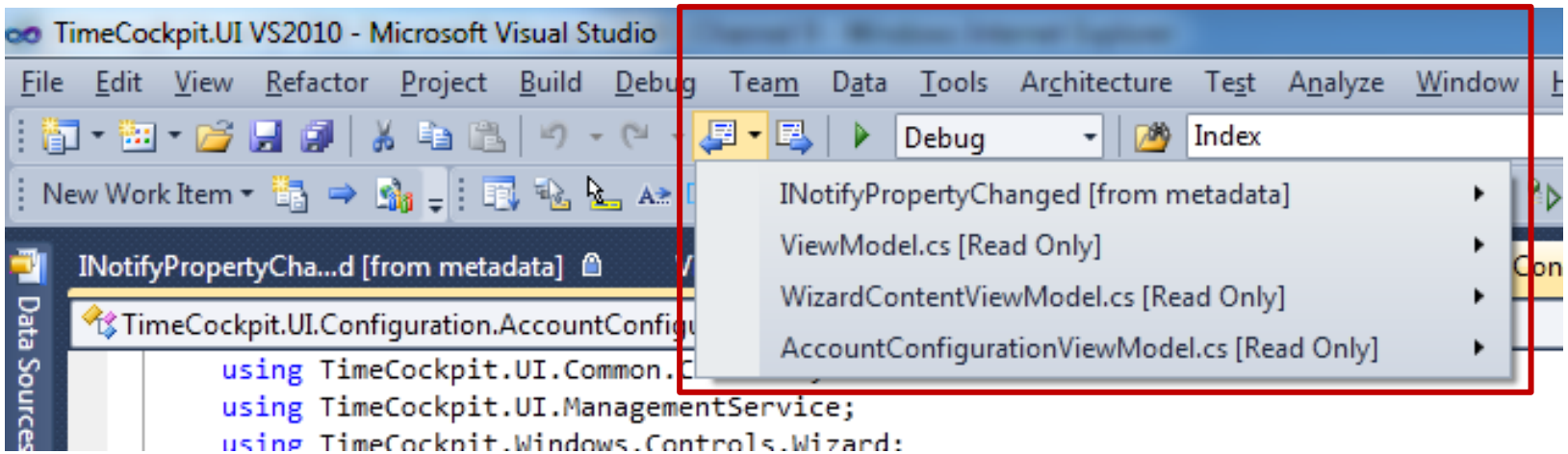


Navigate To (2/3)

- 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)

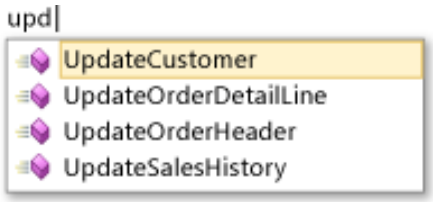
Code Definition Window

- *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*

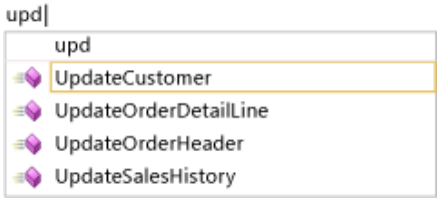
CODE GENERIEREN

IntelliSense Mode

- Modi
  - Completion Mode (wie bisher)



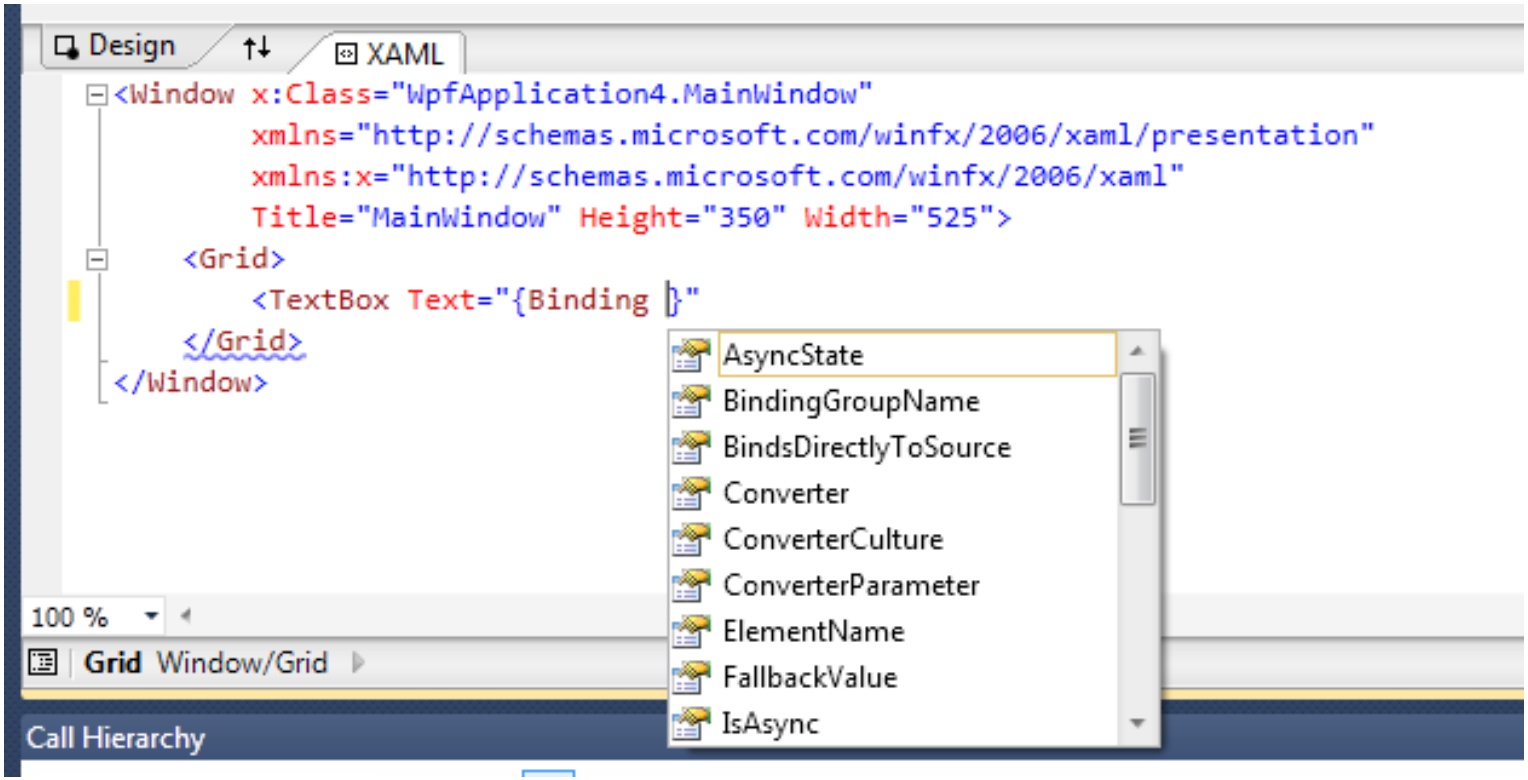
  - Suggestion Mode (für TDD; siehe *Generate From Usage*)



- 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 ☺ ☺ ☺



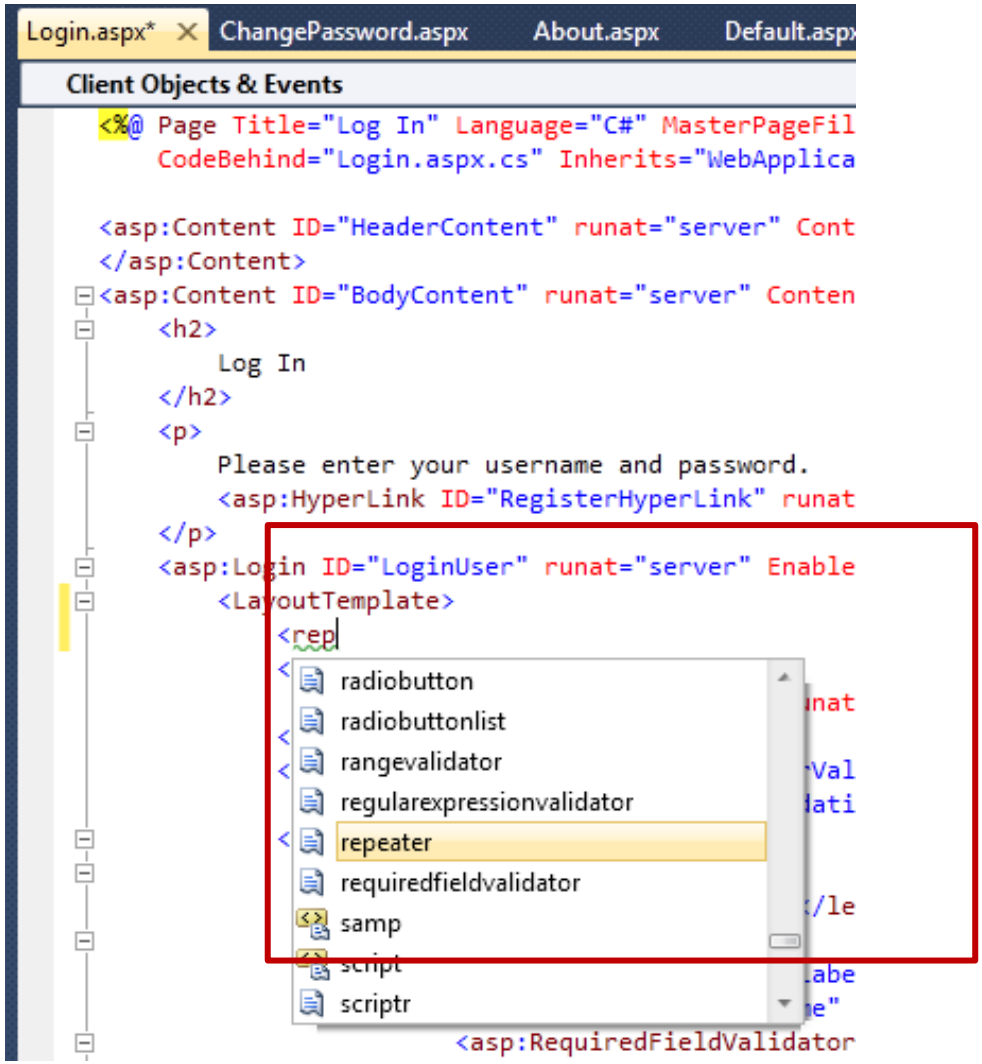
Generate From Usage (1/2)

- 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



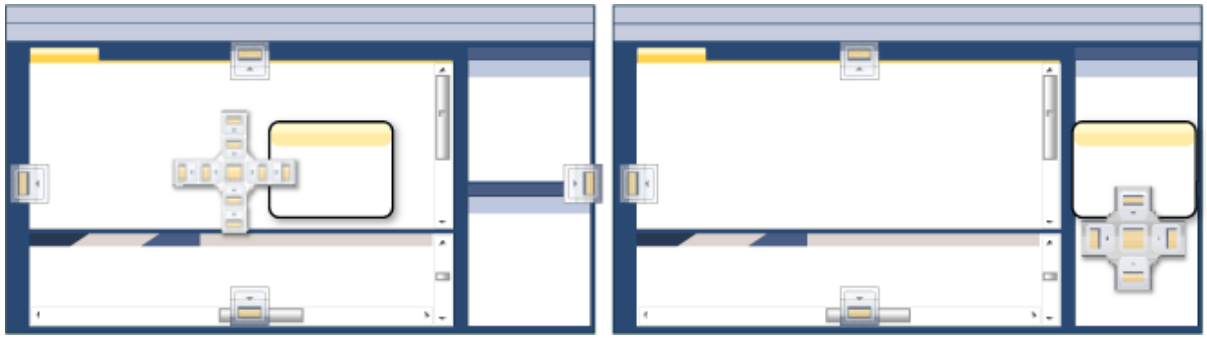
FENSTER- UND ANSICHTSVERWALTUNG

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

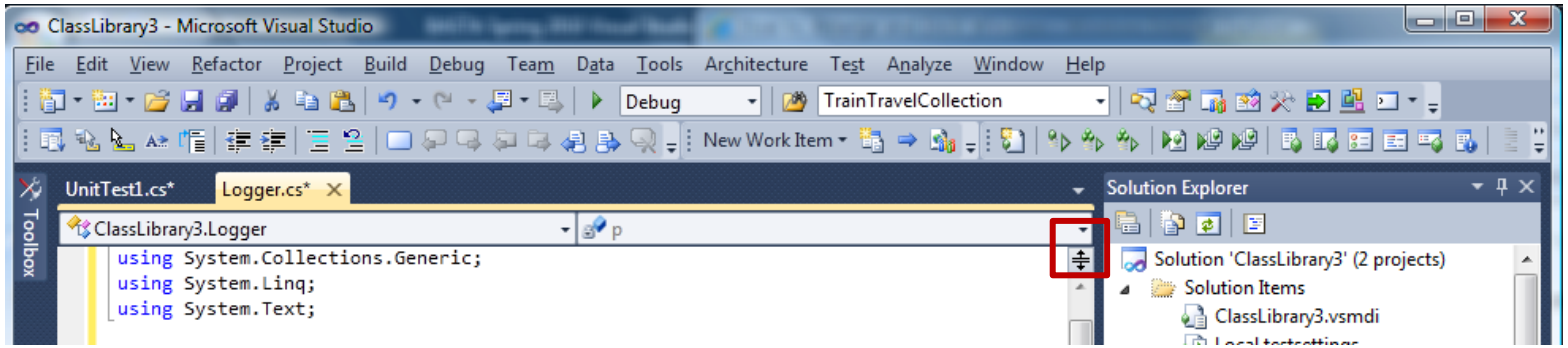
Docking (2/2)

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

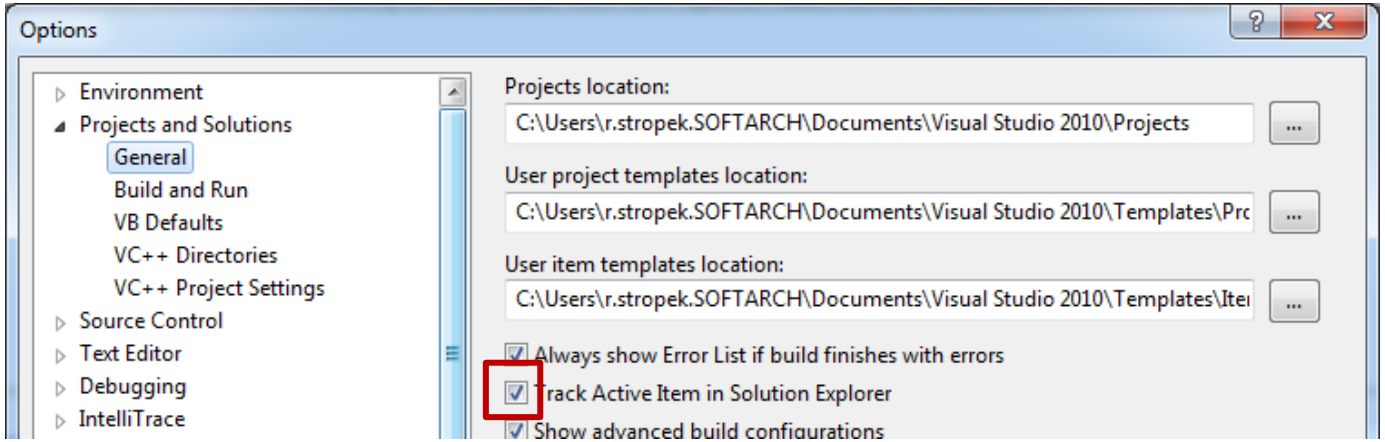


BTW – Kennen Sie den?

