C# Fitness

Rainer Stropek | timecockpit

Your Host

Rainer Stropek

Developer, Entrepreneur
Azure MVP, MS Regional Director

Contact

software architects gmbh
rainer@timecockpit.com
Twitter: @rstropek

Agenda (German)

Es ist soweit! Nach längerer Durststrecke ist Roslyn fertig, und für C#-Entwickler gibt es jede Menge Neuigkeiten. Neue Sprachelemente, neue Entwicklungsumgebung, der Schwenk zu Open Source, Compiler-as-a-Service und vieles mehr – alles Gründe, die Neuerungen in einem BASTA!-Workshoptag durchzuarbeiten. Der Workshop richtet sich an C#-Entwickler, die in einem Tag die C#- und Visual-Studio-Verbesserungen konzentriert kennenlernen möchten. Konkret werden wir unter anderem folgende Themen behandeln:

Neuerungen in C# 6

Neuigkeiten in der Visual Studio IDE für C#-Entwickler

Anwendungsbeispiele für Compiler-as-a-Service (Roslyn) aus der Praxis

Must-have-NuGet-Pakete für C#-Entwickler (Base Class Library und Community)

Rainer Stropek wird in bewährter Weise anhand von Codebeispielen C# und Visual Studio vNext erklären. Die Beispiele stellt Rainer wie immer zu Beginn des Workshops zur Verfügung, damit Teilnehmer auf dem eigenen Laptop mitexperimentieren können. Ein eigener Computer ist aber keine Voraussetzung. Rainer wird wie immer die Codebeispiele auf der Bühne entwickeln und Schritt für Schritt erklären. Der Workshop setzt bei den Teilnehmern mindestens C#-V3-Kenntnisse voraus.

Timeline

Session 1 (9:00 – 10:30)

What's new in .NET and C# 6?
Part one of *ProductionPlanning* sample

Session 2 (11:00 – 12:30)

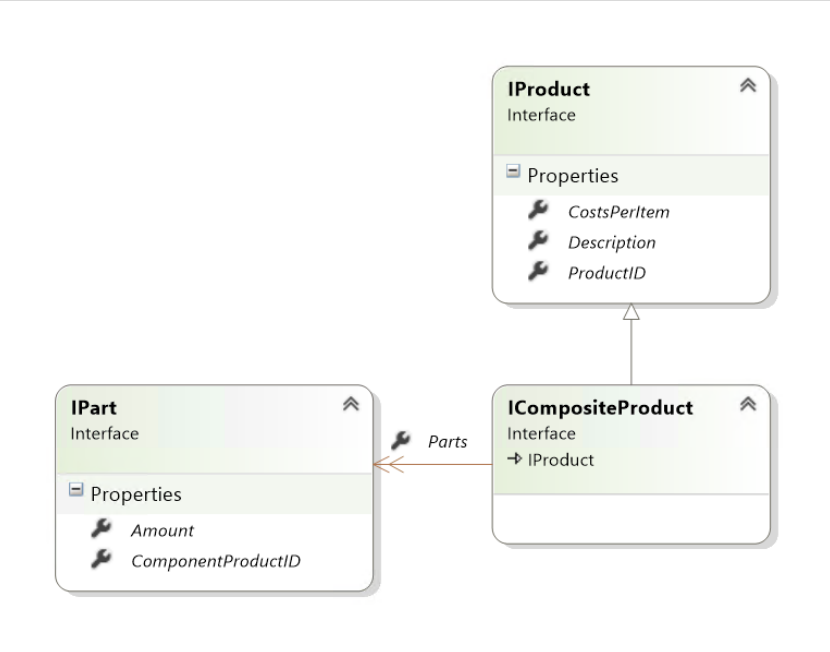
Part two of *ProductionPlanning* sample (focus on immutables)

Session 3 (13:30 – 15:00)

Roslyn

Session 4 (15:30 – 17:00)

Roslyn continued …
NuGet



Sample

Sample for the morning

Products with part lists

- Composite products have parts
- Products have associated costs

First Goal

- Implement a mutable class library acting as the VM for a UI
- `INotifyPropertyChanged`

Second Goal

- Implement an immutable class library for thread-safe calculations

Source code:
https://github.com/rstropek/Samples/tree/master/ProductionPlanning

What's new/coming?

C# 6, .NET 4.6, .NET Core, VS2015, …

What's Coming?

Roslyn

Re-written C# and VB Compilers

.NET 4.6 (aka 4.5.3)

Next release of the full .NET framework

.NET Core (aka .NET 5)

Deployed with your app (NuGet), open source, cross-platform
Some limitations currently apply (only ASP.NET 5 and command line, only Windows)

C# 6 News

Changes

Single expression function bodies, just like lambdas
`nameof` to get the name of e.g. a parameter as a string
Auto-properties can have initializers and no longer require setters
Index initializers inside object initializers.
Exception filters
Null-conditional operators
Using clauses for static classes
Await in catch and finally blocks
String Interpolation

Read more in my blog

Lambda Function Bodies

```
private class CostCalculator :
  VisitingProductCalculator<decimal>
{
  // Note function-bodied syntax here

  public override decimal AggregateInterimResults(
    decimal a, decimal b) => a + b;

  public override decimal VisitProduct(
    ImmutableProduct product) => product.CostsPerItem;

  public override decimal VisitPart(ImmutablePart part) =>
    // Price of part * number of parts
    base.VisitPart(part) * part.Amount;
}
```

```
public class ImmutablePart : IPart
{
  […]

  public ImmutableProduct Part { get; }

  // Note implicit interface implementation with
  // function-bodied property
  Guid IPart.ComponentProductID => this.Part.ProductID;

  public int Amount { get; }
}
```

Lambda Function Props

Getter-only properties with lambdas

```
public ImmutablePart(Guid productID, int amount,
  ProductRepository productRepository)
{
  #region Check preconditions
  if (productRepository == null)
  {
    throw new ArgumentNullException(nameof(productRepository));
  }
  #endregion

  […]
}
```

nameof

New operator

```
public class ImmutableProductRepository
{
  #region Constructors
  // Note that we can write to read-only properties
  // inside of this constructor.

  private ImmutableProductRepository()
  {
    this.Products = ImmutableList<ImmutableProduct>.Empty;
  }
  #endregion

  // Note read-only static property with initialization
  public static ImmutableProductRepository Empty { get; }
    = new ImmutableProductRepository();

  […]
}
```

Auto-props

Now with initializers

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void TestMissingProductID()
{
  Guid productID = Guid.NewGuid();
  try
  {
    new ImmutablePart(productID, 5,
      new List<ImmutableProduct>());
  }
  // Note new exception handling condition here
  catch (ArgumentException ex)
    if (!ex.Message.Contains(productID.ToString()))
  {
    // Suppress exception if message is not correct
  }
}
```

Exception Filters

Null Conditional

```
public ImmutableProductRepository Add(IProduct product)
{
  // Create new immutable (composite) product
  var compositeProduct = product as ICompositeProduct;
  var immutableProduct = (compositeProduct != null &&
    compositeProduct?.Parts?.Count() > 0)
    ? new ImmutableCompositeProduct(
      compositeProduct, this.Products)
    : new ImmutableProduct(product);

  // Return a new repository
  return new ImmutableProductRepository(
    this.Products.Add(immutableProduct));
}


// Note null-conditional call inside expression-bodied function
protected void RaisePropertyChanged(
  [CallerMemberName]string propertyName = null) =>
  PropertyChanged?.Invoke(this,
    new PropertyChangedEventArgs(propertyName));
```

```
this.Part = productRepository.SingleOrDefault(
  p => p.ProductID == productID);
if (this.Part == null)
{
  // Note string interpolation here
  throw new ArgumentException(
    $"Could not find product with ID {productID}",
      nameof(productID));
}
```

String Interpolation

Immutables

Data Structure for a Multi-Threaded World

Introduction

An immutable object is an object whose state cannot be modified after it is created (Wikipedia)

Hold a reference to it, enumerate it, search it, etc., from any thread at any time, and be confident that you'll get the expected behavior

Important attributes of immutables

Inherently thread-safe (if deep immutable)
Can be copied by copying a pointer to the immutable object

Types of immutability

`readonly` fields
Freezables (aka *popsicle* immutability)
Shallow vs. deep immutability
Immutable Facade (e.g. `ReadOnlyCollection<T>`) over *mutable* collections

System.Collections.Immutable

Immutable collection types

