

# #Develop – an Introduction

## VO DevCon Europe 2004

Mike Krüger  
Senior Code Wrangler  
[mike@icsharpcode.net](mailto:mike@icsharpcode.net)



# Agenda 1/2

## ■ #Develop

- IDE Overview
- License

## ■ Projects & Combines

## ■ Editor Features

## ■ Forms Designer

## ■ Quality Management in #Develop

- Static Code Analysis
- Unit Tests
- Profiling

# Agenda 2/2

- XML Documentation
- Tips & Tricks
- Things to come

# IDE Feature Overview 1/2

- Different Programming Languages are supported
  - C#, VB.NET, C++.NET, IL Assembler
  - Non-.NET Languages can be supported
- Numerous Tool Windows: Project Browser, Class Browser, Task List, Sidebar etc.
- Rich Options
- Conversion from C# to VB.NET and vice versa
- Localized for 20 Languages

# IDE Features Overview 2/2

- .NET Object Browser
- Integrated .NET Help
- Visual Studio .NET Solution Importer/Exporter
- Component oriented Design
  - Easily extensible
  - Simple Integration of external Tools
- Actives Internet Forum/User Community

# License

- #Develop is free in the GPL sense
  - #Develop is free (as in Beer)
  - Source code is freely accessible
  - Anybody may modify #Develop (As long as the Modification is licensed under the GPL)
- Copyright Assignment for all Contributions
  - Copyright remains in one Hand
  - #Develop can be defended in Court if Need should arise
  - #Develop or Components can be commercially licensed
- #Develop is available commercially

# Projects & Combines

## ■ Projects

- Contain Source Files
- Are compiled into an Assembly
- Only contain Source Code in ONE language

## ■ Combines

- Contain Projects and Combines (More powerful than VS.NET Solutions!)
- Intended for Structuring of sizeable Projects

# Editor Features

- Configurable Syntax Highlighting
  - Nested Highlighting is possible
- Code Autogeneration
- Code Templates
- Intelligent Indenting/Bracketing
- Bracket Matching
- Bookmarks, Splitting, Code Completion, Folding
- Also can be used outside #Develop

# Forms Designer

- Like in VS.NET, as the Forms Designer is part of the .NET Infrastructure
  - Drag & Drop (Component selection in the Sidebar)
  - Menu Designer
  - Properties/Events Panel
- For C#, VB.NET and XML Forms
  - XML can be used by ANY .NET Language
  - Extensible for new Languages (Parser for the Source Language is required)