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



- *Track Active Item in Solution Explorer*



Selection

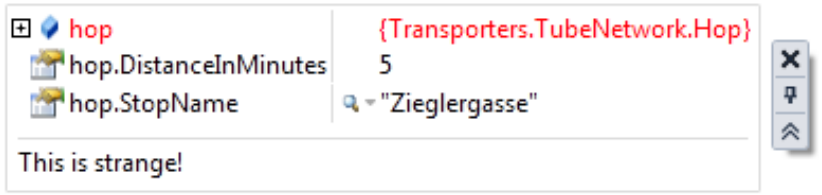
- Wie in früheren Version Boxed Selection mit Alt+Click&Drag
- Neu in VS2010
  - Multi-Line Insert
  - Paste
  - Zero-Length Boxes (multi-line insertion point)

DEBUGGING

Data Tips (1/2)

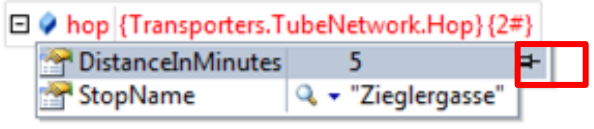
- Wie bisher im Debugger für Variablen im aktuellen Scope
  - Tipp: Data Tip transparent machen mit Ctrl
- Neu:
  - *Pin to source*: Data Tip ist mit Position im Sourcecode verknüpft und scrollt mit
  - Kommentare bei *pinned data tips*

```
};

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

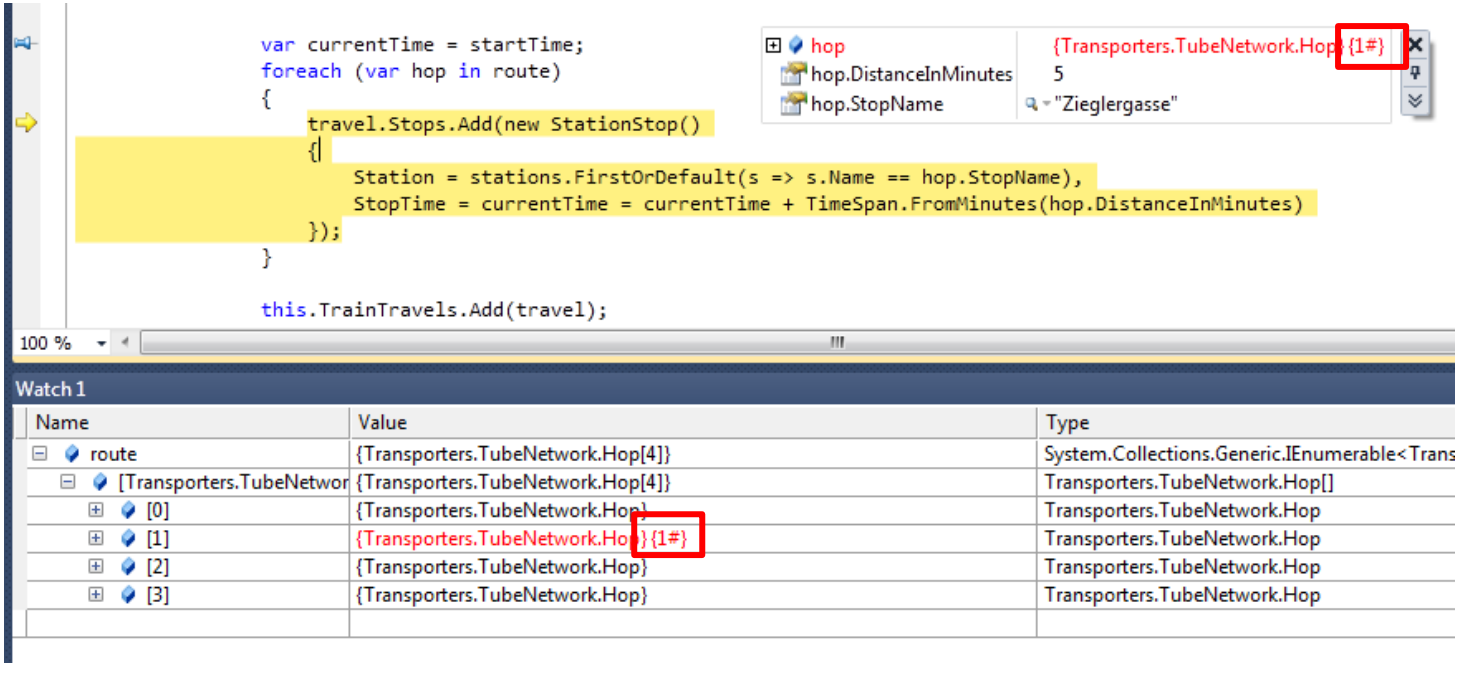


Data Tips (2/2)

- Pinning auch für Subexpressions möglich

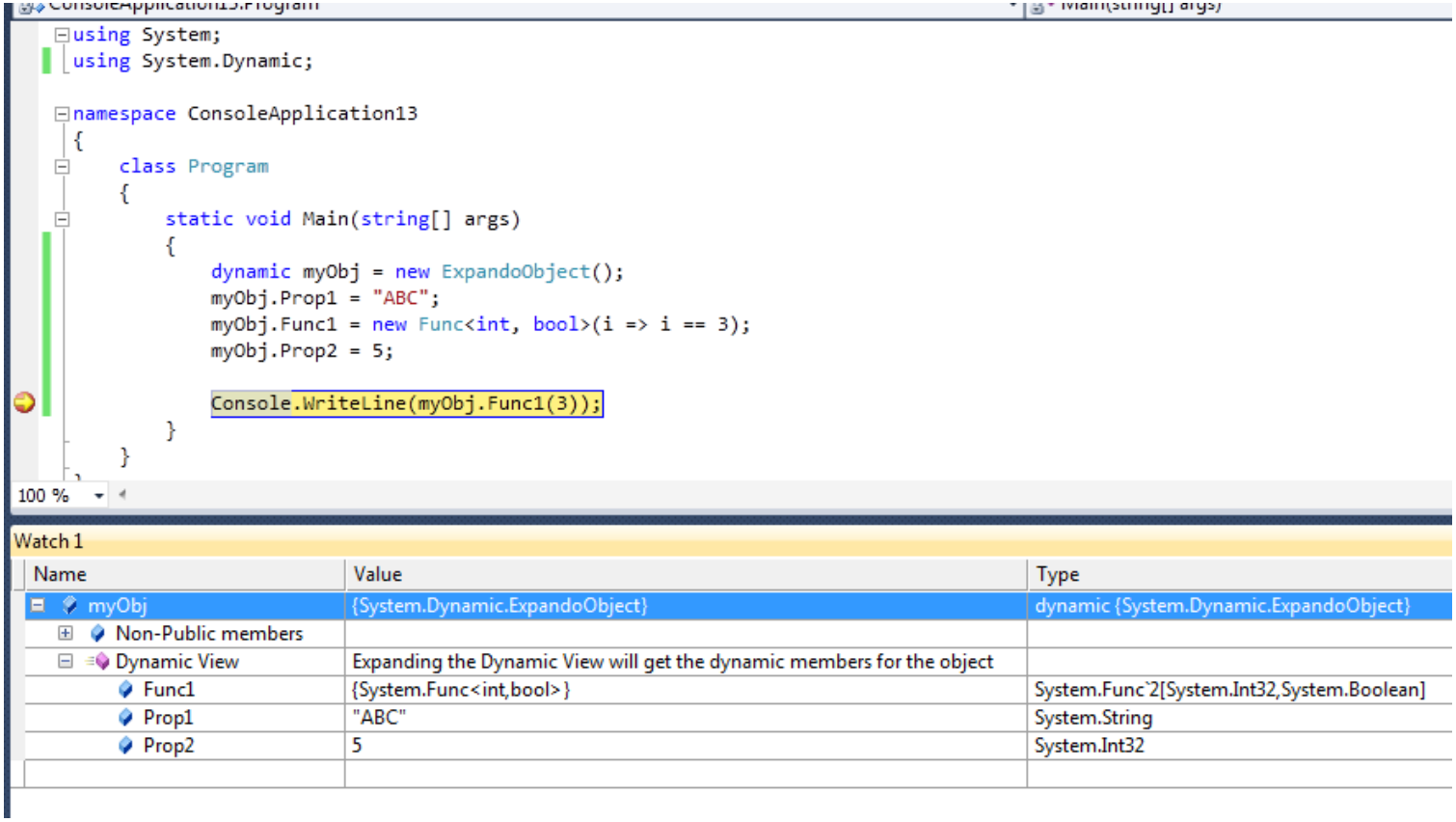


- Tipp: Object-IDs

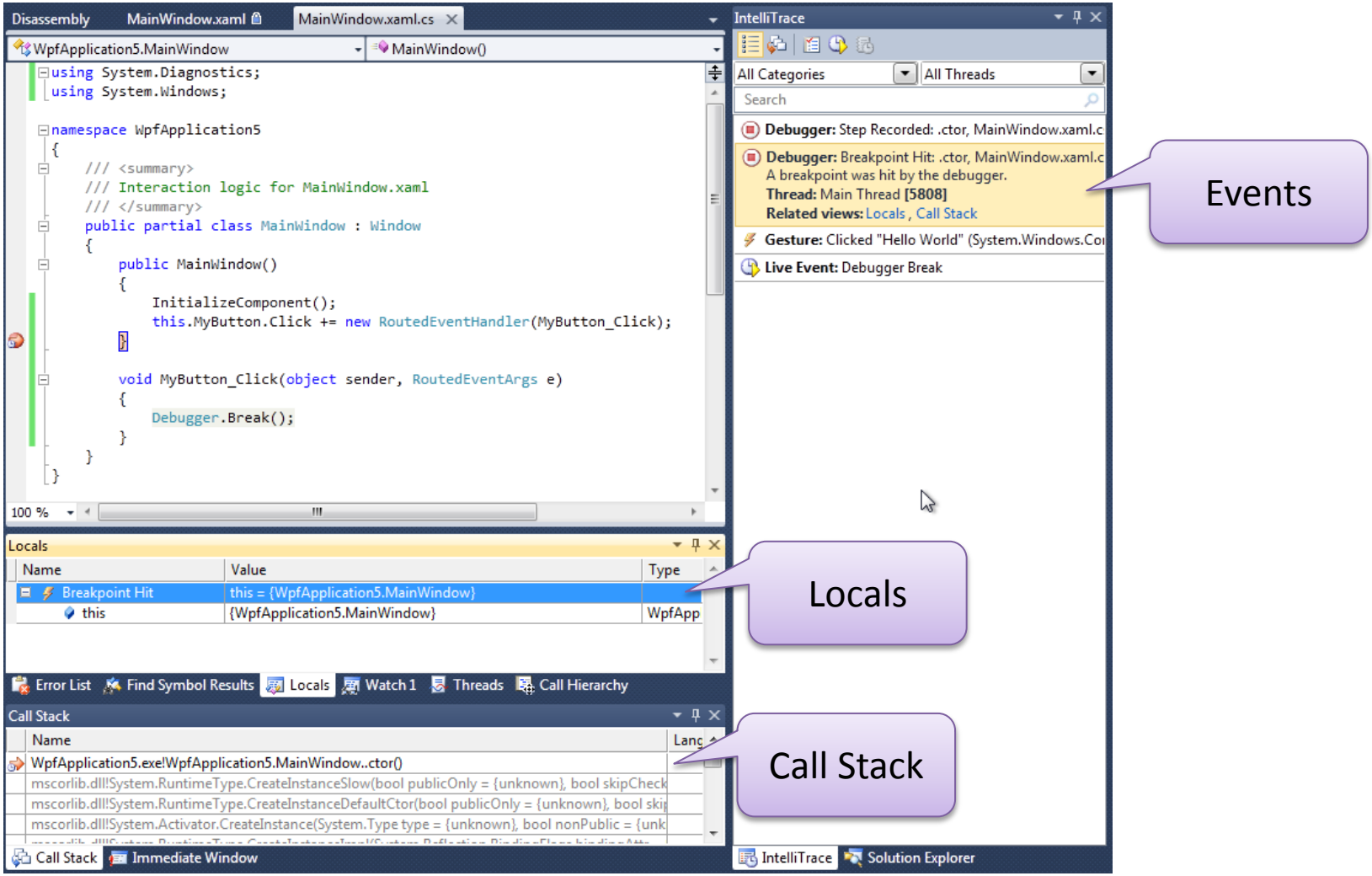


Unterstützung für DLR

- `dynamic` Datentyp wird im Debugger speziell unterstützt



IntelliTrace (1/2)



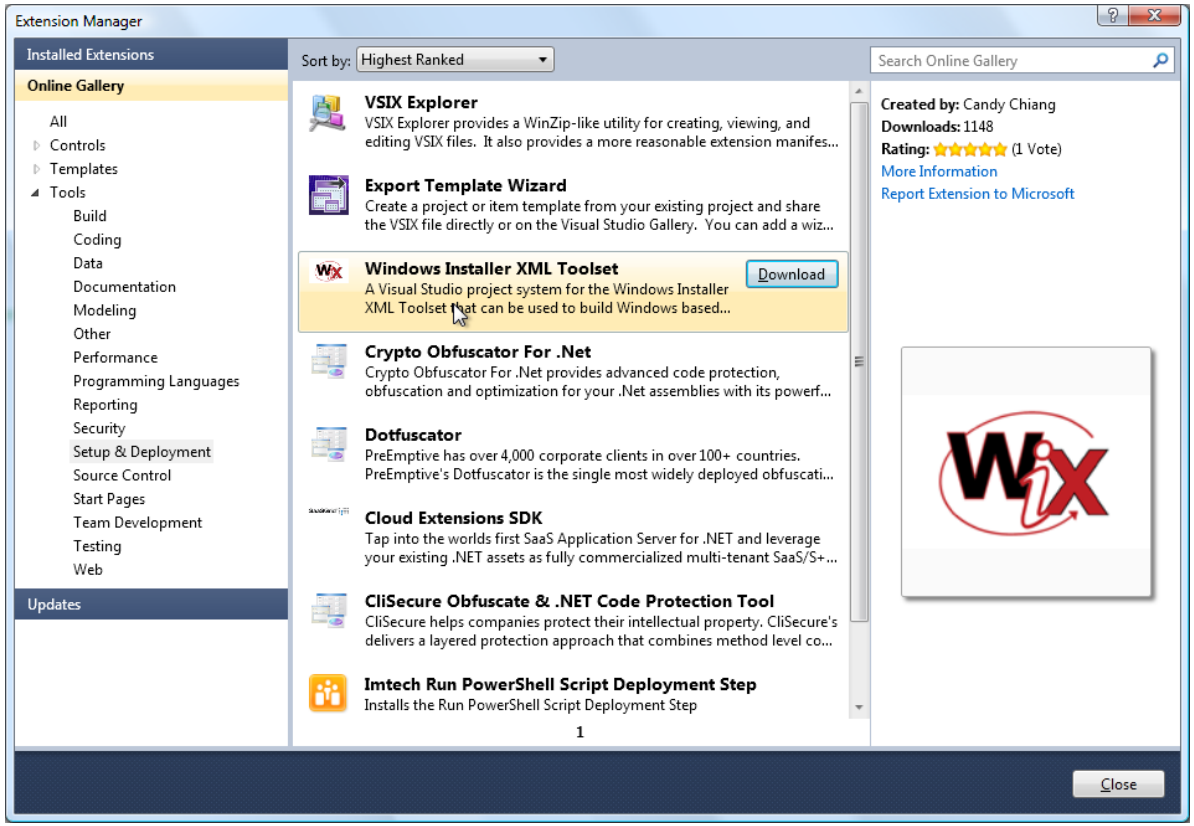
IntelliTrace (2/2)

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

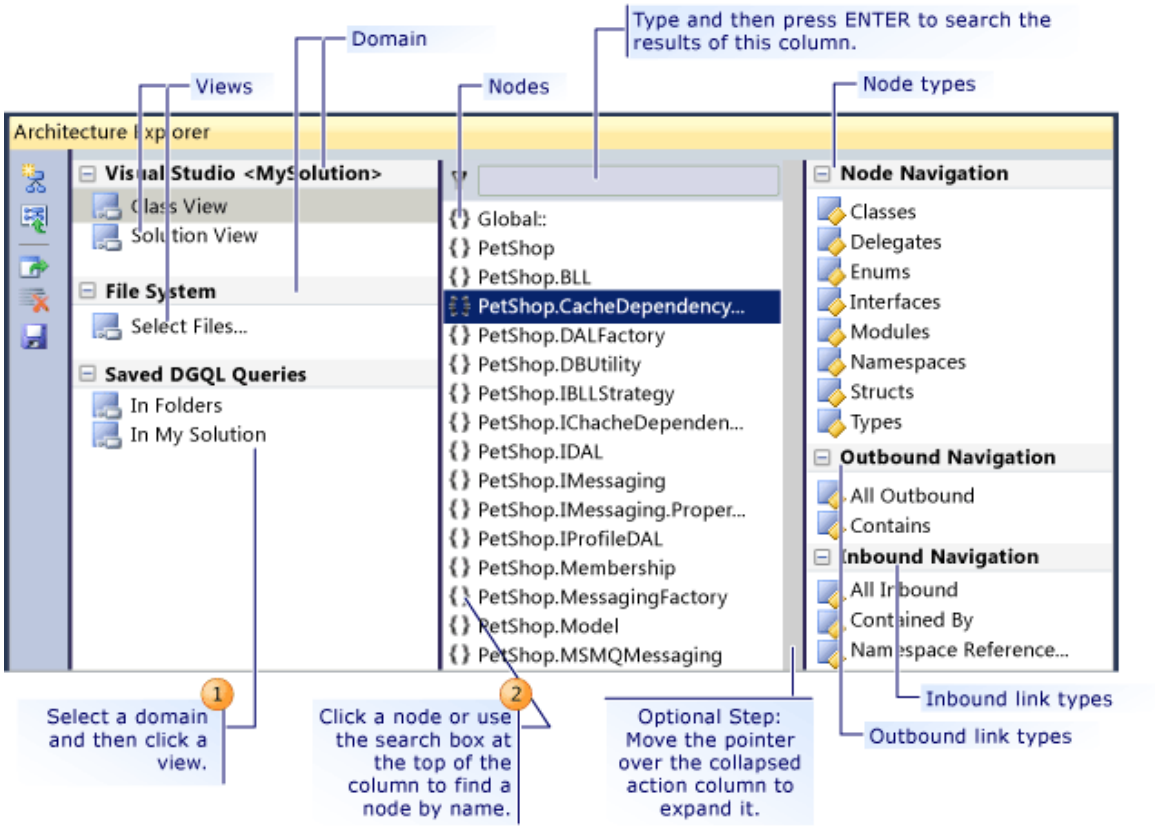
TOOLS

Extension Manager (1/2)

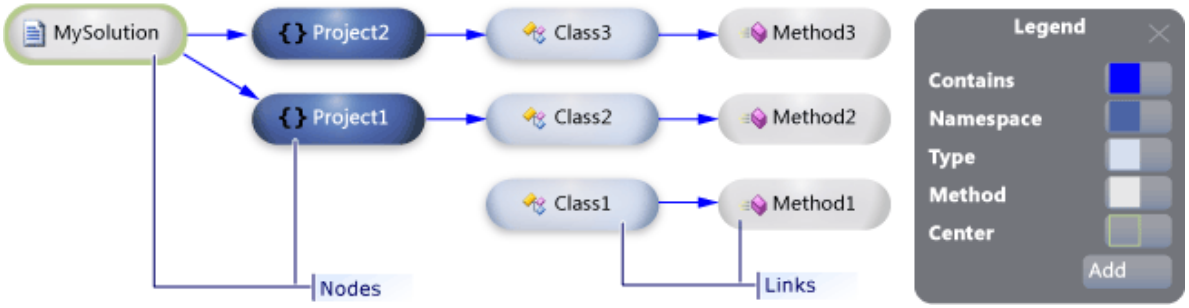
Tools, Extension Manager
(http://visualstudiogallery.msdn.microsoft.com/en-us/)



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

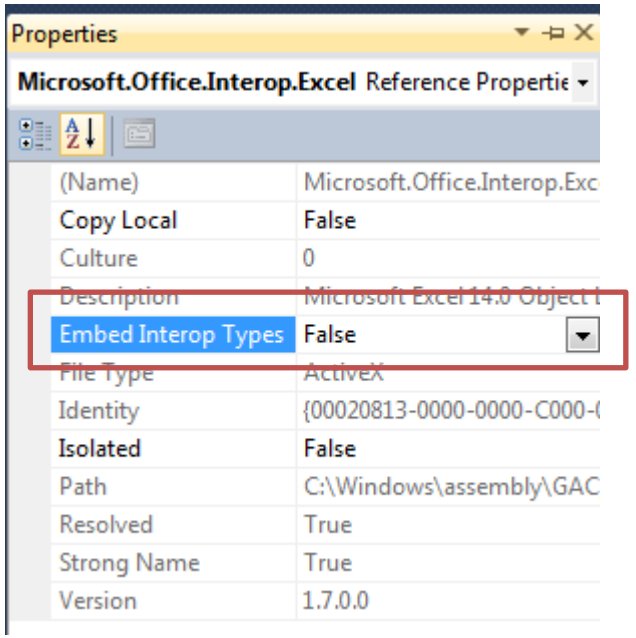
COM, No PIA, Optional Parameters, etc.

OFFICE INTEGRATION

Why Has Office Integration Been Hard?

- Primiary 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…

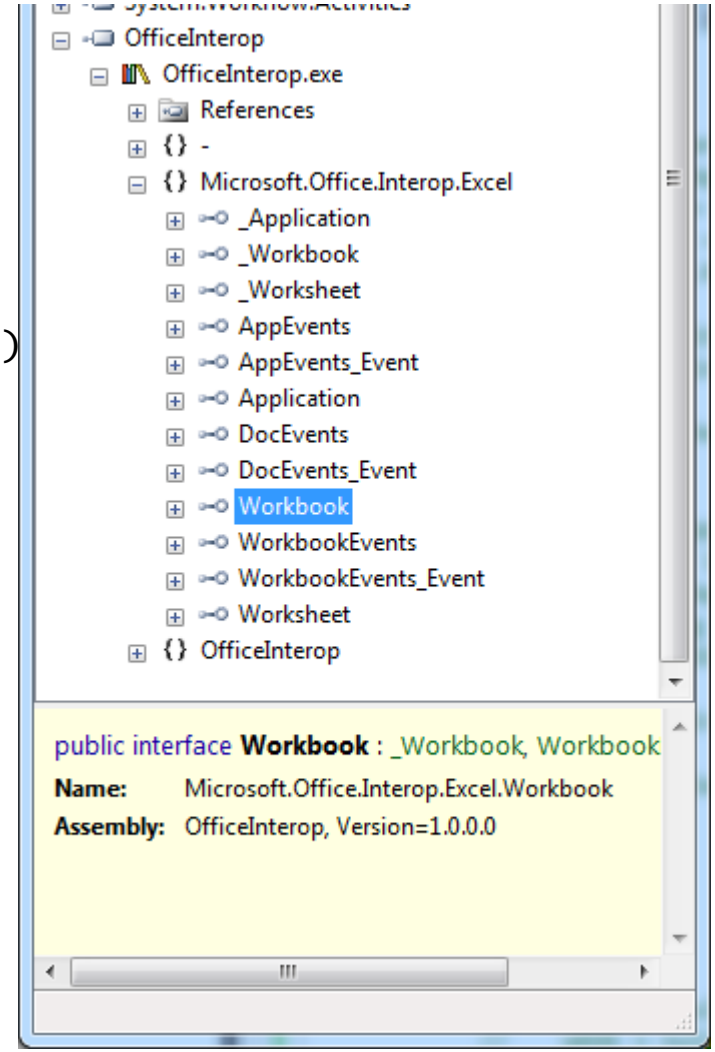
Solution: Embedded Interop Types



- `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 ☺

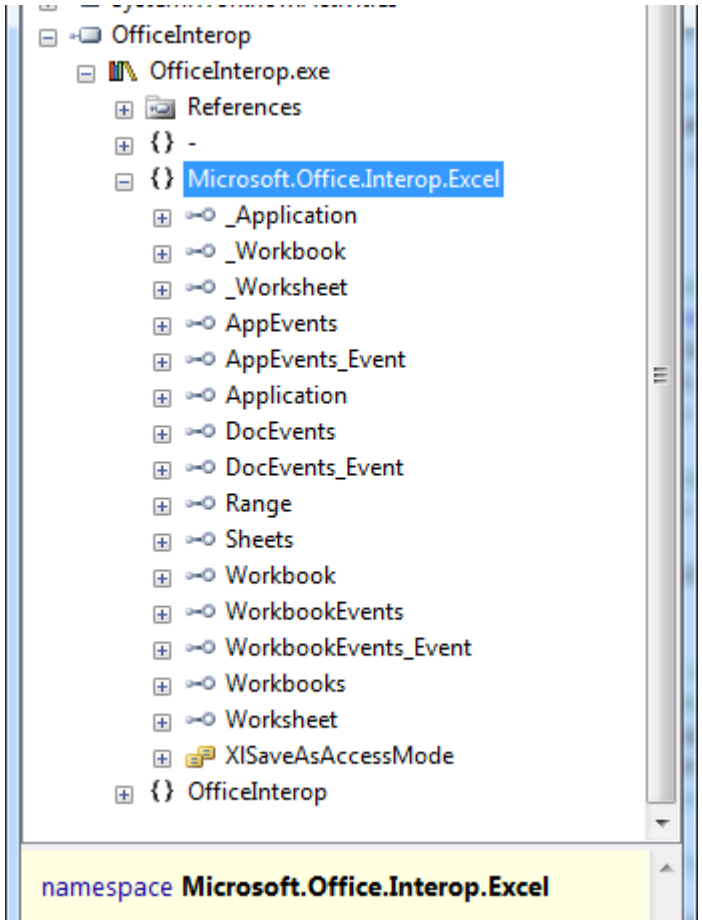
Embedded Interop Types