```
ImmutableArray<T>
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableSortedDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableSortedSet<T>
ImmutableStack<T>
ImmutableQueue<T>
```

Implement `IReadOnly*`

Easy to migrate code

Interfaces

```
IImmutableDictionary<TKey, TValue>
IImmutableList<T>
IImmutableQueue<T>
IImmutableSet<T>
IImmutableStack<T>
```

`ImmutableInterlocked`

Helper methods for atomic operations

When Immutable Collections?

Snapshot semantics

Share your collections in a way that the receiver can count on never changing

Implicit thread-safety in multi-threaded applications

No locks required to access collections

Functional programming friendly

Allow modification of a collection during enumeration

Original collection does not change

```
var fruitBasket = ImmutableList<string>.Empty;
fruitBasket = fruitBasket.Add("Apple");
fruitBasket = fruitBasket.Add("Banana");
fruitBasket = fruitBasket.Add("Celery");

fruitBasket = fruitBasket.AddRange(
  new [] { "Kiwi", "Lemons", "Grapes" });

var hugeList = ImmutableList<int>.Empty.AddRange(
  Enumerable.Range(1, 10000));
// Note that the following line does not duplicate
// memory usage by copying hugeList. Object tree is reused!
var hugeListPlus1 = hugeList.Add(10001);
```

Using Immutables

Static `Empty` property

Changes return new collection
- Object tree is reused

Use Builder if you have to perform many mutations
- String → StringBuilder
- ImmutableList<T> → ImmutableList<T>.ToBuilder()

Read more

Immutable Collection in MSDN

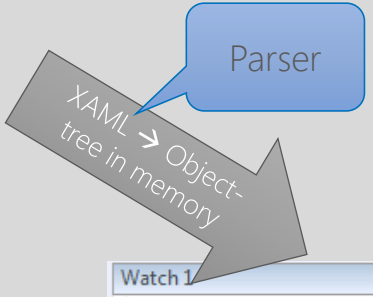
Blog article about launch of immutable collections
Including video

Roslyn

Compiler-as-a-Service

From Text to Tree

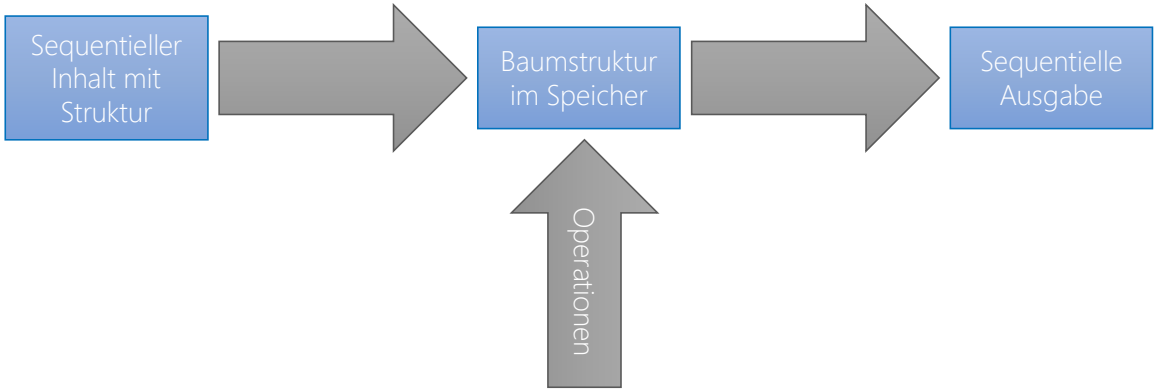
```
<Garden […]>
  <Garden.Trees>
    <Tree>
      <Tree.Fruit>
        <Apple />
      </Tree.Fruit>
    </Tree>
    <Tree>
      <Tree.Fruit>
        <Apple />
      </Tree.Fruit>
    </Tree>
    <Tree>
      <Tree.Fruit>
        <Apricot />
      </Tree.Fruit>
    </Tree>
  </Garden.Trees>
</Garden>
```



Watch 1

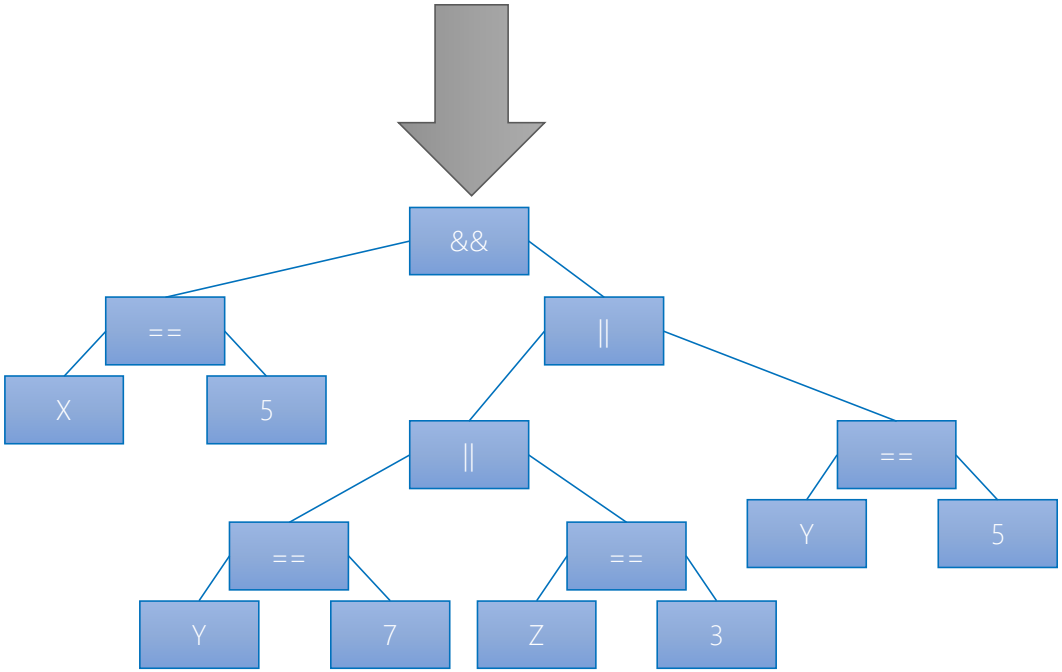
Name	Value
myGarden	{TreeNursery.Xaml.Garden}
Trees	Count = 3
[0]	{TreeNursery.Xaml.Tree}
Fruit	{Apple}
[TreeNursery.Xaml.Apple]	{Apple}
[1]	{TreeNursery.Xaml.Tree}
Fruit	{Apple}
[TreeNursery.Xaml.Apple]	{Apple}
[2]	{TreeNursery.Xaml.Tree}
Fruit	{Apricot}
[TreeNursery.Xaml.Apricot]	{Apricot}
Raw View	

Von Text zum Baum



Wo ist der Baum?

```
X=5 And ( Y=7 Or Z=3 Or Y=5 )
```



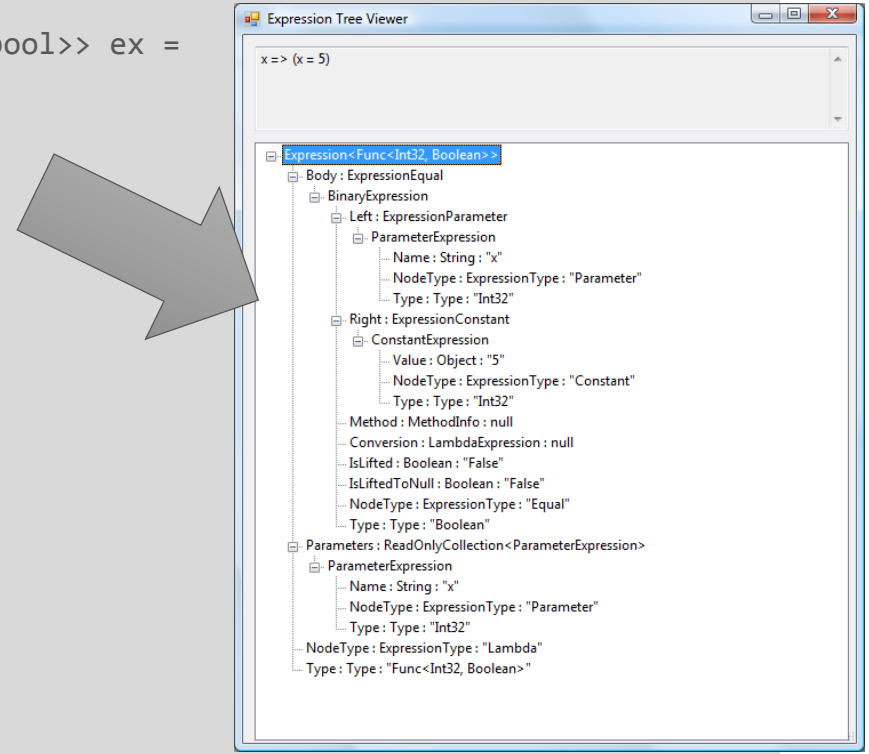
Expression Trees in C#

