Releasing your potential

logica

AOP mit PostSharp

Referent: Thomas Mentzel

Agenda

- AOP
- Alternative Frameworks (MS PIAB, Spring.NET)
- PostSharp
- Beispiele
  - Logging
  - Validation
  - Security
  - Windows Forms

AOP

Aspektorientierte Programmierung (AOP) ist ein **Programmierparadigma**, […] das anstrebt, verschiedene **logische Aspekte** eines Anwendungsprogramms […] **getrennt voneinander** zu entwerfen, zu entwickeln und zu testen.

Die getrennt entwickelten Aspekte werden dann zur **endgültigen Anwendung zusammengefügt**

Aus Wikipedia
http://de.wikipedia.org/wiki/Aspektorientierte_Programmierung

Policy Injection Application Block - Microsoft

- http://msdn.microsoft.com/en-us/library/cc511729.aspx
- Lizenz: Microsoft Library
- Aktuelle Version: Enterprise Library 4.0

- Bestandteil der Microsoft Enterprise Library 4.0
- Prinzip: Proxy

- Beispiel:

```
TargetClass theTarget =
  PolicyInjection.Create<TargetClass>(parameter1, parameter2);
```

Framework: Spring.NET

- http://www.springframework.net/
- Lizenz: Apache License, Version 2.0
- Aktuelle Version: Spring.NET 1.3.0 RC1

- AOP ist ein Aspekt ;) des Spring.NET Frameworks
- Prinzip: Proxy

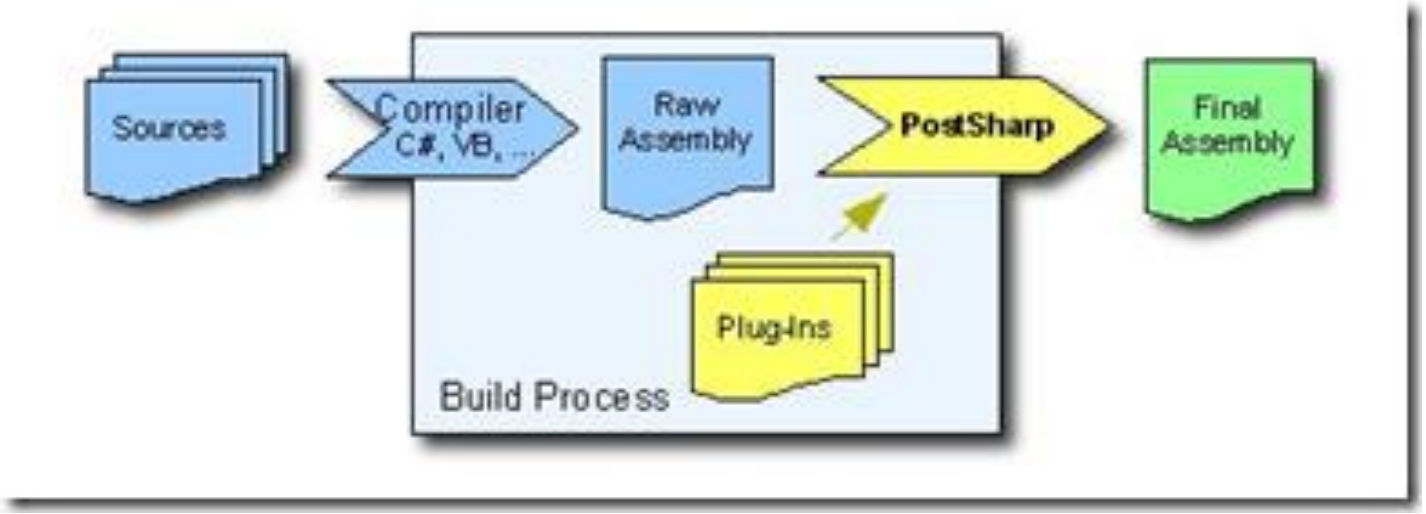
- Beispiel:

```
ProxyFactory factory = new ProxyFactory(new ServiceCommand());
factory.AddAdvice(new ConsoleLoggingAroundAdvice());
ICommand command = (ICommand) factory.GetProxy();
command.Execute("This is the argument");
```

PostSharp

- http://www.postsharp.org/
- Lizenz: commercial/GPL-LGPL dual license
- Version: 1.0 SP2 bzw. 1.5 RTM

- Prinzip: Attribute & Post-Compiler



Beispiele

- Demo
  - Logging
  - Validation
  - Caching
- Live Projekt
  - Security

Post Compiler Magic

```
[LogAcpect()]
private static void DoSomething()
{
    Console.WriteLine("Hello World");
}
```

```
[Serializable()]
public class LogAcpect : OnMethodBoundaryAspect
{
    public override void OnEntry(MethodExecutionEventArgs eventArgs)
    {
        Console.WriteLine("--- Calling: {0} ---", eventArgs.Method);
    }

    public override void OnExit(MethodExecutionEventArgs eventArgs)
    {
        Console.WriteLine("--- Leaving: {0} ---", eventArgs.Method);
    }

    public override void OnSuccess(MethodExecutionEventArgs eventArgs)
    {
        Console.WriteLine("--- Success: {0} ---", eventArgs.Method);
    }

    public override void OnException(MethodExecutionEventArgs eventArgs)
    {
        Console.WriteLine("--- Exception: {0} @ {1} ---", eventArgs.Exception.Message, eventArgs.Method);
    }
}
```

Post Compiler Magic

```
private static void DoSomething()
{
    MethodExecutionEventArgs ~laosEventArgs~1;
    try
    {
        ~laosEventArgs~1 = new MethodExecutionEventArgs(~PostSharp~Laos~Implementation.~targetMethod~1,
        ~PostSharp~Laos~Implementation.LogAcpect~1.OnEntry(~laosEventArgs~1);
        if (~laosEventArgs~1.FlowBehavior != FlowBehavior.Return)
        {
            Console.WriteLine("Hello World");
            ~PostSharp~Laos~Implementation.LogAcpect~1.OnSuccess(~laosEventArgs~1);
        }
    }
    catch (Exception ~exception~0)
    {
        ~laosEventArgs~1.Exception = ~exception~0;
        ~PostSharp~Laos~Implementation.LogAcpect~1.OnException(~laosEventArgs~1);
        switch (~laosEventArgs~1.FlowBehavior)
        {
            case FlowBehavior.Continue:
            case FlowBehavior.Return:
                return;
        }
        throw;
    }
    finally
    {
        ~PostSharp~Laos~Implementation.LogAcpect~1.OnExit(~laosEventArgs~1);
    }
}
```

Beispiele

- Windows Forms Demo
  - Logik in Aspekten
  - Controller als „MessageBus“

Fragen

Es gibt keine dummen Fragen, es gibt nur dumme Antworten!

E-Mail/MSN: thomas.mentzel@logica.com

Blog: http://thomas.mentzel.name

Twitter: http://twitter.com/ThomasMentzel