```
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 folderPath = Path.GetDirectoryName(filePath);
    if (!Directory.Exists(folderPath))
    {
        Directory.CreateDirectory(folderPath);
    }
    wkbk.SaveAs(filePath,
}
catc
{
}
fina
{

}
```

```
void _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

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

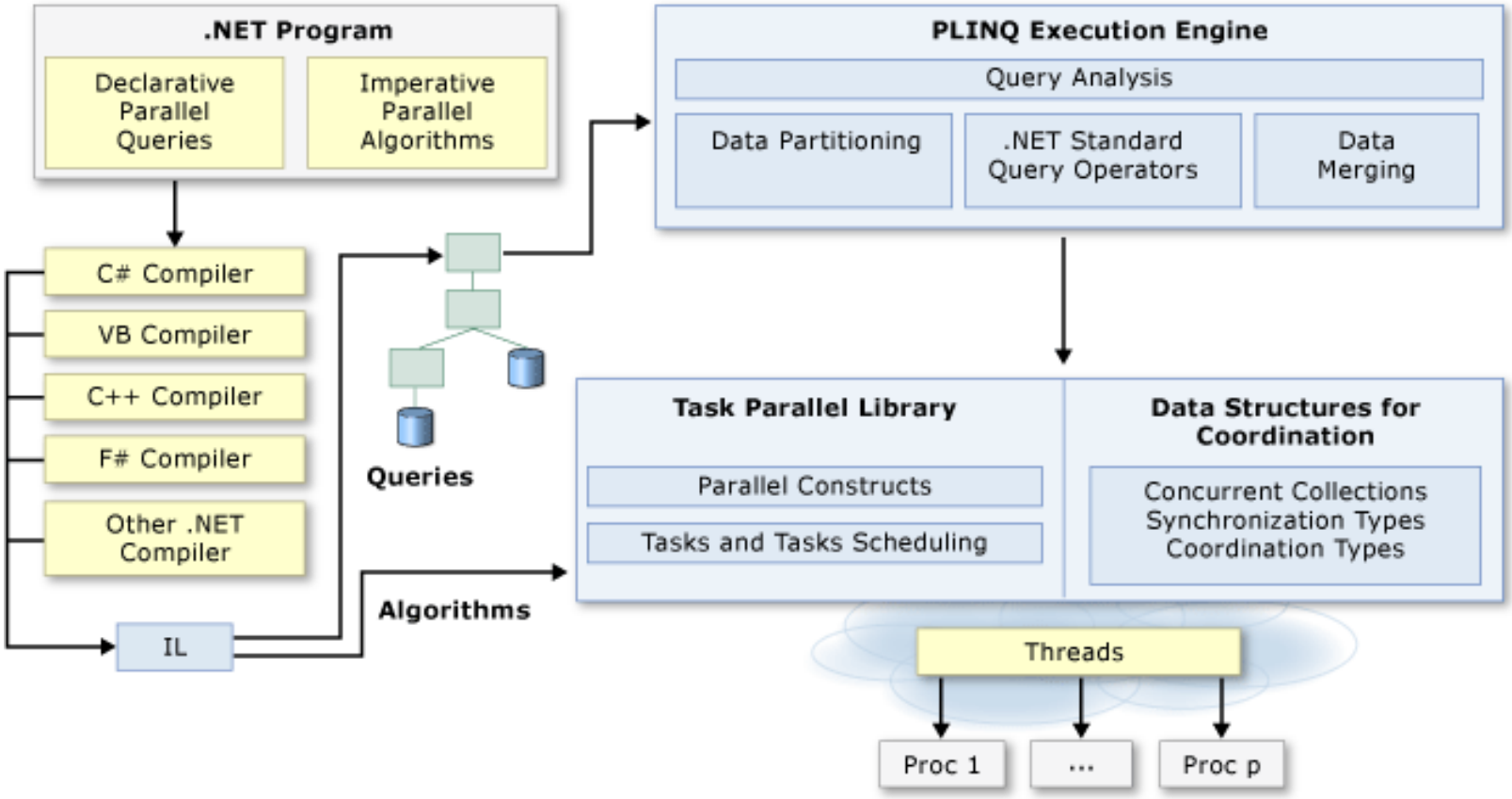
```
using System;

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

        static void Main()
        {
            DoSomething();
        }
    }
}
```

```
.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 ☹

MSDN

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!

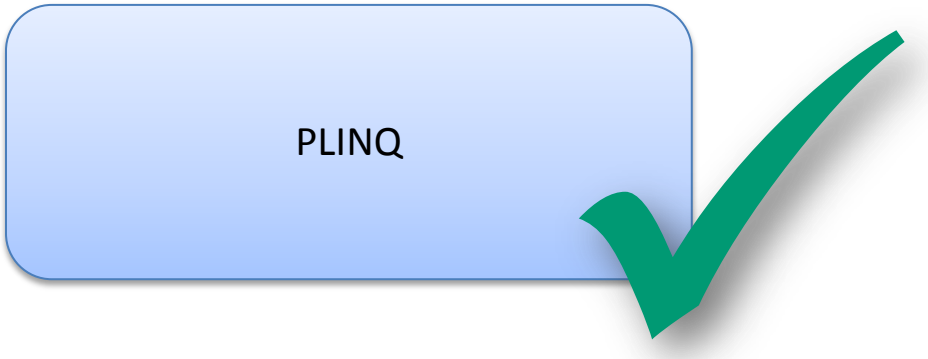
Schleifen - `Parallel.ForEach`

```
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();
   }));
```



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

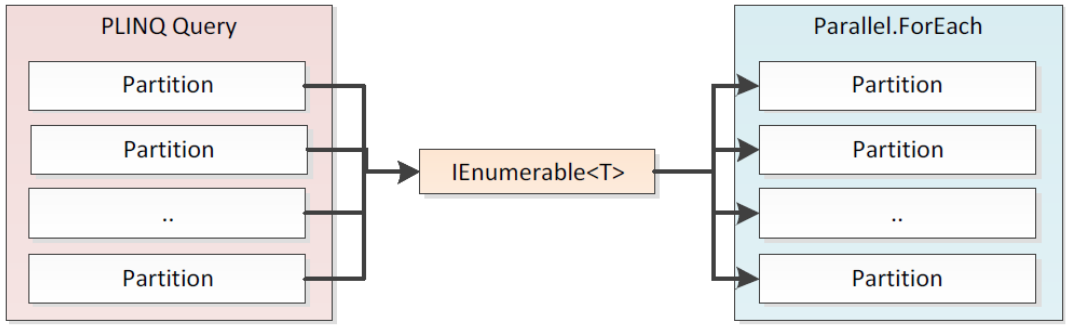
Performancetipps für PLINQ

- 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);
                    }
                }
            });
    }));
```

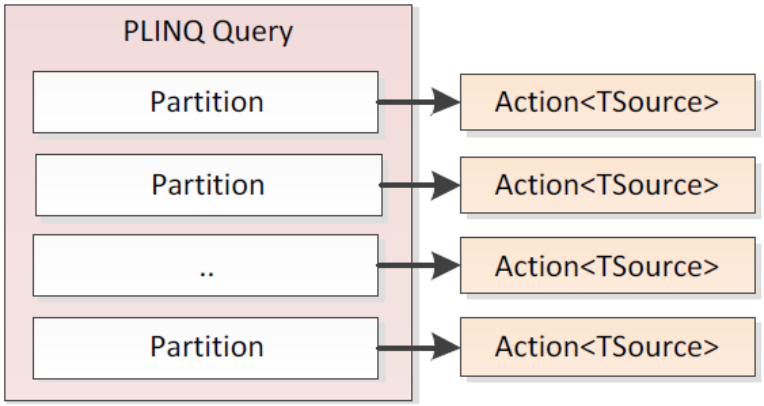
Parallel.ForEach verwendet `IEnumerable<T>` → unnötige Merge-Schritte



Was läuft hier falsch? (Code)

```
Console.WriteLine(
    "Paralleler Durchlauf mit Parallel.ForAll: {0}",
    MeasuringTools.Measure(() =>
    {
        source.AsParallel().ForAll(
            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 => { … });
```

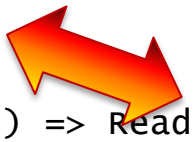
Was läuft hier falsch? (Code)

```
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[urls.Length];
            for (int i = 0; i < urls.Length; i++)
            {
                tasks[i] = Task.Factory.StartNew(() => ReadUrl(urls[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[urls.Length];
for (int i = 0; i < urls.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[urls.Length];
for (int i = 0; i < urls.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 procesing
            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)

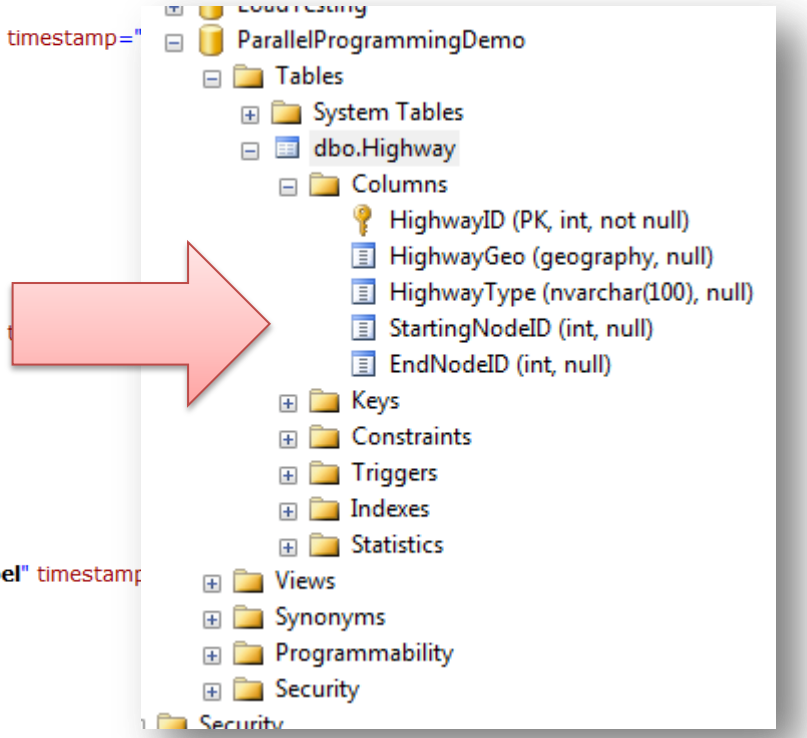
```
var buffer = new BlockingCollection<long>(10);
var cancelTokenSource = new CancellationTokenSource();