```
0 references
static void Main(string[] args)
{
    Func<int, bool> f =
        (x) => x == 5;

    Expression<Func<int, bool>> ex =
        (x) => x == 5;
}

// The following code is generated behind the scen
0 references
private static void Main2(string[] args)
{
    ParameterExpression expression2 = null;
    Func<int, bool> func = new Func<int, bool>([... anonymous method ...]);
    ParameterExpression[] parameters = new ParameterExpression[] { expression2 };
    Expression<Func<int, bool>> expression = Expression.Lambda<Func<int, bool>>(
        Expression.Equal(
            expression2 = Expression.Parameter(typeof(int), "x"),
            Expression.Constant(5, typeof(int))),
        parameters);
}
```

Compiler lets you access syntax tree during runtime

Expression Trees in C#

```
Func<int, bool> f =
    (x) => x==5;

Expression<Func<int, bool>> ex =
    (x) => x == 5;
```



Expression Trees in C#

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

2012

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

2008

Roslyn

Re-write of C# and VB compiler in C# and VB

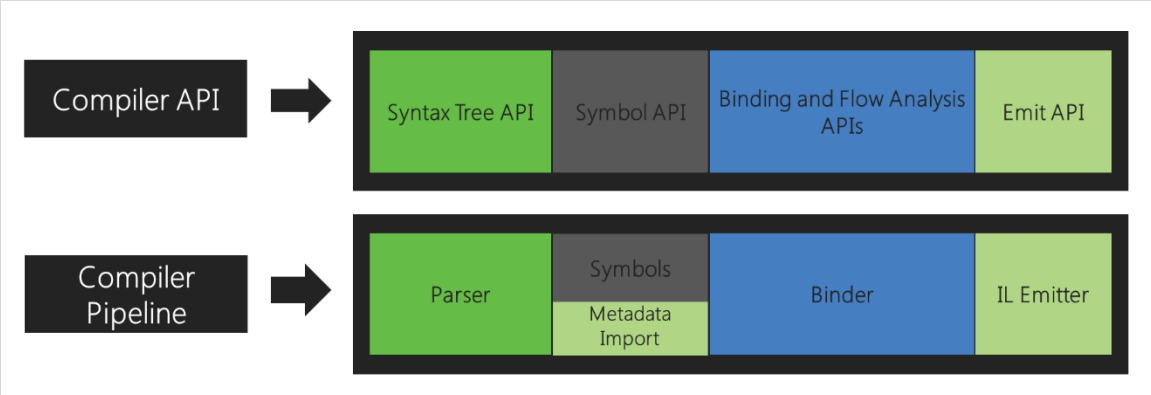
Multi-year preview, launch will be in VS2015
Open source, cross platform

Opening the compiler platform

Compiler-as-a-Service
Write applications that can understand C# and VB source code
Revolutionizes writing tools that act on source code-level

NuGet Roslyn Compiler Package

Microsoft.CodeAnalysis
Language-specific packages for C# and VB



Roslyn

Compiler API Structure

Syntax Tree API/Parser

Lexer (source is tokenized)
Syntax parser based on grammar
→ Syntax Tree (e.g. code formatting)

Symbols

Symbols analyzed from source
→ Hierarchical Symbols Table
(e.g. Object Browser)

Binding