develop

ic#code

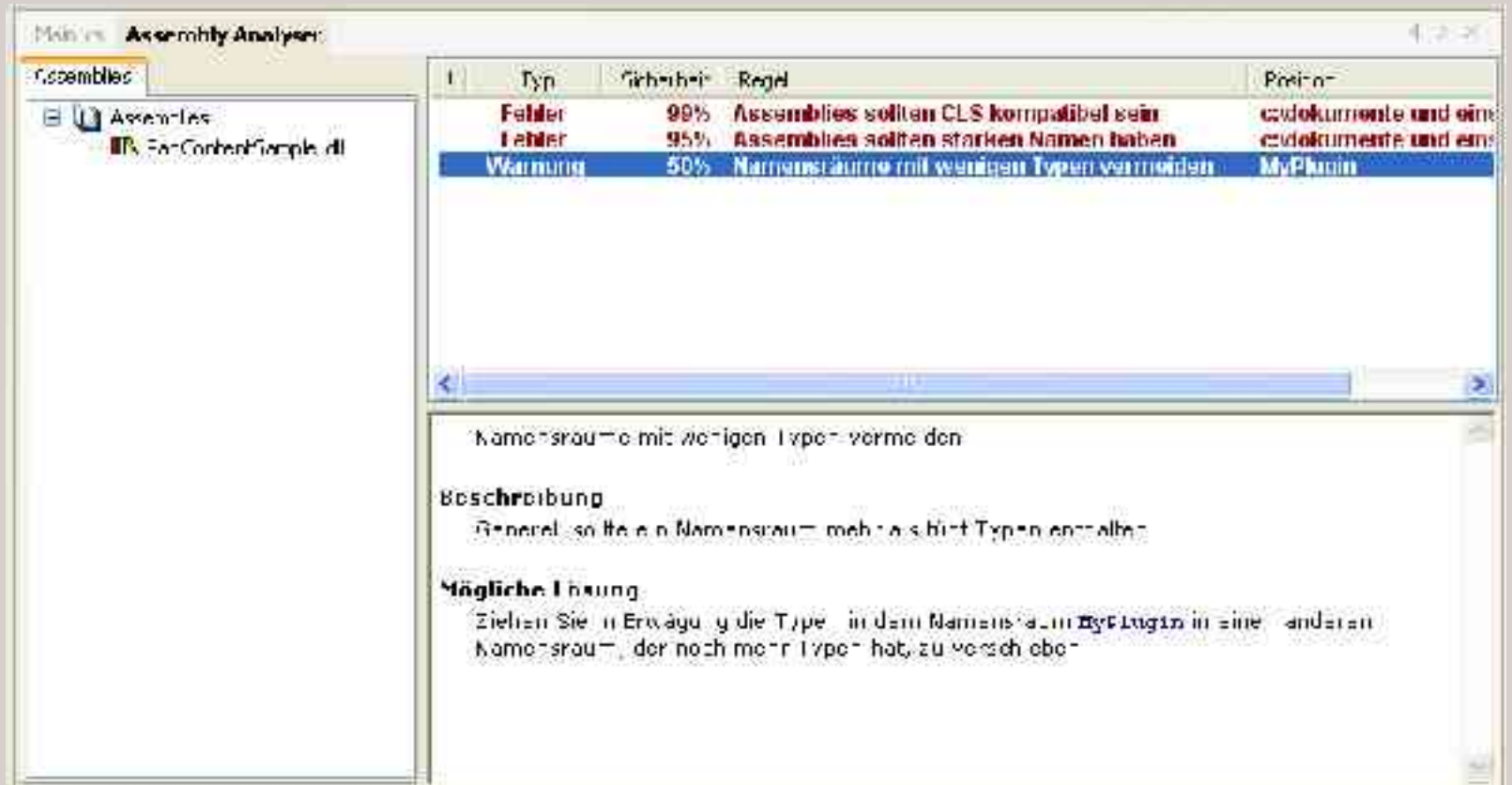
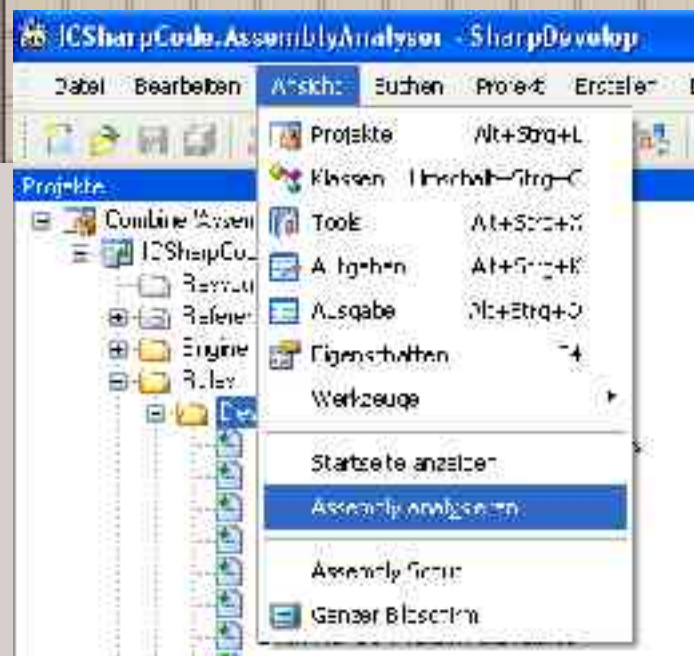
# DEMO

# Quality Management

- Goal: Automatic Quality Control of the Project Code
  - Static Code Analysis – Checking for Compliance with .NET Conventions
  - Unit Tests – Automated Testing of individual Modules
  - Profiling – Verification of Run Time Behaviour (For Performance Optimization)

# Static Code Analysis

- Tests for semantic 'Ugliness' not found by the Compiler
  - Naming (Pascal/Camel-Casing, EventHandler Renaming etc.)
  - Assembly Attributes (Strong Naming, CLS Compliance, Version number etc.)
  - Structure (e.g. No empty Interfaces, Properties should not be set only etc.)
- New Checks can be easily added



# Unit Tests

## ■ What are Unit Tests?

- „Selftesting Code“
- Small methods testing all Variants of a Method Call
- Test for expected Behaviour

## ■ Why Unit Tests?

- Increased Confidence in Code
- Acceleration of Development
  - Errors are found earlier – less Debugging
  - Regression is avoided
- Code Changes are safer

# Unit Tests in #Develop

## ■ #Develop integrates NUnit

- Port of JUnit
- NUnit is free
- „Quasi Standard“
- Even Used in the MS Development Process Best Practice