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:(
    <node id="172540" lat="52.5647252" lon="13.3364064" version="7" changeset="5702878" user="Woschl" uid="121042" timestamp="2010-09-06T21:(
    <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="2010-09-06T19:3
      <tag k="highway" v="traffic_signals" />
    </node>
  - <way id="30770007" version="2" changeset="2121805" uid="6669" user="Elwood" timestamp="
      <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" t
      <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
      <nd ref="172544" />
      <nd ref="172545" />
      <tag k="highway" v="footway" />
    </way>
  </osm>
```

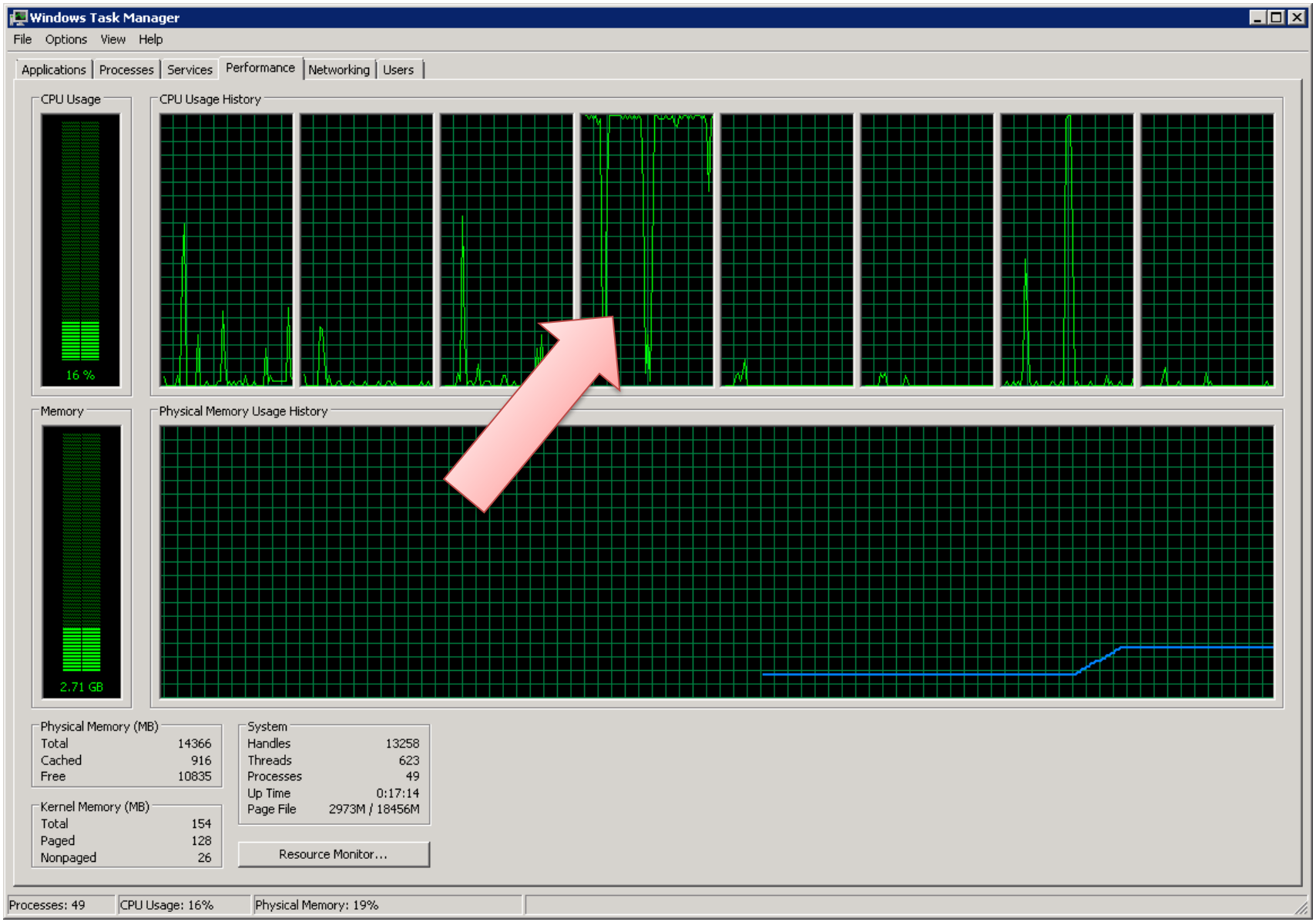


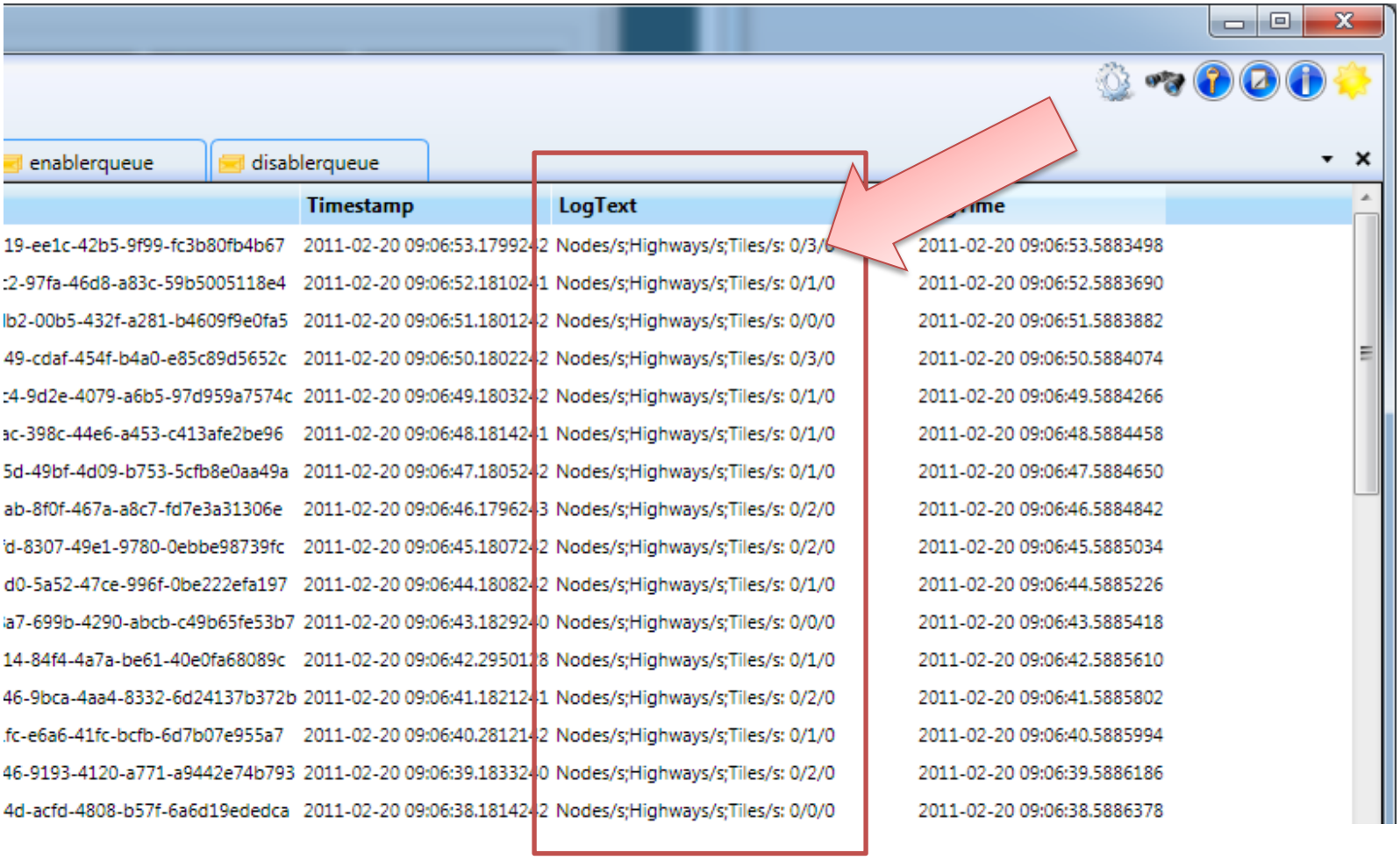
Mission Impossible

- 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

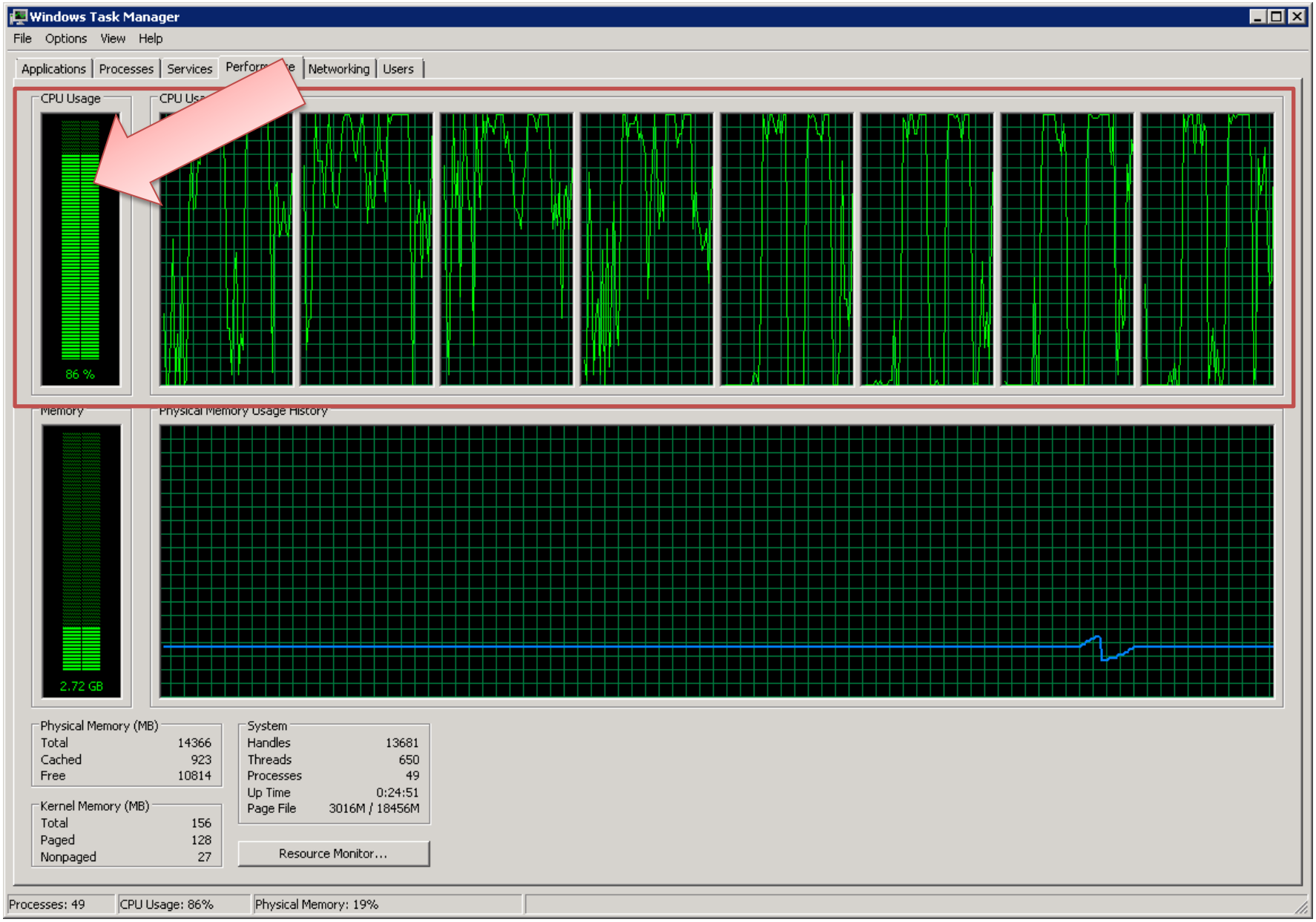
```
XElement tmpNode;
XDocument doc;
foreach (var row in
    (doc = XDocument.Load(@"https://loadtesting.blob.core.windows.net/osm/berlin.xml"))
    .Descendants("way")
    .Where(w => w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").Count() > 0)
    .Select(w =>
        new
        {
            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
}
```

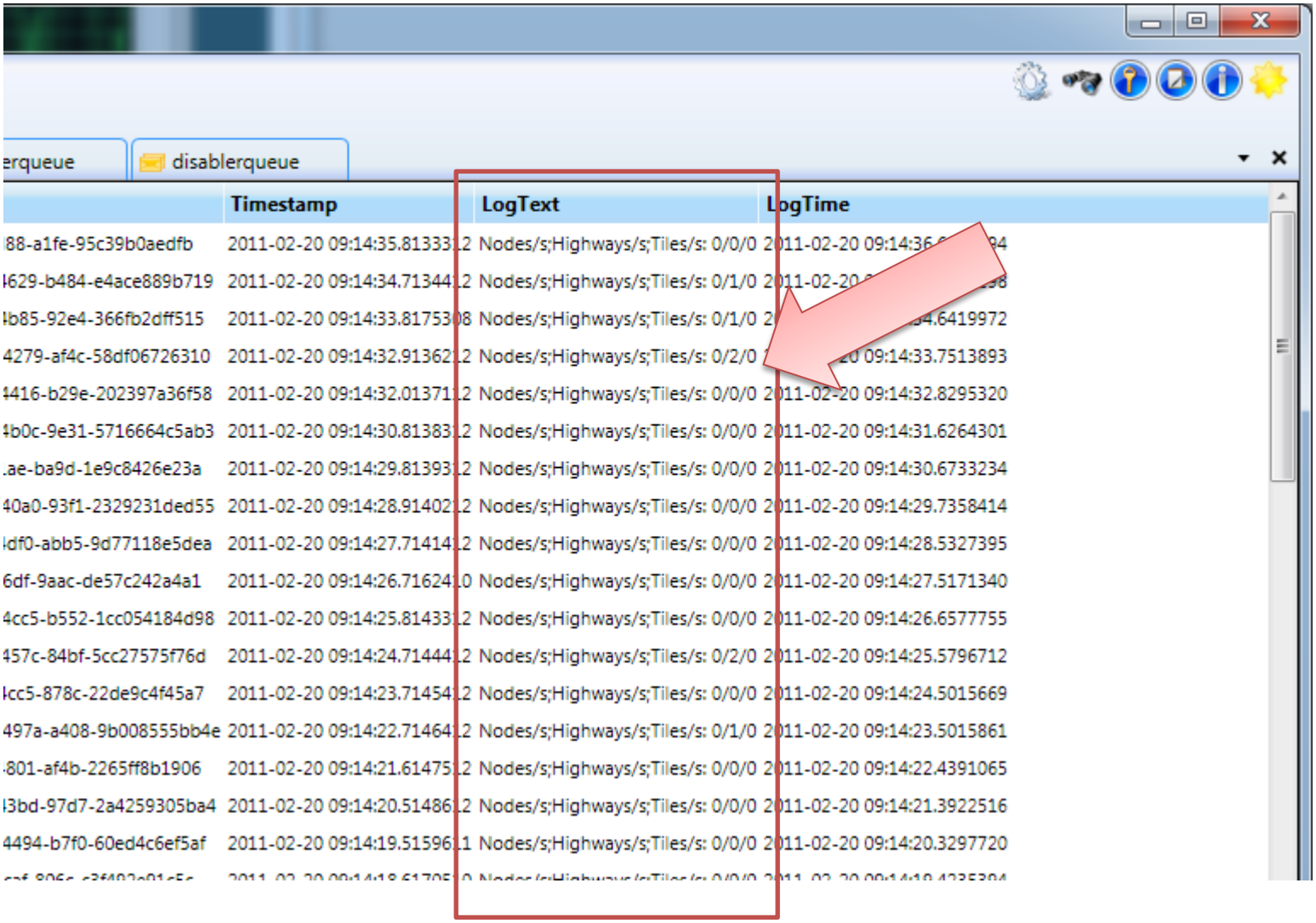

Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
CPU Usage
16 %
CPU Usage History
Memory
2.71 GB
Physical Memory Usage History
Physical Memory (MB)
Total 14366
Cached 916
Free 10835
Kernel Memory (MB)
Total 154
Paged 128
Nonpaged 26
System
Handles 13258
Threads 623
Processes 49
Up Time 0:17:14
Page File 2973M / 18456M
Resource Monitor...
Processes: 49
CPU Usage: 16%
Physical Memory: 19%


enablerqueue
disablerqueue
Timestamp
LogText
19-ee1c-42b5-9f99-fc3b80fb4b67 2011-02-20 09:06:53.1799242 Nodes/s;Highways/s;Tiles/s: 0/3/0 2011-02-20 09:06:53.5883498
:2-97fa-46d8-a83c-59b5005118e4 2011-02-20 09:06:52.1810241 Nodes/s;Highways/s;Tiles/s: 0/1/0 2011-02-20 09:06:52.5883690
lb2-00b5-432f-a281-b4609f9e0fa5 2011-02-20 09:06:51.1801242 Nodes/s;Highways/s;Tiles/s: 0/0/0 2011-02-20 09:06:51.5883882
49-cdaf-454f-b4a0-e85c89d5652c 2011-02-20 09:06:50.1802242 Nodes/s;Highways/s;Tiles/s: 0/3/0 2011-02-20 09:06:50.5884074
:4-9d2e-4079-a6b5-97d959a7574c 2011-02-20 09:06:49.1803242 Nodes/s;Highways/s;Tiles/s: 0/1/0 2011-02-20 09:06:49.5884266
ac-398c-44e6-a453-c413afe2be96 2011-02-20 09:06:48.1814241 Nodes/s;Highways/s;Tiles/s: 0/1/0 2011-02-20 09:06:48.5884458
5d-49bf-4d09-b753-5cfb8e0aa49a 2011-02-20 09:06:47.1805242 Nodes/s;Highways/s;Tiles/s: 0/1/0 2011-02-20 09:06:47.5884650
ab-8f0f-467a-a8c7-fd7e3a31306e 2011-02-20 09:06:46.1796243 Nodes/s;Highways/s;Tiles/s: 0/2/0 2011-02-20 09:06:46.5884842
'd-8307-49e1-9780-0ebbe98739fc 2011-02-20 09:06:45.1807242 Nodes/s;Highways/s;Tiles/s: 0/2/0 2011-02-20 09:06:45.5885034
d0-5a52-47ce-996f-0be222efa197 2011-02-20 09:06:44.1808242 Nodes/s;Highways/s;Tiles/s: 0/1/0 2011-02-20 09:06:44.5885226
a7-699b-4290-abcb-c49b65fe53b7 2011-02-20 09:06:43.1829240 Nodes/s;Highways/s;Tiles/s: 0/0/0 2011-02-20 09:06:43.5885418
14-84f4-4a7a-be61-40e0fa68089c 2011-02-20 09:06:42.2950128 Nodes/s;Highways/s;Tiles/s: 0/1/0 2011-02-20 09:06:42.5885610
46-9bca-4aa4-8332-6d24137b372b 2011-02-20 09:06:41.1821241 Nodes/s;Highways/s;Tiles/s: 0/2/0 2011-02-20 09:06:41.5885802