Identifiers are matched to symbols

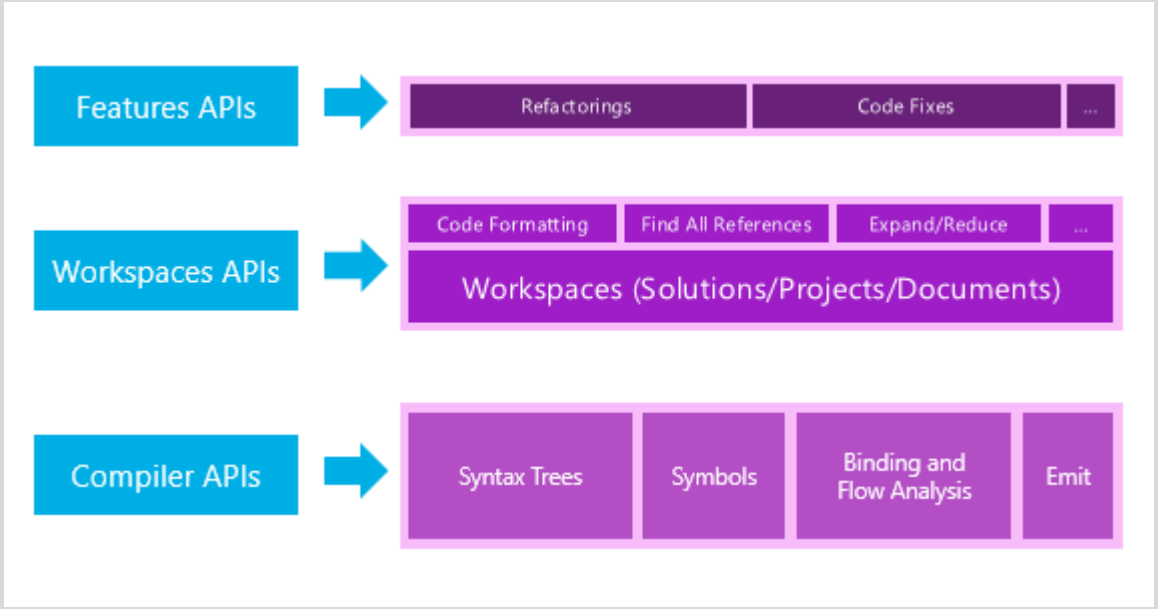
Emit

Assembly generation
(e.g. Edit-and-Continue)

Image source:
https://github.com/dotnet/roslyn/wiki/Roslyn%20Overview#introduction

Roslyn

Workspace API



- Object model for project and solution structure
- No dependencies on VS
- Not in the scope of this workshop

Image source:
https://github.com/dotnet/roslyn/wiki/Roslyn%20Overview#api-layers

Roslyn Syntax Tree

Process the syntactic structure of source code

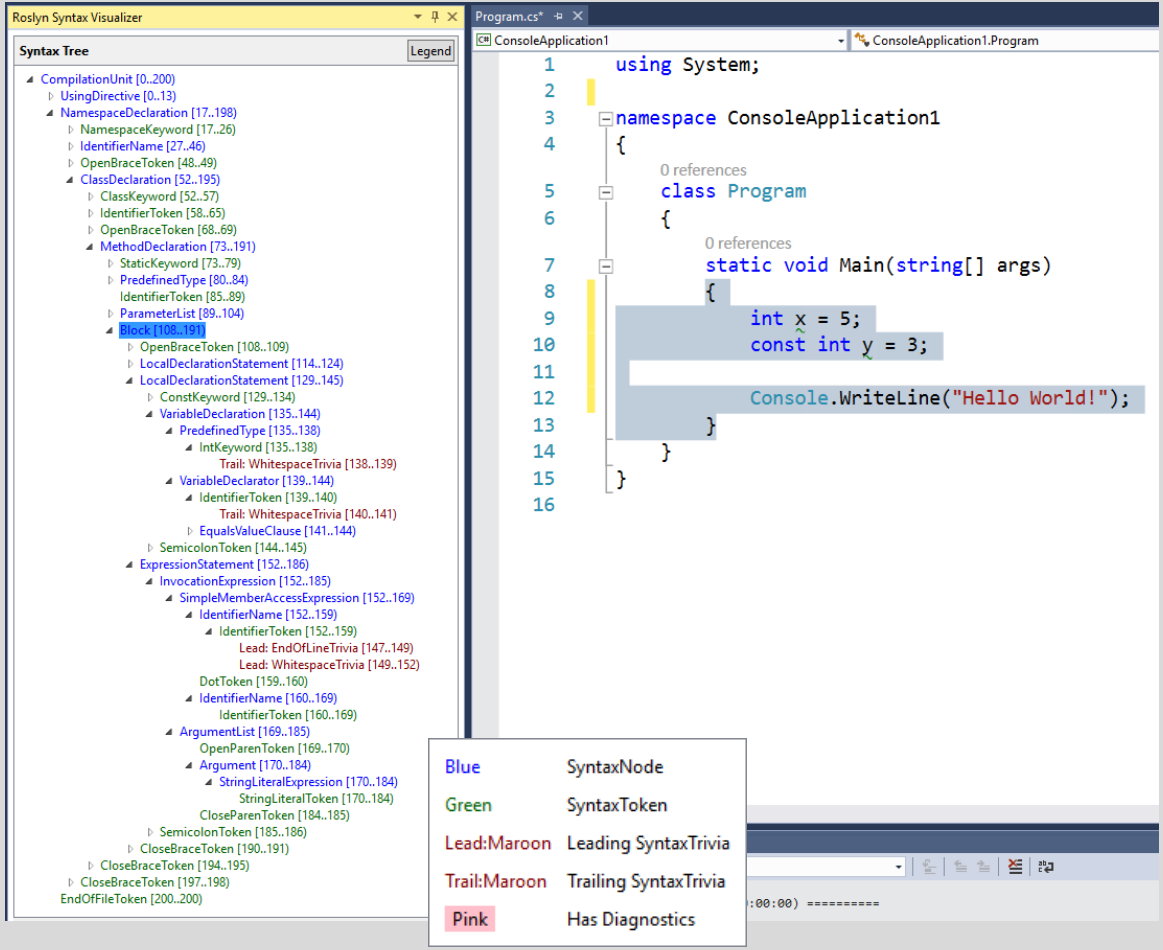
Syntax tree even contains errors and e.g. information about skipped tokens

Create, modify, and rearrange source code

No need to edit text directly
Code is changed by creating and manipulating trees
Completely round-tripable

Immutable and thread-safe

Each change to the tree is done by creating a new snapshot
Tree nodes are reused in the background



Syntax Tree

Tip: Syntax Visualizer

Syntax Nodes

Syntactic constructs

E.g. declarations, statements, clauses, and expressions

Derived from `SyntaxNode` class

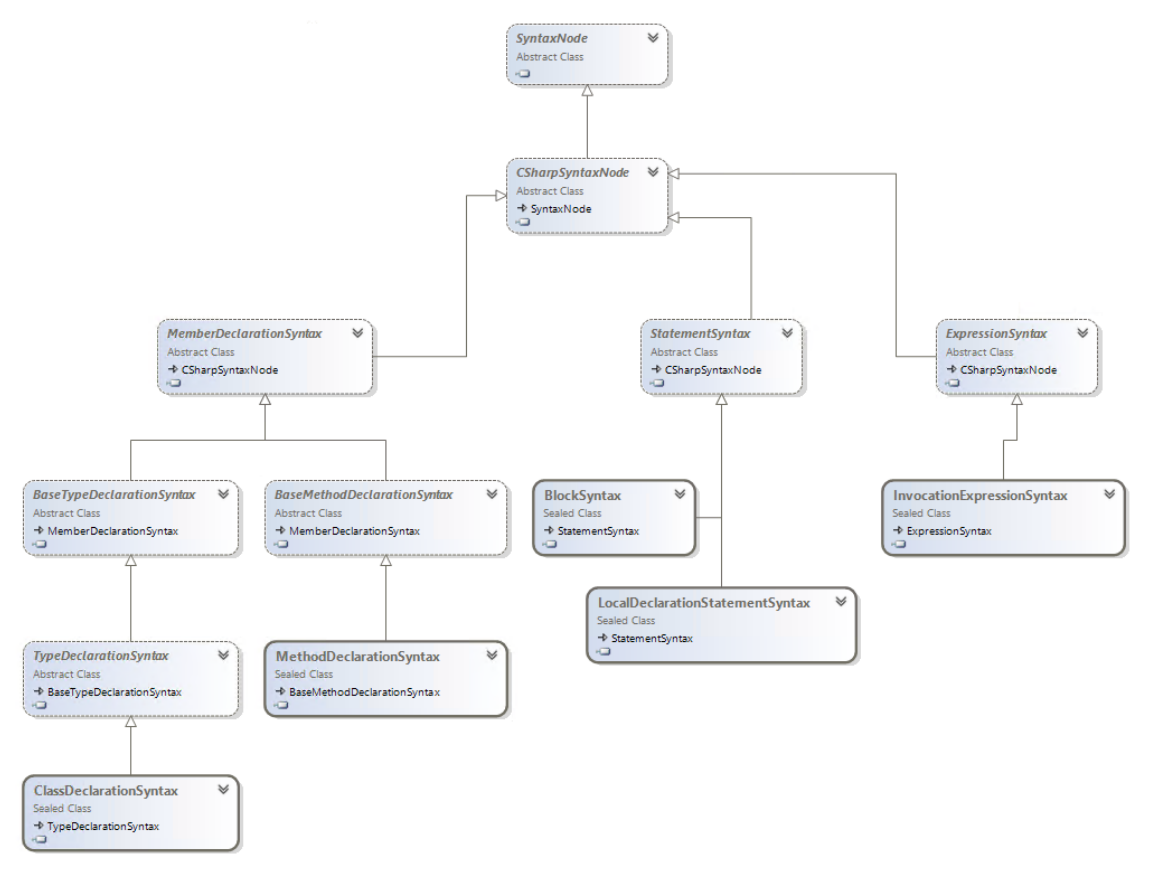
Class names: ...Syntax (e.g. `InvocationExpressionSyntax`)

Non-terminal nodes

All syntax nodes have children (`ChildNodes` property, `Parent` property)

Syntax Nodes

Excerpt from syntax node class diagram



Syntax Tokens

Terminal nodes

Smallest syntactic fragment of the code
E.g. keywords, identifiers, literals

`SyntaxToken` struct

Because of performance reasons a value type

Syntax Trivia

Insignificant for understanding the code

E.g. whitespace, comments, preprocessor directives

`SyntaxTrivia` struct

Because of performance reasons a value type

Access from token

Via `LeadingTrivia`/`TrailingTrivia` collections