The screenshot displays the Visual Studio IDE with the NUnit test framework integrated. The main window shows the source code for a test class named `MyTest` in the `UnitTests` namespace. The code includes a `TestFixture` attribute and two test methods: `TestMethod` and `FailTestMethod`. The `FailTestMethod` method is shown failing, with the output window displaying the error message: `UnitTests.Tests.MyTest.FailTestMethod failed: expected: <15> but was: <5> at NUnit.Framework.Assert.Fail(String message, Object[] args)`. The Test Explorer window on the right shows the test results, with `FailTestMethod` marked as failed (red) and `TestMethod` as passed (green).

```
using NUnit.Framework;

[TestFixture]
public class MyTest
{
    [Test]
    public void TestMethod()
    {
        Assert.AreEqual(5, 5);
    }

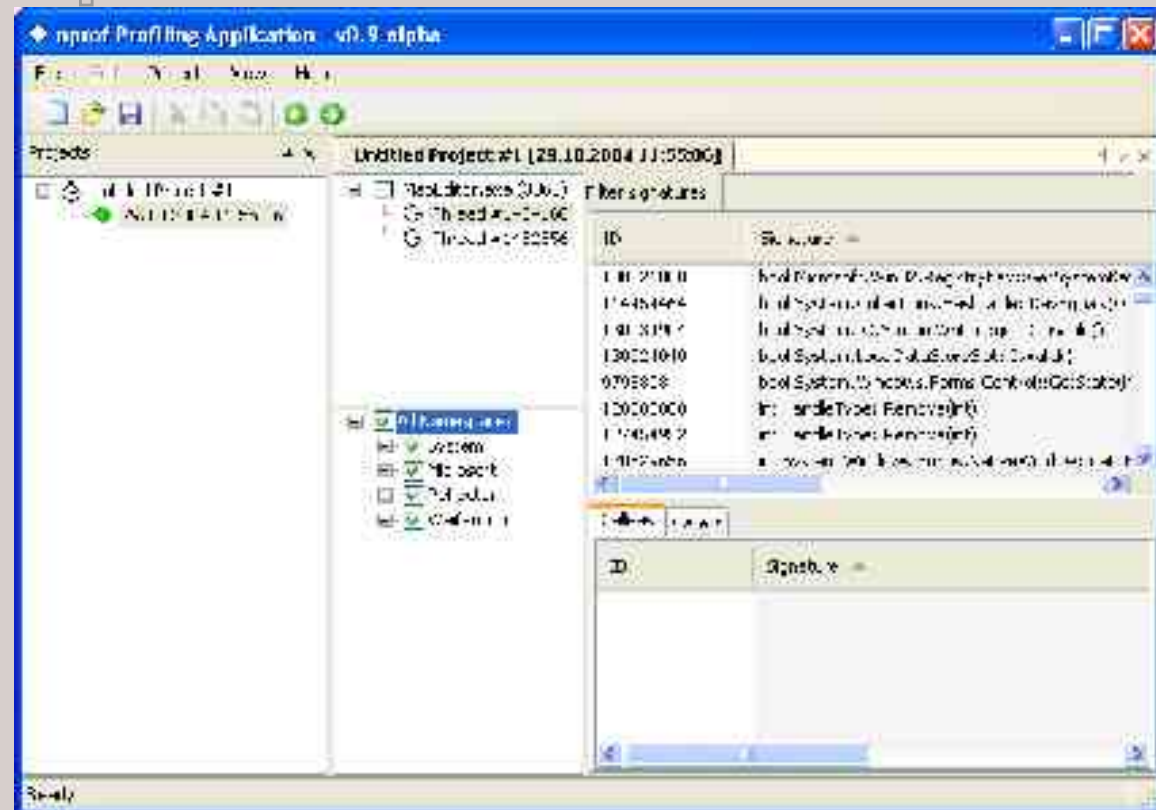
    [Test]
    public void FailTestMethod()
    {
        Assert.AreEqual(15, 5);
    }
}
```

Output:

```
UnitTests.Tests.MyTest.FailTestMethod failed:
    expected: <15>
    but was: <5>
    at NUnit.Framework.Assert.Fail(String
message, Object[] args)
```

# Profiling

- Analysis of the Run Time Behaviour of a Program
- #Develop integrates Nprof
- Simplifies Finding of Bottlenecks



develop

ic#code

# DEMO

# Tips & Tricks

- Selecting the Language Environment
  - The „Feel“ of the IDE can be changed this Way
  - Affects the Display of Classbrowser, Code Completion etc.
- Standard Headers
- Projects can be exported to HTML
- Wordcount (#Develop: about 220 kLOC)
- Regular Expression Toolkit

# XML Documentation

- Editor supports XML-Doc Input
- Brief XML „Preview“ in the Editor
- Generation of complete XML Documentation using NDoc

develop

ic#code

# DEMO

# Things to come

- Debugger
- Subversion Support
- .NET 2.0 Support
- Refactoring Browser
- Localization for even more Languages
- ASP.NET 2 Support and other Add-ins through external Contributors

# Links

- #Develop Homepage:  
[www.icsharpcode.net/OpenSource/SD](http://www.icsharpcode.net/OpenSource/SD)
- NUnit : [www.nunit.org](http://www.nunit.org)
- NProf: [nprof.sourceforge.net](http://nprof.sourceforge.net)
- NDoc: [ndoc.sourceforge.net](http://ndoc.sourceforge.net)

develop

ic#code

# Questions?



develop

ic#code