.fc-e6a6-41fc-bcfb-6d7b07e955a7 2011-02-20 09:06:40.2812142 Nodes/s;Highways/s;Tiles/s: 0/1/0 2011-02-20 09:06:40.5885994
46-9193-4120-a771-a9442e74b793 2011-02-20 09:06:39.1833240 Nodes/s;Highways/s;Tiles/s: 0/2/0 2011-02-20 09:06:39.5886186
4d-acfd-4808-b57f-6a6d19ededca 2011-02-20 09:06:38.1814242 Nodes/s;Highways/s;Tiles/s: 0/0/0 2011-02-20 09:06:38.5886378

Solution 2: Making it Parallel

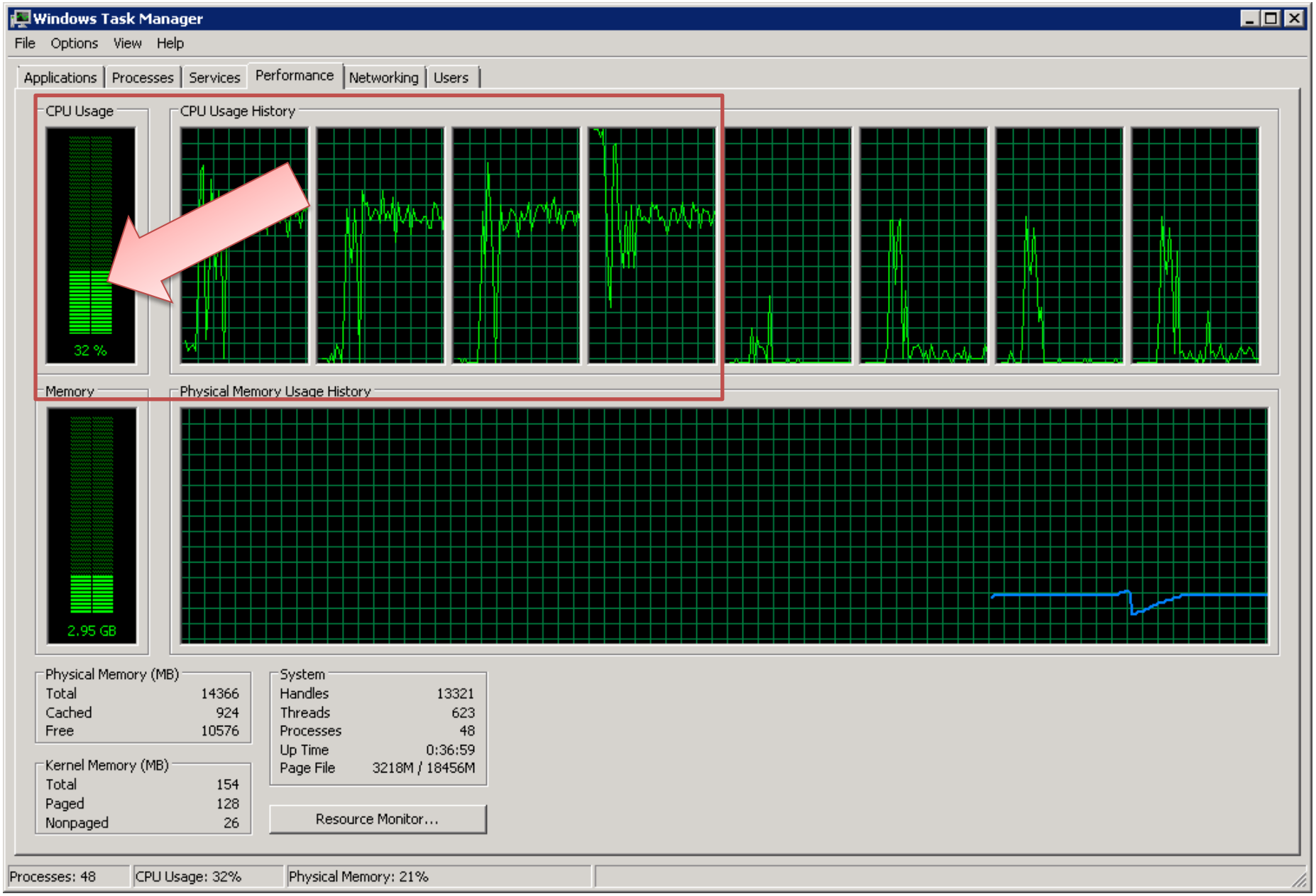
```
XElement tmpNode;
XDocument doc;
(doc = XDocument.Load(@"https://loadtesting.blob.core.windows.net/osm/berlin.xml"))
    .Descendants("way")
    .AsParallel()
    .Where(w => w.Descendants("tag").Where(t => t.Attribute("k").Value == "highway").Count() > 0)
    .Select(w =>
        new
        {
            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
    });
```

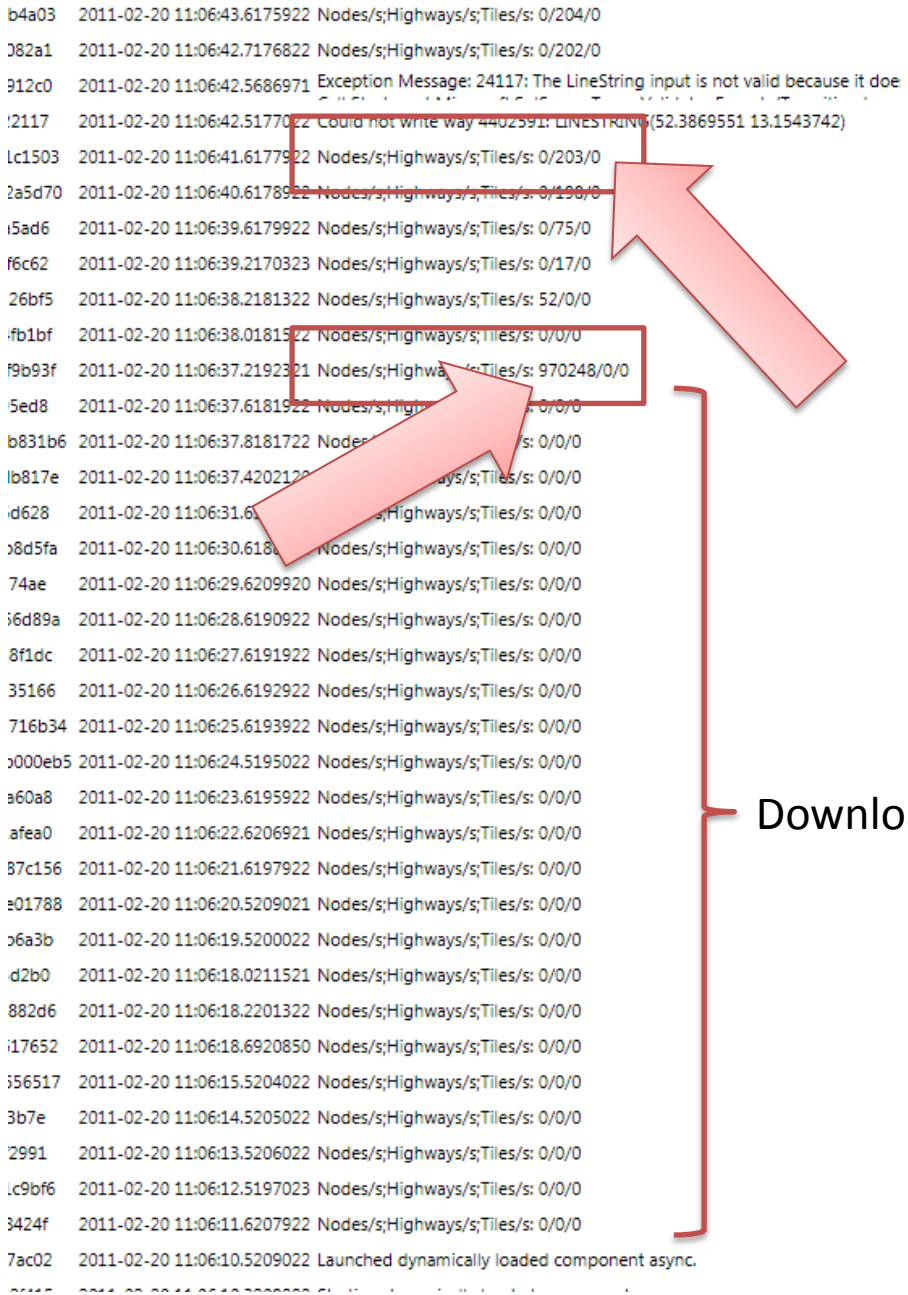

Windows Task Manager
File Options View Help
Applications Processes Services Networking Users
CPU Usage
86 %
Memory
2.72 GB
Physical Memory Usage History
Physical Memory (MB)
Total 14366
Cached 923
Free 10814
Kernel Memory (MB)
Total 156
Paged 128
Nonpaged 27
System
Handles 13681
Threads 650
Processes 49
Up Time 0:24:51
Page File 3016M / 18456M
Resource Monitor...
Processes: 49
CPU Usage: 86%
Physical Memory: 19%


erqueue
disablerqueue
Timestamp
LogText
LogTime
88-a1fe-95c39b0aedfb 2011-02-20 09:14:35.8133312 Nodes/s;Highways/s;Tiles/s: 0/0/0
629-b484-e4ace889b719 2011-02-20 09:14:34.7134412 Nodes/s;Highways/s;Tiles/s: 0/1/0
b85-92e4-366fb2dff515 2011-02-20 09:14:33.8175308 Nodes/s;Highways/s;Tiles/s: 0/1/0
4279-af4c-58df06726310 2011-02-20 09:14:32.9136212 Nodes/s;Highways/s;Tiles/s: 0/2/0
4416-b29e-202397a36f58 2011-02-20 09:14:32.0137112 Nodes/s;Highways/s;Tiles/s: 0/0/0
4b0c-9e31-5716664c5ab3 2011-02-20 09:14:30.8138312 Nodes/s;Highways/s;Tiles/s: 0/0/0
.ae-ba9d-1e9c8426e23a 2011-02-20 09:14:29.8139312 Nodes/s;Highways/s;Tiles/s: 0/0/0
40a0-93f1-2329231ded55 2011-02-20 09:14:28.9140212 Nodes/s;Highways/s;Tiles/s: 0/0/0
df0-abb5-9d77118e5dea 2011-02-20 09:14:27.7141412 Nodes/s;Highways/s;Tiles/s: 0/0/0
6df-9aac-de57c242a4a1 2011-02-20 09:14:26.7162410 Nodes/s;Highways/s;Tiles/s: 0/0/0
4cc5-b552-1cc054184d98 2011-02-20 09:14:25.8143312 Nodes/s;Highways/s;Tiles/s: 0/0/0
457c-84bf-5cc27575f76d 2011-02-20 09:14:24.7144412 Nodes/s;Highways/s;Tiles/s: 0/2/0
cc5-878c-22de9c4f45a7 2011-02-20 09:14:23.7145412 Nodes/s;Highways/s;Tiles/s: 0/0/0
497a-a408-9b008555bb4e 2011-02-20 09:14:22.7146412 Nodes/s;Highways/s;Tiles/s: 0/1/0
801-af4b-2265ff8b1906 2011-02-20 09:14:21.6147512 Nodes/s;Highways/s;Tiles/s: 0/0/0
3bd-97d7-2a4259305ba4 2011-02-20 09:14:20.5148612 Nodes/s;Highways/s;Tiles/s: 0/0/0
4494-b7f0-60ed4c6ef5af 2011-02-20 09:14:19.5159611 Nodes/s;Highways/s;Tiles/s: 0/0/0
2011-02-20 09:14:33.7513893
2011-02-20 09:14:32.8295320
2011-02-20 09:14:31.6264301
2011-02-20 09:14:30.6733234
2011-02-20 09:14:29.7358414
2011-02-20 09:14:28.5327395
2011-02-20 09:14:27.5171340
2011-02-20 09:14:26.6577755
2011-02-20 09:14:25.5796712
2011-02-20 09:14:24.5015669
2011-02-20 09:14:23.5015861
2011-02-20 09:14:22.4391065
2011-02-20 09:14:21.3922516
2011-02-20 09:14:20.3297720

Solution 3: Enhaced 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
    });
}
```