Not part of the syntax tree → no parent

Find associated token using `Token` property

Spans

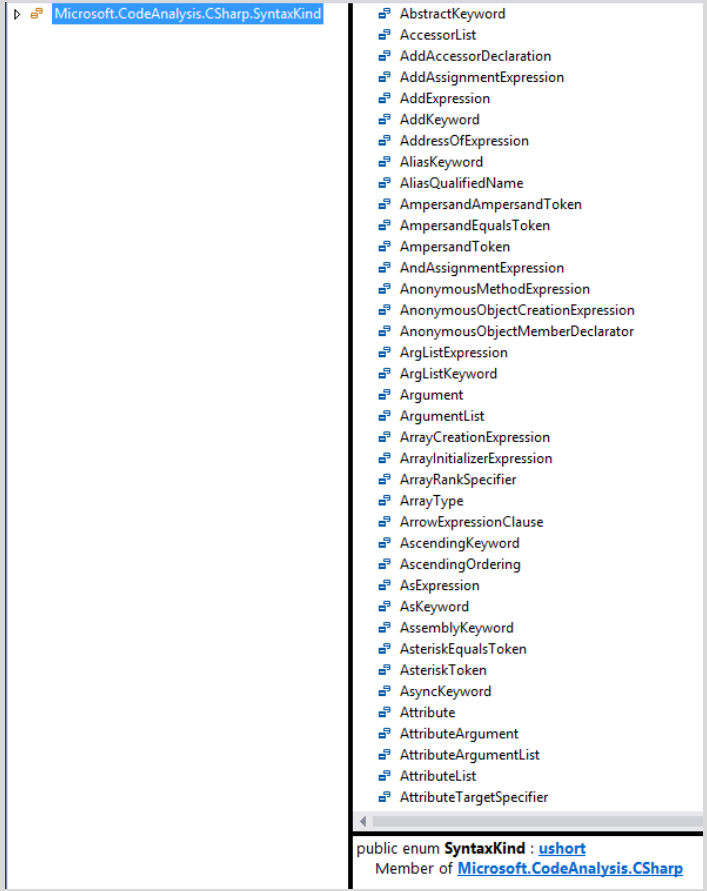
Represents position of node/token/trivia

Zero-based Unicode character index + char count (`TextSpan`)

Char count can be zero to indicate location between two characters

`Span` vs. `FullSpan`

Span is without surrounding trivia, FullSpan with



Kinds

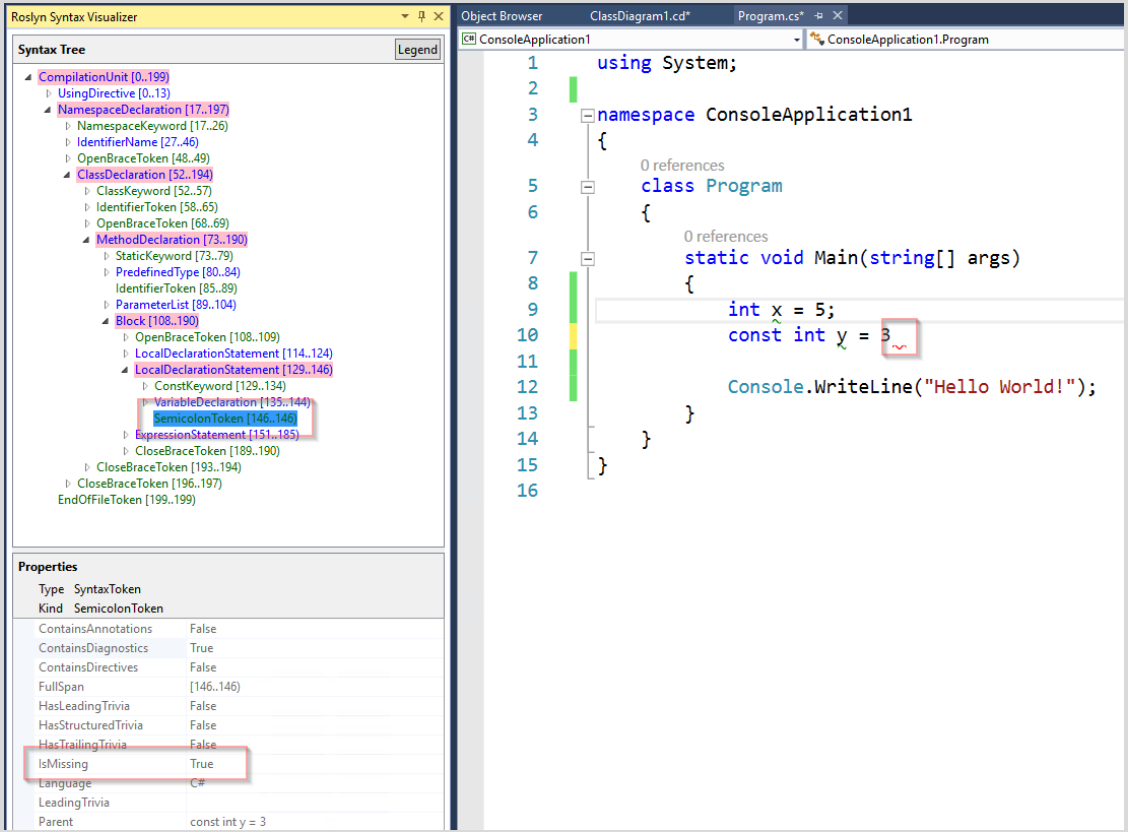
`RawKind` property (Int32)

Enum `SyntaxKind`

- Language-specific
- Use `CSharpSyntaxKind()` extension method for conversion

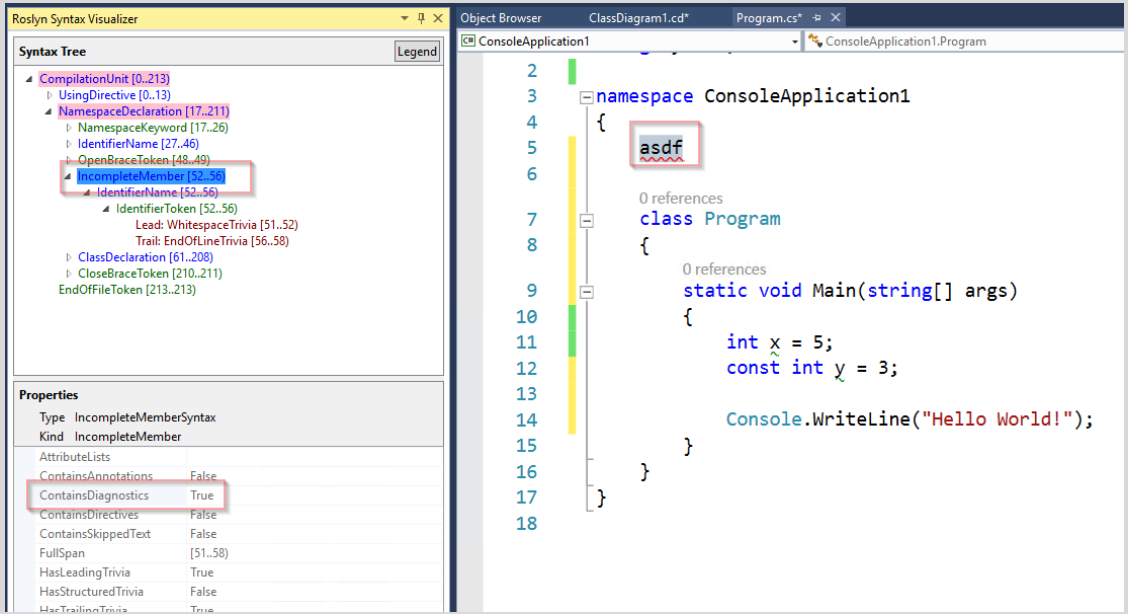
Errors

Missing tokens



Errors

Incomplete Member Example



Semantics

Syntax tree is not enough

E.g. variables with identical name, `var` declarations, etc.

Semantic model offers a solution

Compilation
Symbols
SemanticModel

Compilation, Semantic Model

Compilation = everything needed to compile a C# program

E.g. assembly references, compiler options, and source files

Immutable

Compilation can be based on an existing one (e.g. additional source file)

Semantic model

Semantic information for a single source file

Symbols

Sources

Declared by the source code
Imported from an assembly as metadata
Namespace, type, method, property, field, event, parameter, local variable

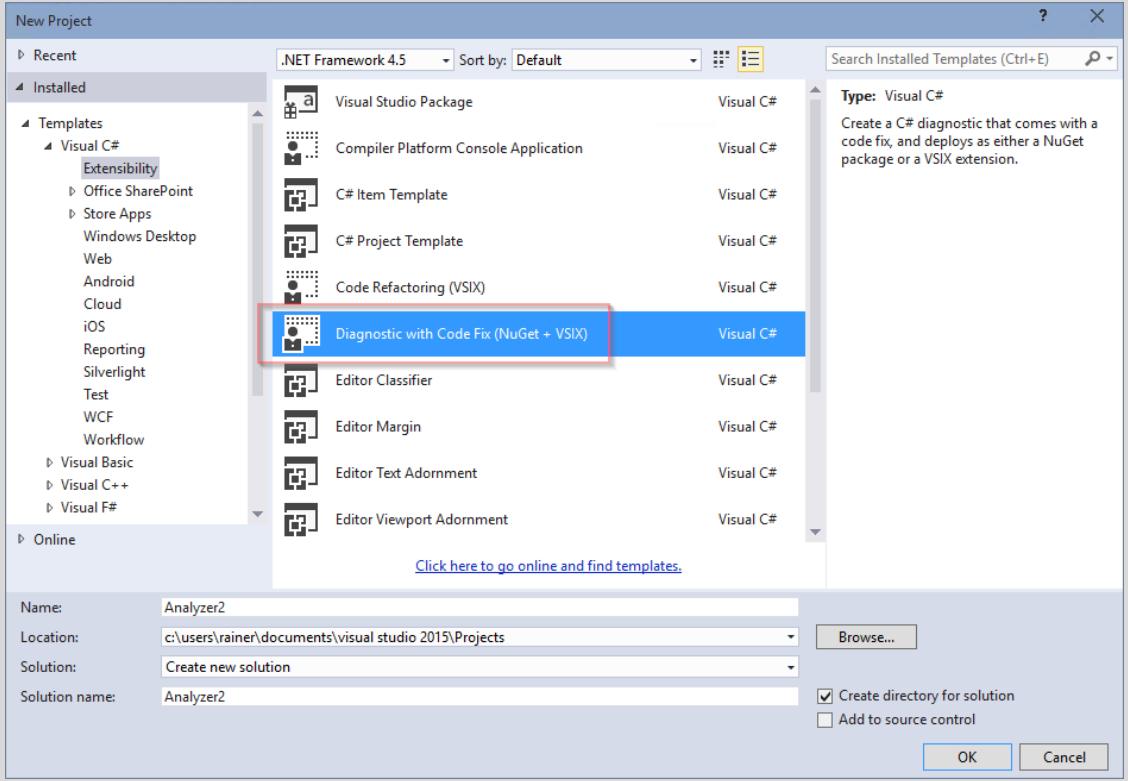
Derived from `ISymbol`

E.g. `IMethodSymbol`, `ILocalSymbol`

Demo

Roslyn Demos

See GitHub Samples Repo



Advanced Topics

Diagnostics and fixes

Download .NET Compiler Platform SDK Templates for CTP5

NuGet

Package Management for Visual Studio

What's NuGet?

Tool for deploying packages (=libraries and tools)

Everything you need in a single package

Binaries for different platforms
Optional symbols and sources
Script for project customizations (e.g. change references, change app/web.config, etc.)

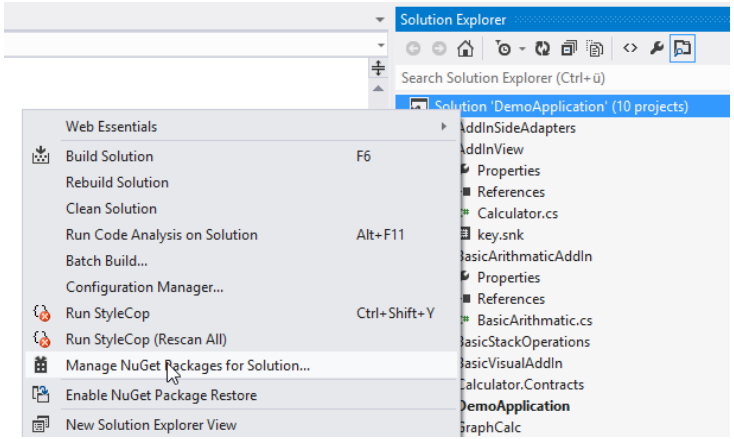
UI-Integration in Visual Studio

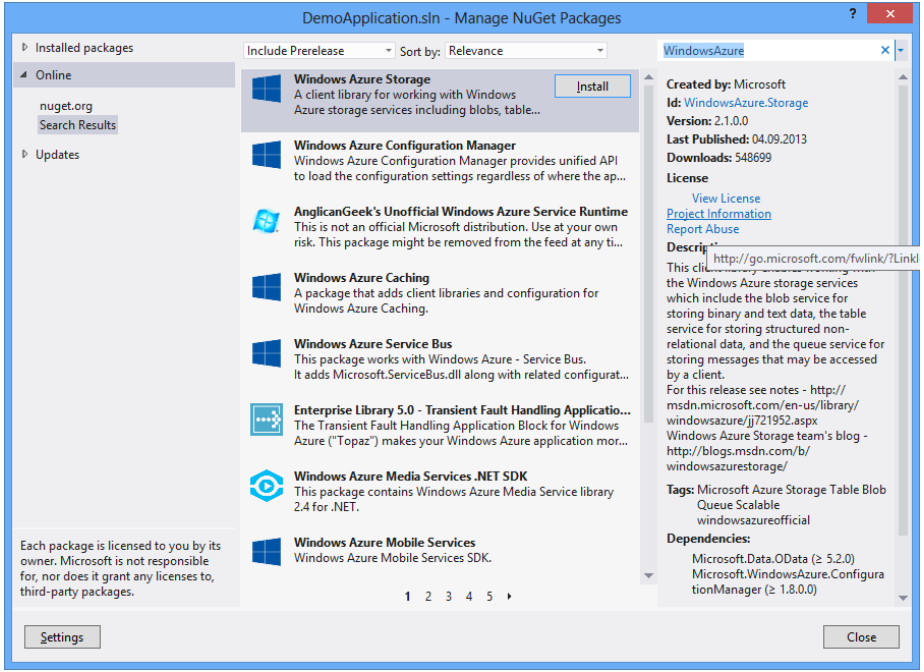
Started in VS2010
Greatly enhanced in VS2015

http://www.nuget.org

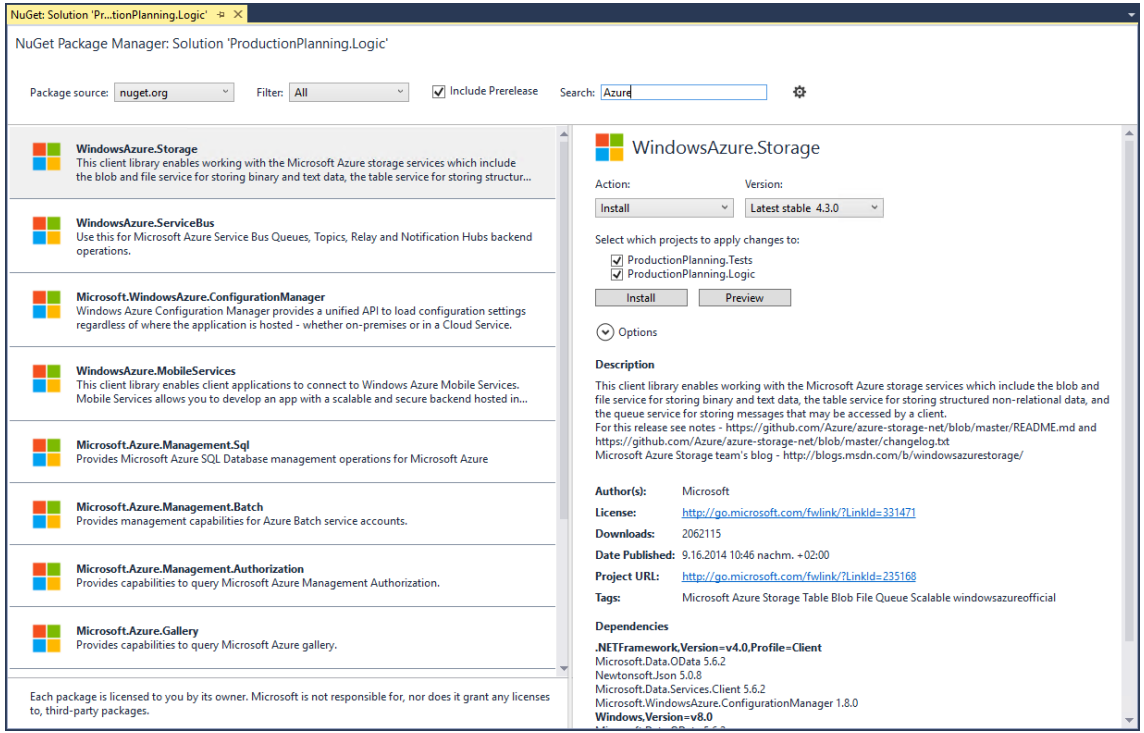
http://nuget.codeplex.com/ (Sourcecode)

NuGet in Visual Studio 2013





NuGet in Visual Studio 2015



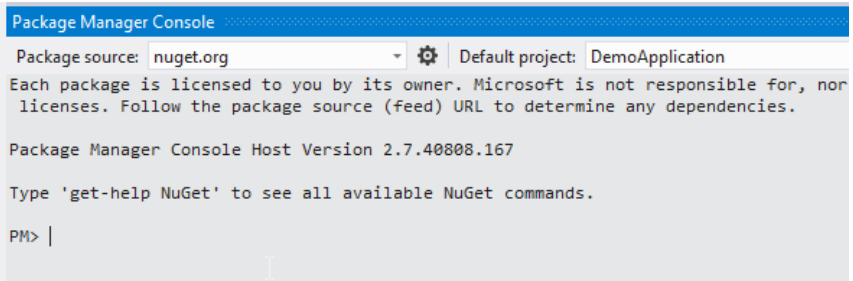
NuGet in Visual Studio

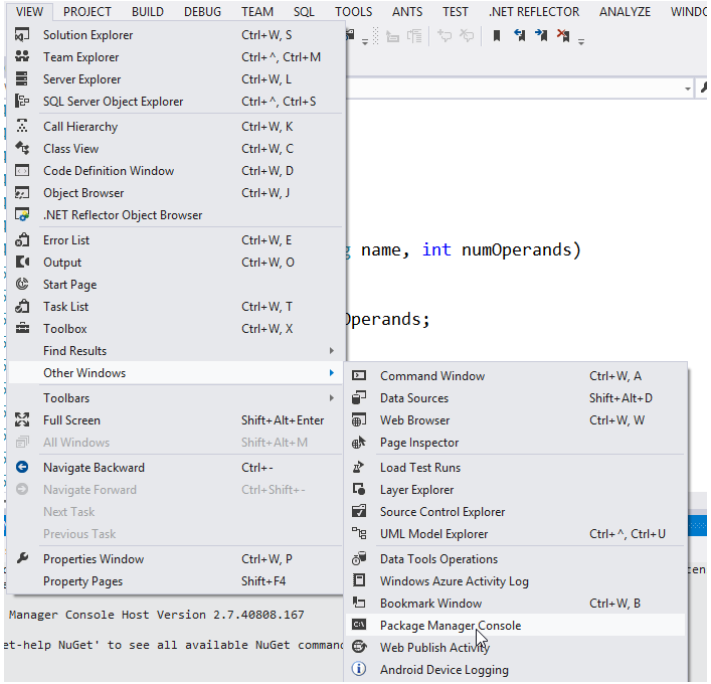
Package Manager Console

PowerShell console in Visual Studio

Automate Visual Studio and NuGet

[NuGet PowerShell Reference]





Role of NuGet in .NET

NuGet is used to ship parts of the .NET framework

See list of MS-supported packages: Microsoft .NET Framework NuGet Packages
Microsoft also supports selected open source packages (Json.NET, jQuery)

Enhanced *Manage NuGet Packages* dialog in VS2015

Use NuGet to distribute your own libraries

Between teams and developers
Between you and your customers/partners
Tip: Consider private NuGet feeds using services like myget

NuGet vs. Bower vs. NPM

NuGet for .NET development

Full Clients and server-side web development

NPM = package manager for Node.js

Installer for node-based tools like Grunt, Bower, etc.

Bower = front-end package management

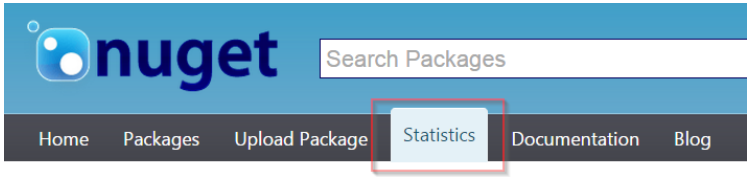
For web development, especially SPAs

Microsoft added support for Bower in VS2015

Announcement

Top .NET NuGet Packages

Json.NET
Newtonsoft.Json



Entity Framework
EntityFramework

Web development with Microsoft ASP.NET
MVC, WebAPI, SignalR, OWIN, and related tools (e.g. jQuery, WebGrease)

Microsoft HTTP Client Libraries (details see next slide)
Microsoft.Net.Http