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
CPU Usage
32 %
CPU Usage History
Memory
2.95 GB
Physical Memory Usage History
Physical Memory (MB)
Total 14366
Cached 924
Free 10576
Kernel Memory (MB)
Total 154
Paged 128
Nonpaged 26
System
Handles 13321
Threads 623
Processes 48
Up Time 0:36:59
Page File 3218M / 18456M
Resource Monitor...
Processes: 48
CPU Usage: 32%
Physical Memory: 21%



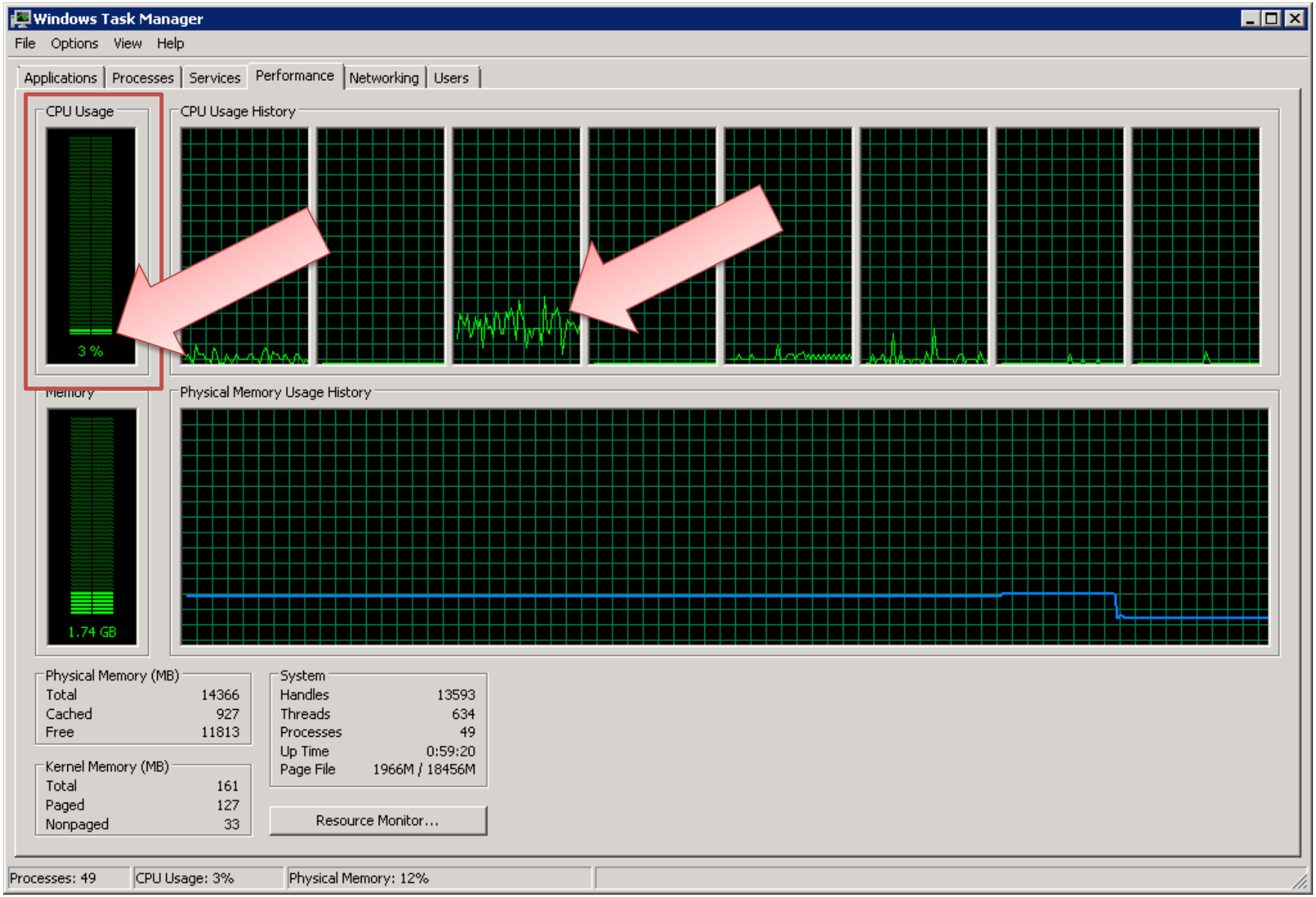
Total: 5,835 Min.