```
using (var client = new HttpClient())
{
  try
  {
    var response =
      await client.GetAsync("http://www.contoso.com/");
    response.EnsureSuccessStatusCode();
    string responseBody =
      await response.Content.ReadAsStringAsync();

    // Could be replaced with helper method:
    // string responseBody =
    //   await client.GetStringAsync(uri);

    Console.WriteLine(responseBody);
  }
  catch(HttpRequestException e)
  {
    Console.WriteLine("\nException Caught!");
    Console.WriteLine("Message :{0} ",e.Message);
  }
}
```

HTTP Client Libraries

Sample

Easy way to consume REST services in .NET

Common programming model
for full .NET, Store apps, and PCL

Documentation incl. sample see MSDN

Top .NET NuGet Packages

OData

Microsoft.Data.Odata

Tip: Related packages Simple.OData.V4.Client, RESTier (details), Spatial

ANTLR

Antlr

Caution: NuGet package not up to date; see http://www.antlr.org/ for current version

Moq

Moq

Tip: Consider MS Fakes if you want to stick to Microsoft tools

BCL NuGet Packages (1/2)

Microsoft Base Class Library (BCL)

NuGet packages for delivering BCL features out-of-band
Supported by Microsoft

.NET Compatibility

BCL Portability Pack (Microsoft.Bcl)
async/await for VS2012/.NET 4 (Microsoft.Bcl.Async)

Compression

PCL for ZIP compression (Microsoft.Bcl.Compression)

BCL NuGet Packages (2/2)

Immutable Collections

Microsoft.Bcl.Immutable

Managed Extensibility Framework 2 (MEF 2)

PCL version of MEF 2 (Microsoft.Composition)

TPL Dataflow

Microsoft.Tpl.Dataflow

Azure NuGet Packages

.NET SDKs for Azure-related development

Management libraries
Client libraries

Get list of all Azure SDKs

http://azure.microsoft.com/en-us/documentation/api/

Additional useful libraries

Active Directory Authentication Library ADAL (Microsoft.IdentityModel.Clients.ActiveDirectory)
Azure SQL Database Elastic Scale

Popular MV* and DI Libraries

Castle.Core

Castle Windsor Inversion of Control container

MvvmLight

MVVM Light Toolkit

Caliburn.Micro

Caliburn.Micro MVVM Framework

Prism

Microsoft's Prism Library for composite applications using MVVM

Useful Other (less known?) Packages

AutoMapper
Convention-based object-object mapper

log4net
Apache logging framework

Autofac
Autofac Inversion of control container

Dapper
Micro-ORM

Useful Other (less known?) Packages

ClosedXML

Create and manipulate Excel 2007/2010 files

PDFSharp

Open Source .NET library to create and process PDF

Microsoft Report Viewer

Microsoft.ReportViewer

Creating NuGets

Creating your own NuGet packages

Creating NuGet Packages

Command line tool *nuget.exe*

Create packages (*Pack* Command)
Publish packages (*Push*, *Delete* Command)
Install packages (*Install*, *Restore*, *Update* Command)
Generate *nuspec*-files (*Spec* Command)
Can be used during automated build
Command line reference

NuGet Package Explorer

UI to create/modify/view NuGet packages and *nuspec* files
http://npe.codeplex.com/

```
<?xml version="1.0" encoding="utf-16"?>
<package xmlns="http://schemas.microsoft.com/packaging/2012/06/nuspec.xsd">
  <metadata>
    <id>CockpitFramework.Data</id>
    <version>$version$</version>
    <title>Cockpit Framework Data Layer</title>
    <authors>software architects gmbh</authors>
    <owners>software architects gmbh</owners>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <description>...</description>
    <releaseNotes></releaseNotes>
    <dependencies>
      <group targetFramework=".NETFramework4.0">
        <dependency id="CockpitFramework.Dependencies"
          version="[$version$]" />
      </group>
      <group targetFramework="sl5">
        <dependency id="CockpitFramework.Dependencies"
          version="[$version$]" />
      </group>
    </dependencies>
  </metadata>
```

Example

nuspec File

```
<dependencies>
  <group targetFramework=".NETFramework4.0">
    <dependency id="log4net" version="[1.2.11]" />
    <dependency id="Microsoft.SqlServer.Compact.Private"
      version="[4.0.8482.1]" />
    <dependency id="AvalonEdit" version="[4.2.0.8783]" />
    <dependency id="ClosedXML" version="[0.68.1]" />
    <dependency id="DocumentFormat.OpenXml" version="[1.0]" />
    <dependency id="IronPython" version="[2.7.3]" />
    <dependency id="LumenWorks.Framework.IO" version="[1.0.0]" />
    <dependency id="Newtonsoft.Json" version="[5.0.6]" />
    <dependency id="WindowsAzure.Storage" version="[2.0.5.1]" />
    <dependency id="Microsoft.Bcl.Async" version="[1.0.16]" />
  </group>

  <group targetFramework="sl5">
    …
  </group>
</dependencies>
```

Example

nuspec File

Version range syntax

```
1.0   = 1.0 ≤ x
(,1.0]  = x ≤ 1.0
(,1.0)  = x < 1.0
[1.0] = x == 1.0
(1.0) = invalid
(1.0,) = 1.0 < x
(1.0,2.0) = 1.0 < x < 2.0
[1.0,2.0] = 1.0 ≤ x ≤ 2.0
empty = latest version.
```

```
  <files>
    <!-- net4 -->
    <file src=".\$configuration$\TimeCockpit.Common.dll"
      target="lib\net4" />
    <file src=".\$configuration$\TimeCockpit.Data.dll"
      target="lib\net4"/>
    …

    <!-- sl5 -->
    <file src=".\SL\$configuration$\TimeCockpit.Common.dll"
      target="lib\sl5" />
    <file src=".\SL\$configuration$\TimeCockpit.Data.dll"
      target="lib\sl5" />
    …

    <!-- include source code  for symbols -->
    <file src=".\...\*.cs" target="src\TimeCockpit.Common" />
    <file src=".\...\*.cs" target="src\TimeCockpit.Data" />
</package>
```

Example

nuspec File

Folder Structure

For Details see NuGet Docs

```
\lib
    \net11
        \MyAssembly.dll
    \net20
        \MyAssembly.dll
    \net40
        \MyAssembly.dll
    \sl40
        \MyAssembly.dll
```

```
\content
    \net11
        \MyContent.txt
    \net20
        \MyContent20.txt
    \net40
    \sl40
        \MySilverlightContent.html

\tools
    init.ps1
    \net40
        install.ps1
        uninstall.ps1
    \sl40
        install.ps1
        uninstall.ps1
```

Versioning Notes

Things to remember

NuGet never installs assemblies machine-wide (i.e. not in GAC)
You cannot have multiple versions of the same DLL in one AppDomain

DLL Hell

Policy too loose: Problems with breaking changes
Policy too tight: Problems with library having dependencies on other libraries
(e.g. ANTLR and ASP.NET MVC, everyone depending on Newtonsoft JSON)

For Library Publishers: SemVer

X.Y.Z (Major.Minor.Patch)
Rethink your strong naming policies

```
<runtime>
   <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
         <assemblyIdentity name="mscorcfg"
           publicKeyToken="b03f5f7f11d50a3a" culture=""/>
         <bindingRedirect
           oldVersion="0.0.0.0-65535.65535.65535.65535"
           newVersion="1.0.3300.0"/>
      </dependentAssembly>
   </assemblyBinding>
</runtime>
```

Binding Redirects

Note: NuGet can generate this for you

Add-BindingRedirect Command

See online reference for details

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
    <package id="SomePackage"
      version="2.1.0"
      allowedVersions="[2,3)" />
</packages>
```

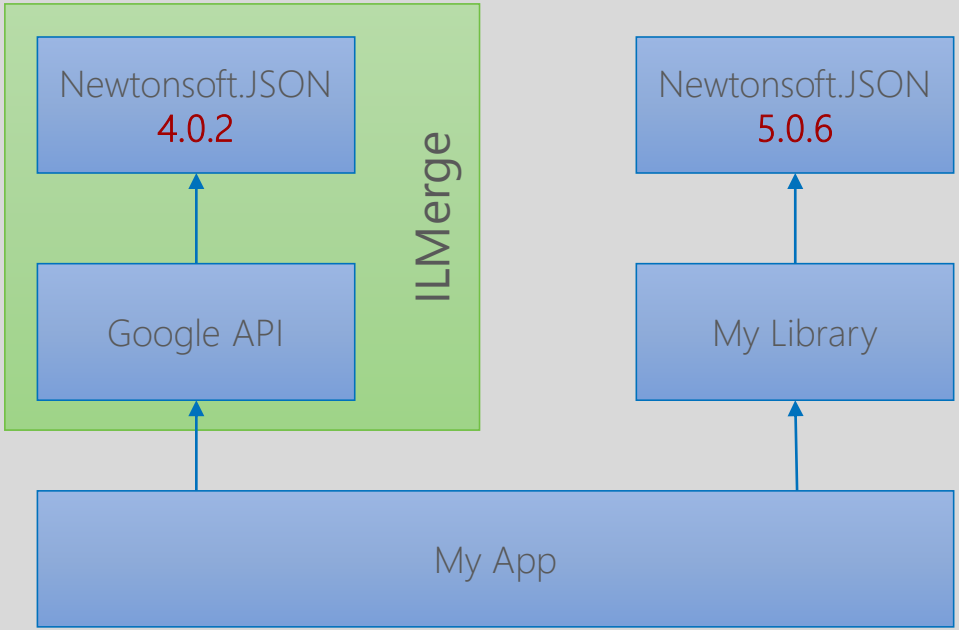
Versioning

Constraints in packges.config

Manual editing necessary

ILMerge

Soliving version conflicts



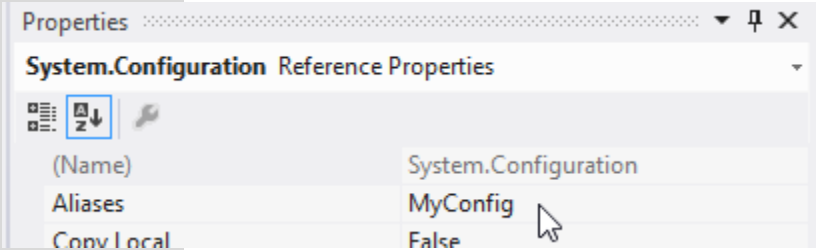
[Microsoft Download]

```
ILMerge
  "Assemblies\Google.Apis.Authentication.OAuth2.dll"
  "Assemblies\Google.Apis.dll"
  "Assemblies\Google.Apis.Latitude.v1.dll"
  "Assemblies\DotNetOpenAuth.dll" "Assemblies\log4net.dll"
  "Assemblies\Newtonsoft.Json.Net35.dll"
  /out:"c:\temp\Google.Apis.All.dll" /lib:"Lib„

extern alias MyConfig;
using Conf = MyConfig::System.Configuration;

namespace MyTinyMvvmToolkit
{
  public class NotificationObject
  {
    public void ReadConfiguration()
    {
      var setting =
        Conf.ConfigurationManager.AppSettings["MyDB"];
    }
  }
}
```

ILMerge

Solving version conflicts

C# *extern alias*



```
<?xml version="1.0" encoding="utf-16"?>
<package xmlns="http://schemas.microsoft.com/packaging/2012/06/nuspec.xsd">
  <metadata>…</metadata>

  <files>
    <file src="content\app.config.transform"
      target="content\" />
    <file src="content\TimeCockpitInitialization.cs.pp"
      target="content\" />
  </files>
</package>
```

```
<configuration>
    <startup>
      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
   </startup>
   <system.data>
      <DbProviderFactories>
         <remove invariant="System.Data.SqlServerCe.4.0"/>
         <add name="Microsoft SQL Server Compact Data Provider 4.0"
               invariant="System.Data.SqlServerCe.4.0"
               description=".NET Framework Data Provider for Microsoft SQL Server Compact"
               type="System.Data.SqlServerCe.SqlCeProviderFactory, System.Data.SqlServerCe,
               Version=4.0.0.1, Culture=neutral, PublicKeyToken=89845dcd8080cc91"/>
      </DbProviderFactories>
   </system.data>
</configuration>
```

Content Files

New in NuGet 2.6: XDT

```
…

namespace $rootnamespace$
{
  using System;

  /// <summary>
  /// Class taking care of cockpit framework initialization
  /// </summary>
  public class TimeCockpitInitialization
  {
    …
  }
}
```

Content Files

Sourcecode Transformations in .cs.pp File

Available properties see MSDN

User PowerShell scripts to modify project properties
NuGet Docs

Publishing NuGet Packages

http://www.nuget.org
Public NuGet Feed

Create your private feed
File system
Private NuGet Server
For details see NuGet Help

Use a NuGet SaaS like *MyGet*
http://www.myget.org/