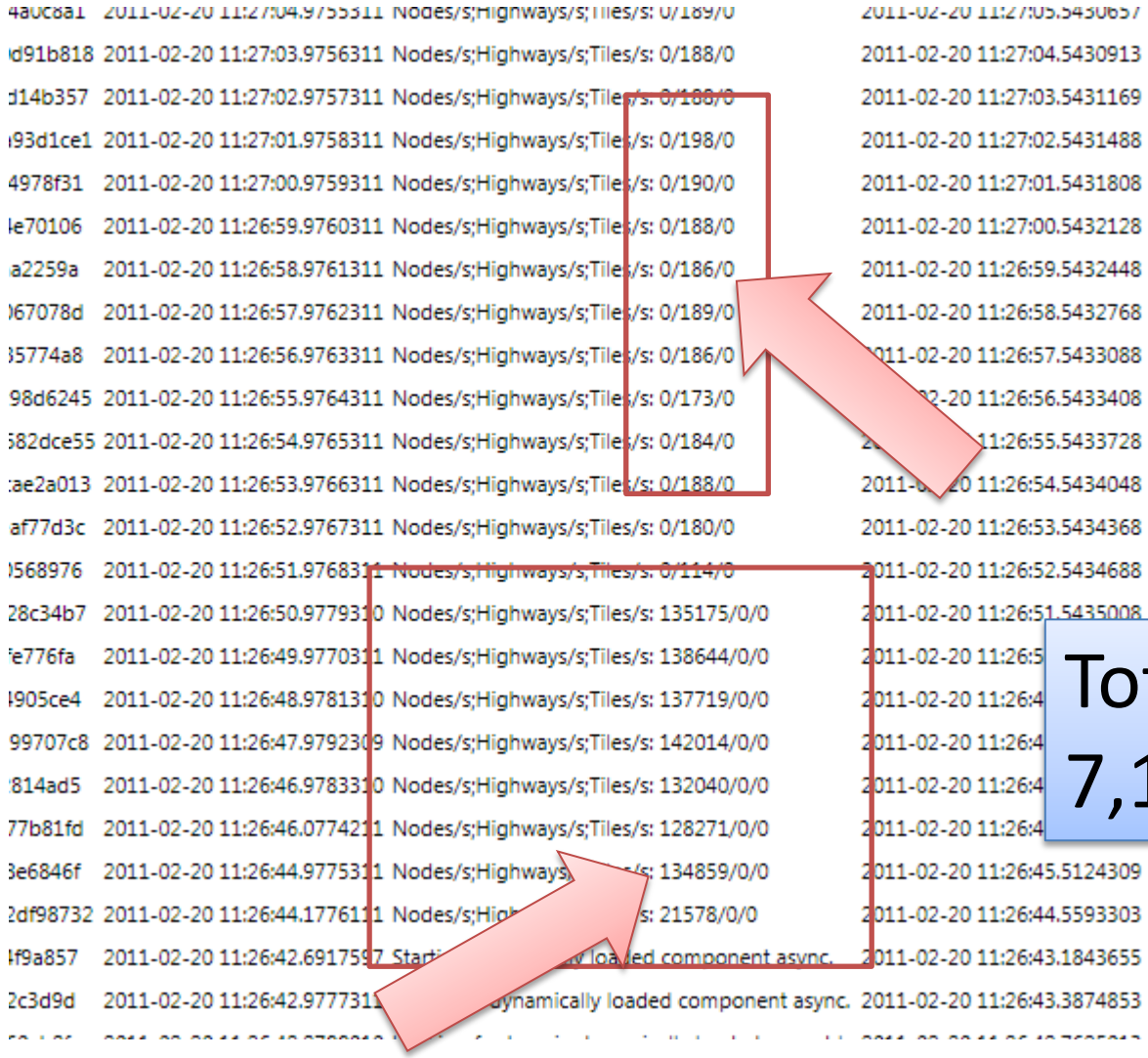
Solution 4: XmlReader

```
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)
```

Solution 4: XmlReader

```
}
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;
    }
}
```


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
CPU Usage
3 %
CPU Usage History
Memory
1.74 GB
Physical Memory Usage History
Physical Memory (MB)
Total 14366
Cached 927
Free 11813
Kernel Memory (MB)
Total 161
Paged 127
Nonpaged 33
System
Handles 13593
Threads 634
Processes 49
Up Time 0:59:20
Page File 1966M / 18456M
Resource Monitor...
Processes: 49
CPU Usage: 3%
Physical Memory: 12%

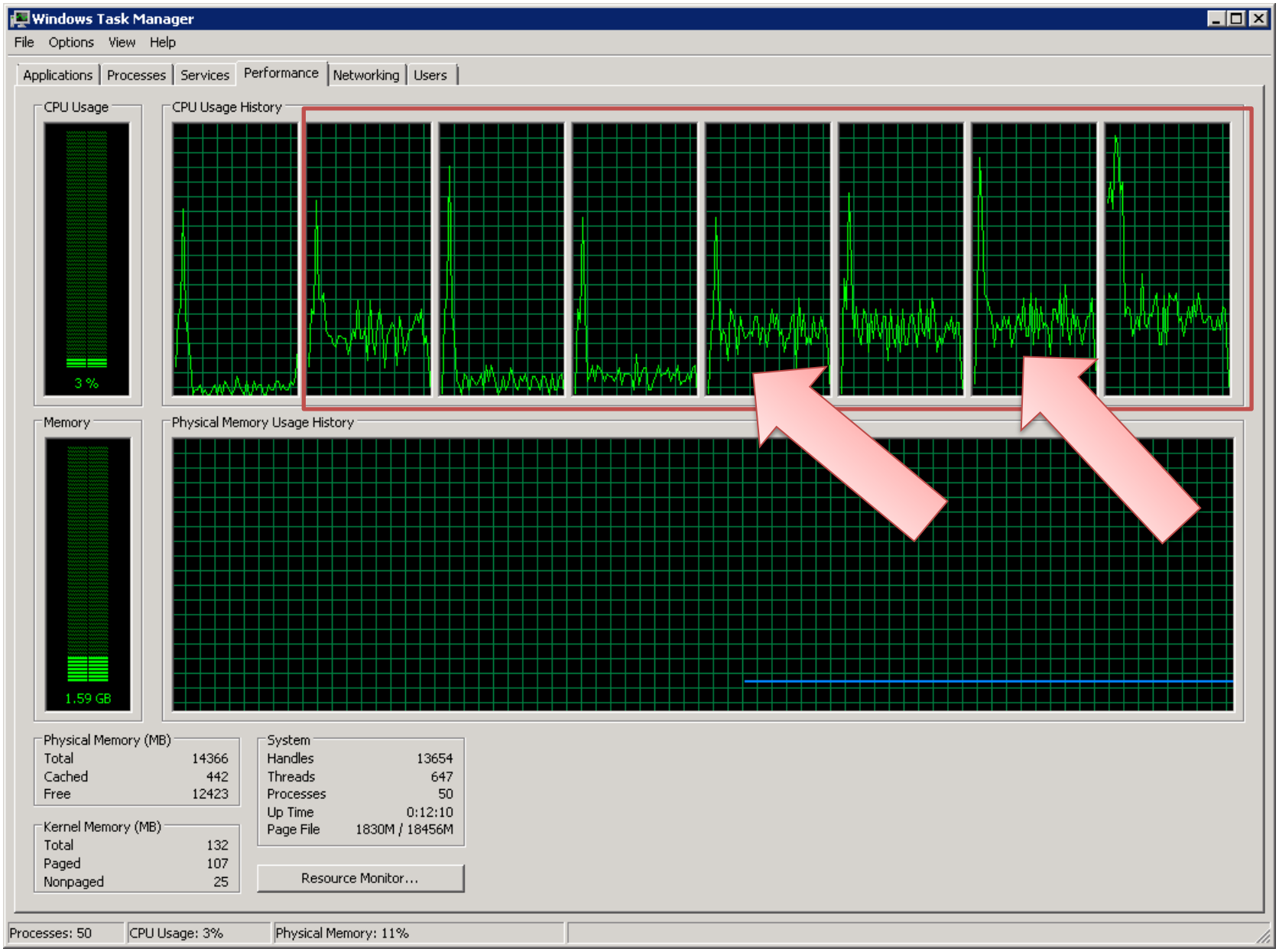


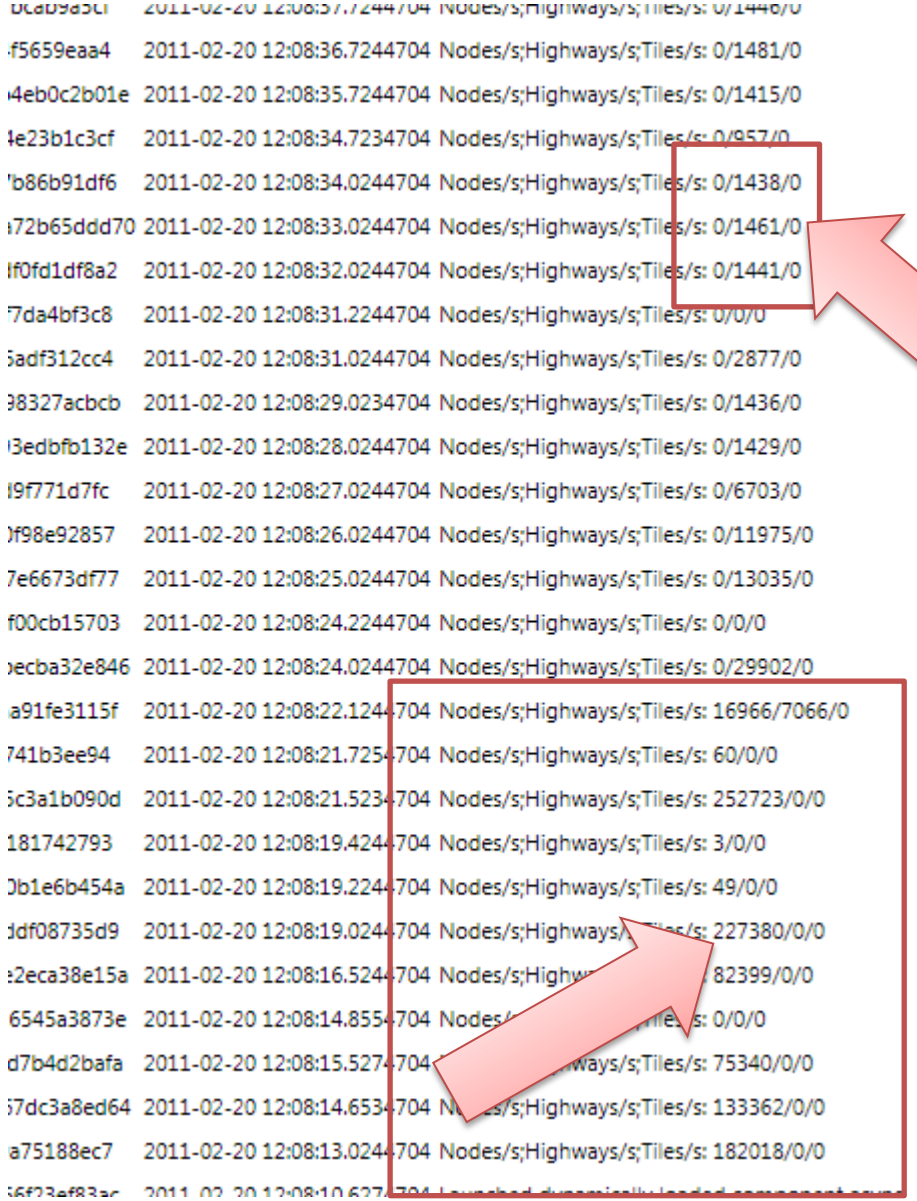
Total: 6,24 Min.
7,1% slower

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)
});
```


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
CPU Usage
3 %
CPU Usage History
Memory
1.59 GB
Physical Memory Usage History
Physical Memory (MB)
Total 14366
Cached 442
Free 12423
Kernel Memory (MB)
Total 132
Paged 107
Nonpaged 25
System
Handles 13654
Threads 647
Processes 50
Up Time 0:12:10
Page File 1830M / 18456M
Resource Monitor...
Processes: 50
CPU Usage: 3%
Physical Memory: 11%



Total: 0,98 Min.
83% faster

Dynamic, DLR

C# DYNAMIC FEATURES

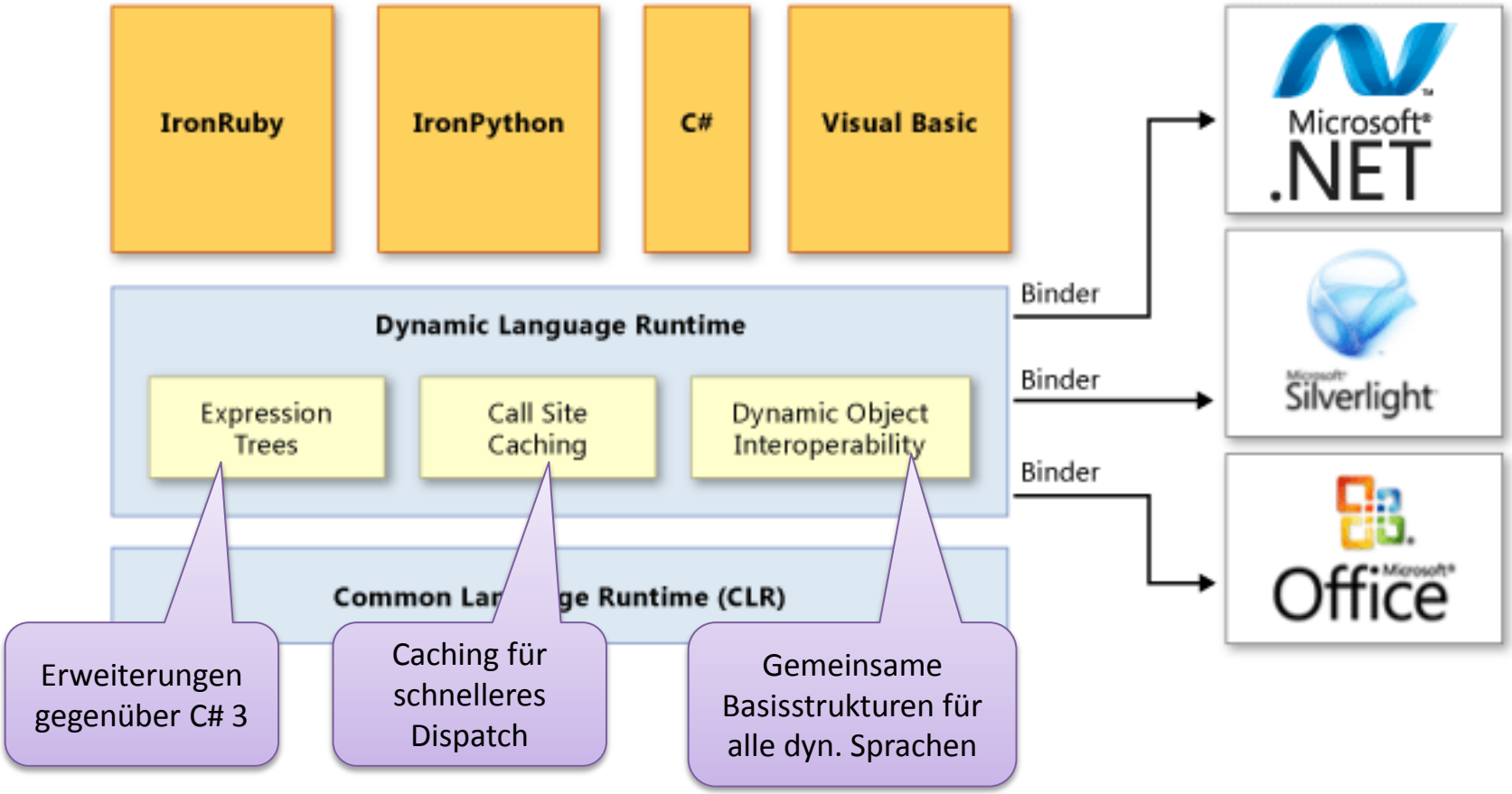
C# dynamic Keyword

- 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 wereden (z.B. `is`, `as`, `typeof`)

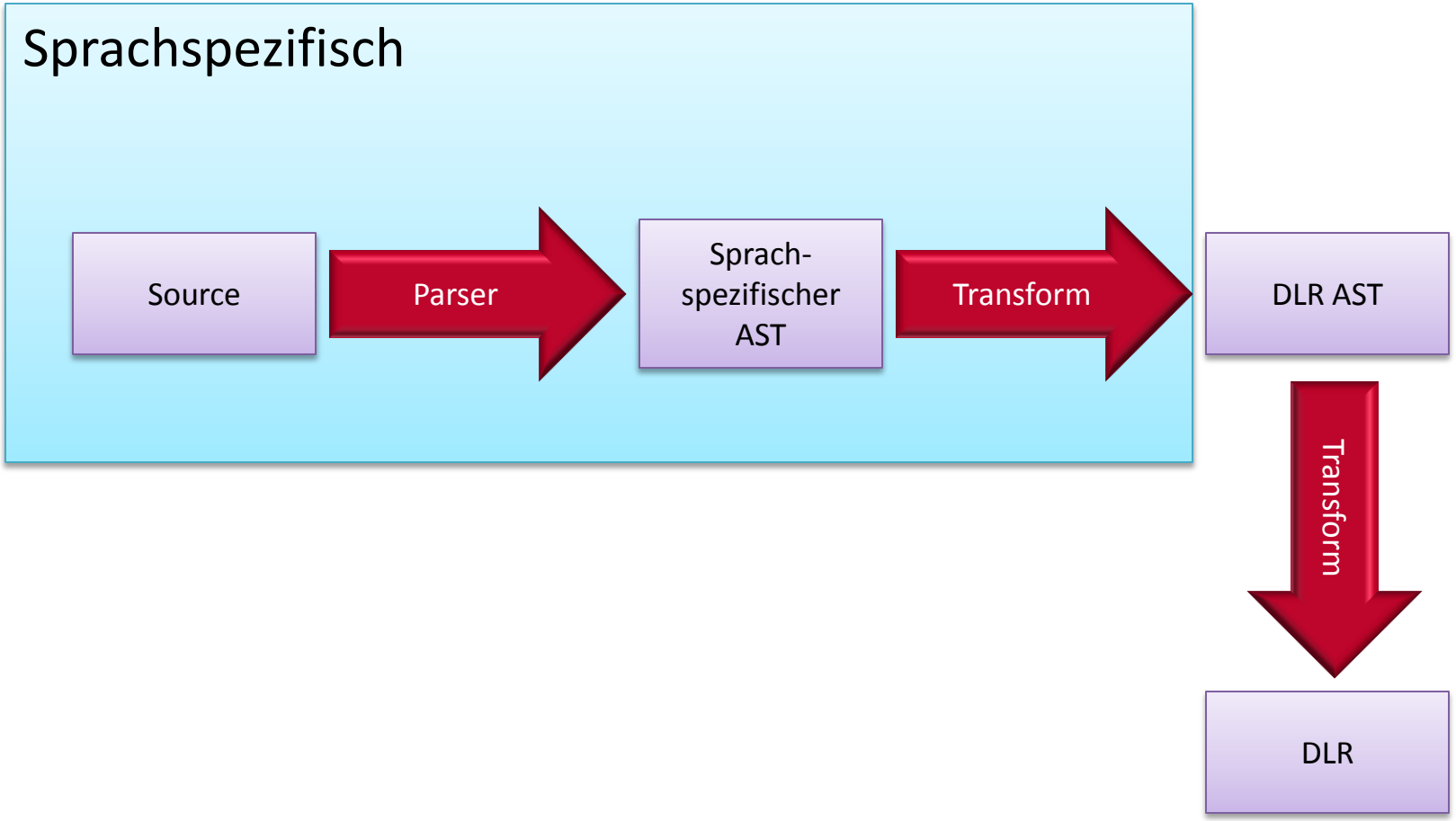
C# dynamic Keyword

- 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
Source
Parser
Sprach-
spezifischer
AST
Transform
DLR AST
Transform
DLR

· ExpressionTrees in C#

2010

Inheritance Hierarchy

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

2008

Inheritance Hierarchy

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 `INotifyPropertyChanged`

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()`

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

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!"
```

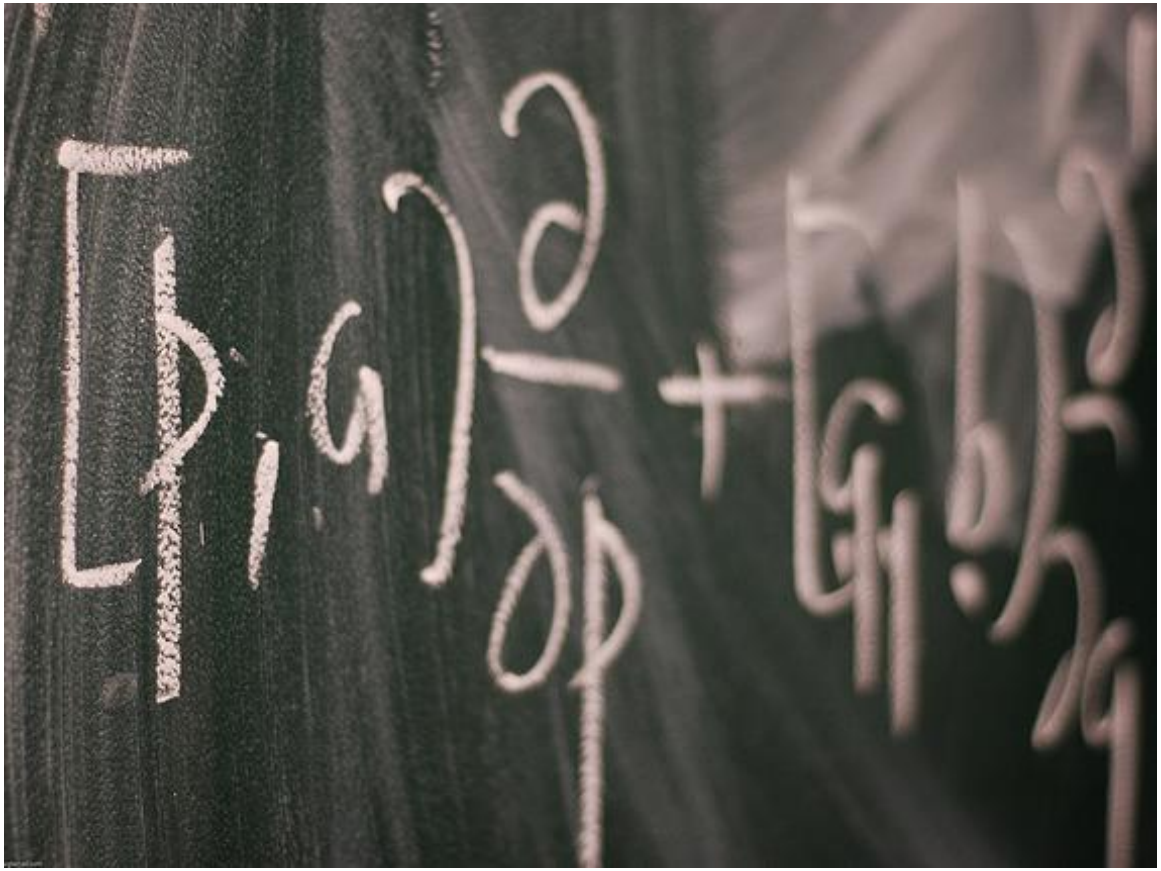
ViewModel geschrieben in Python (Python implementiert ICustomTypeDescriptor)

Zugriff auf Elemente der C# Anwendung

Dynamisches Laden des XAML-Codes

Auf WPF-Event in Python reagieren

BeginInvoke wegen Hintergrundthread



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);
    }
}
```

Aufruf der zuvor kompilierten Funktion

Dynamischer Aufruf über Reflection

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)

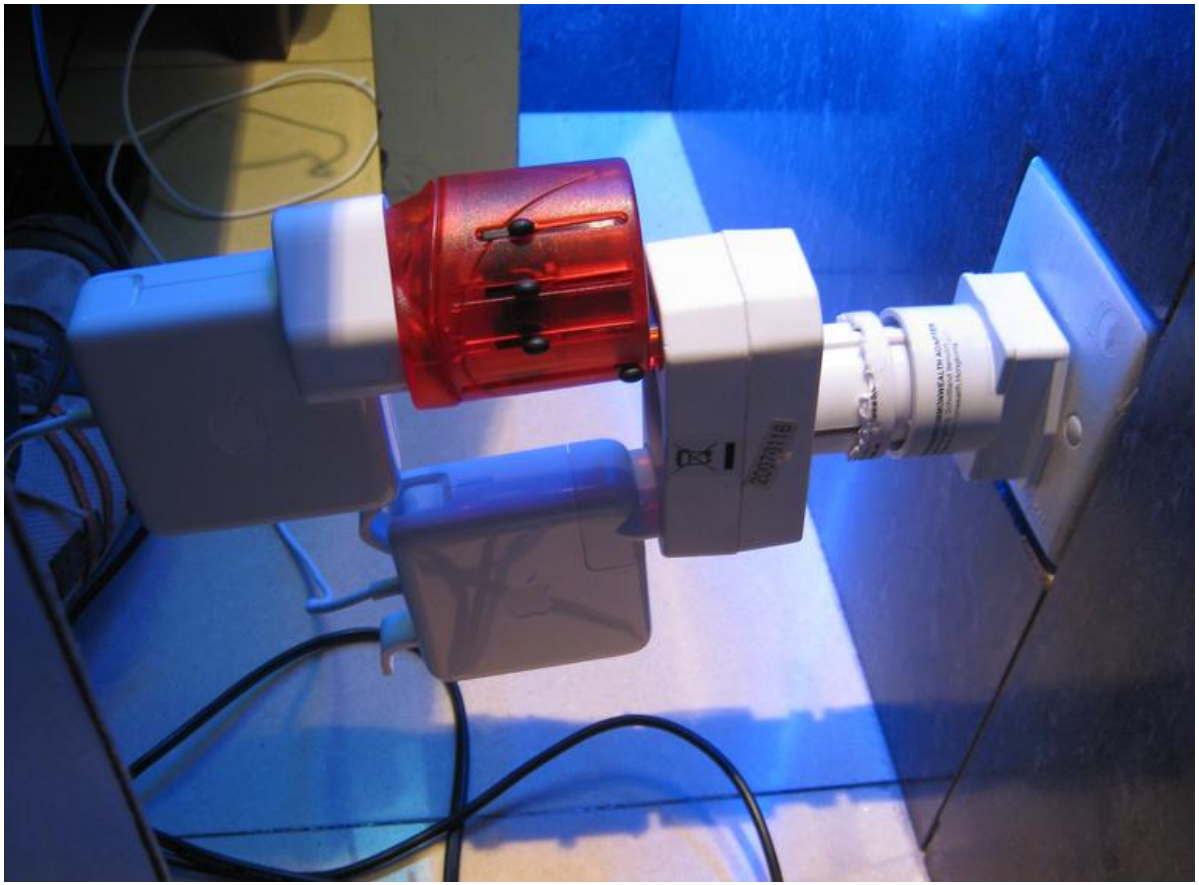
Advanced: LINQ in Python

```
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



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;
}
```

Converter

```
<DataGridTemplateColumn  Header="Number of approved Sessions">
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
            <Border Background="{Binding Path=NumberOfApprovedSessions,
    Converter={StaticResource ResourceKey=IronPythonExpressionConverter},
    ConverterParameter='&quot;Red&quot; if Value == 0 else &quot;Green&quot;'}">
                <TextBlock Text="{Binding Path=NumberOfApprovedSessions}" />
            </Border>
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
```

WPF

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);
```

Funktionsdefinition abfragen

Aufruf der Funktion mit `ObjectOperations`

Weitere Ressourcen

- IronPython Dokumentation
  - http://www.ironpython.net
  - http://docs.python.org
  - Sourcecode (DLR und IronPython sind auf codeplex)
- 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(typeof(Shape))]
public class Circle : Shape
{
  // Implementation
}

[Export]
public class Toolbox
{
  [ImportMany]
  public Shape[] Shapes { get; set; }
  // Additional implementation...
}
[…]
var catalog = new AssemblyCatalog(typeof(Square).Assembly);
var container = new CompositionContainer(catalog);
Toolbox toolbox = container.GetExportedValue<Toolbox>();
```

Export with name or type

„Attributed Programming Model“

Defaults to `typeof(Toobox)`

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; }
}

public class NumTwo : NumOne
{
}

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

public class NumFour : NumThree
{
}
```

Import automatically inherited

Export NOT inherited
→ NumTwo has no exports

Member-level exports are never inherited

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

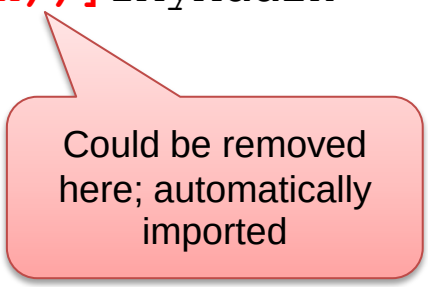
- Imported object is not instantiated immediately
  - Imported (only) when accessed
- Sample:

```
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;
}
```



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; }
  }
}
```

```
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();
    }
  }
}
```

Export Metadata can be mapped to metadata view interface

Metadata (continued)

```
namespace MetadataSample
{
   class Program
   {
      static void Main(string[] args)
      {
         var catalog = new AssemblyCatalog(
            typeof(ITranslator).Assembly);
         var container = new CompositionContainer(catalog);

         // We need a translator from hungarian to english
         Lazy<ITranslator, ITranslatorMetadata> translator =
            container
            .GetExports<ITranslator, ITranslatorMetadata>()
            .Where(t => t.Metadata.SourceLanguage == "hu-HU"
               && t.Metadata.TargetLanguage == "en-US")
            .FirstOrDefault();
      }
   }
}
```

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

VIELEN DANK FÜR IHRE MITARBEIT!