Logging (mit log4net)

17.02.2009

Thomas Mentzel

EMail: **thomas.mentzel@logica.com**
Website: http://www.logica.com/de

Agenda

- Captain's log, USS Enterprise
- Do It Yourself
- Buy it, don‘t make it
- Mehr ist Mehr ist Mehr ist Mehr

„Captain's log, USS Enterprise“

„Eine Logdatei (engl. log file) beinhaltet das automatisch erstellte Protokoll *aller oder bestimmter Aktionen* von Prozessen auf einem Computersystem. […]

Wichtige Anwendungen finden sich vor allem im Bereich der Prozess-Kontrolle und im Automatisierungsbereich. Prinzipiell werden *alle Aktionen* mitgeschrieben, für die ein *späteres Audit* erforderlich ist oder sein könnte. […]“ - Wikipedia

Loggen

- Zentrale Komponenten wie Controller
- Datenbankoperationen / Persistenzschicht
- Zwischenergebnisse bei Algorithmen
- *Hinweis: Visual Studio Output Window ist sehr langsam, also nicht wundern!*

Vorteile (Entwicklung)

- Debuggen ohne Breakpoint
- Ablaufkontrolle
- Blick unter die Motorhaube beim Fahren
- →Integrieren in die normale Arbeit

Vorteile (Kunden)

- Nützliche objektive Informationen
- Detailierte Fehlerverfolgung
- Reproduzieren des Fehlers
- →Fehleranalyse beim Kunden

Loggen?

...
2009-02-13 11:12:41,372 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:41,512 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,184 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,200 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
...
...
2009-02-13 11:12:44,403 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,434 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,434 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,450 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,450 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,450 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,450 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
2009-02-13 11:12:44,450 [1] INFO Vap.Psm.Basisklassen.Controller [(null)] - Controller.ReportProgress wird ausgelöst
...

2009-02-13 11:12:10,184 [1] DEBUG Vap.Psm.Basisklassen.SqlCommandCache [(null)] - Initialisieren der Datenbank
2009-02-13 11:12:10,184 [1] DEBUG Vap.Psm.Basisklassen.SqlCommandCache [(null)] - Initialisieren des Comand Objektes
2009-02-13 11:12:10,215 [1] DEBUG Vap.Psm.Basisklassen.SqlCommandCache [(null)] - Initialisieren des Comand Objektes
2009-02-13 11:12:10,215 [1] DEBUG Vap.Psm.Basisklassen.SqlCommandCache [(null)] - Initialisieren des Comand Objektes

Loggen!

2009-02-13 11:12:08,762 [1] DEBUG Vap.Psm.Datenzugriff.DBInit [(null)] - Erstellen der Prozedur/View 'Vap.Psm.Datenzugriff.Sql.SVAktualisiereBefAlterChance.prc.init.sql'

2009-02-13 11:12:08,778 [1] DEBUG Vap.Psm.Datenzugriff.Internal.DBBase [(null)] - Ausführen von IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = OBJECT_ID(N'[SVAktualisiereBefAlterChance]') AND OBJECTPROPERTY(id,N'IsProcedure') = 1)

2009-02-13 11:12:08,793 [1] DEBUG Vap.Psm.Datenzugriff.Internal.DBBase [(null)] - Ausführen von CREATE PROCEDURE [SVAktualisiereBefAlterChance]

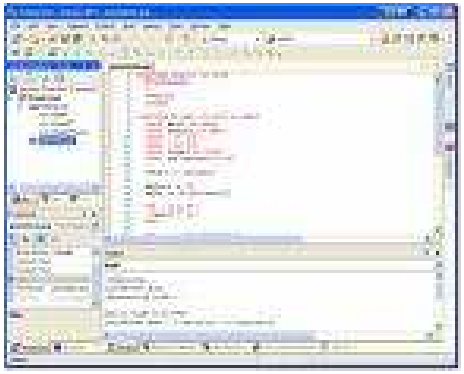
2009-02-13 11:12:08,793 [1] DEBUG Vap.Psm.Datenzugriff.DBInit [(null)] - Erstellen der Prozedur/View 'Vap.Psm.Datenzugriff.Sql.SVAktualisierePerskat.prc.init.sql'

2009-02-13 11:12:08,793 [1] DEBUG Vap.Psm.Datenzugriff.Internal.DBBase [(null)] - Ausführen von IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = OBJECT_ID(N'[SVAktualisierePersKat]') AND OBJECTPROPERTY(id,N'IsProcedure') = 1)

2009-02-13 11:12:08,809 [1] DEBUG Vap.Psm.Datenzugriff.Internal.DBBase [(null)] - Ausführen von CREATE PROCEDURE [SVAktualisierePersKat]

Demo 1: Einfaches Loggen

- Logging einer Math-Klasse
  - Logging über die Ausgabe- und Debug-Konsole
  - Logging einfach abschaltbar



Nachteile

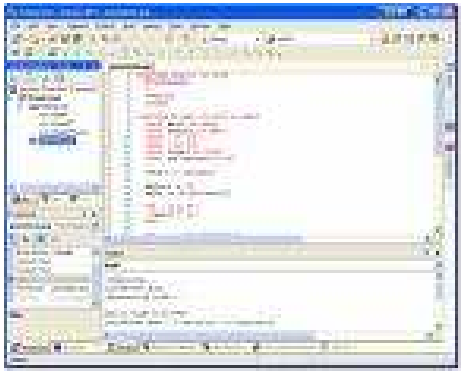
- KISS Prinzip verletzt
- DRY Prinzip verletzt
- OCP verletzt
- Abstufung der Fehler (Level)

Do It Yourself

- Erweiterbarkeit der Ausgabe
  - Datei, Datenbank, …
- Fehler-Kontext
- Einfache Einbindung
  - Wenig Code (eine Zeile)
- Kategorisierung der Fehler
  - Error, Info, Debug
  - Fatal, Warn

Demo 2: Logging Framework Selbstbau

- Logging Bibliothek
  - Wiederverwendbar
  - Erweiterbar
  - Einfach zu benutzen



Do It Yourself (Review)

- Erweiterbarkeit der Ausgabe
  - Datei, Datenbank, …
- Fehler-Kontext
- Einfache Einbindung
  - Wenig Code (eine Zeile)
- Kategorisierung der Fehler
  - Error, Info, Debug
  - Fatal, Warn

- IAppender
- ILog
- Logger Factory
- Static Instance
- ILog.{LogLevel}
- Log-Level

Buy it, don‘t make it

- Fertige Frameworks …
  - sind günstiger als selbst erstellte
  - enthalten mehr Features
  - sind vielfach bewährt
  - sind besser dokumentiert
  - haben eine gute Architektur

log4net

- Apache Logging Project
  - http://logging.apache.org/log4net/index.html

- Apache License, Version 2.0

Log4net Features

- Mehrere Logging Ziele (Appender)
- XML Konfiguration
- Modular und Erweiterbar
- Logger Hierarchie

Logging in 5 Minuten

Referenz hinzufügen: log4net.dll

Assembly.cs: XML Konfiguration benutzen

```
[assembly: XmlConfigurator()]
```

Logging in 5 Minuten

App.config

```
<configSections>
   <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,log4net" />
</configSections>

<log4net>
   <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
      <layout type="log4net.Layout.PatternLayout">
         <conversionPattern value="%date [%thread] %-5level %logger - %message%newline" />
      </layout>
   </appender>

   <root>
      <level value="DEBUG"/>
      <appender-ref ref="ConsoleAppender" />
   </root>

</log4net>
```

Logging in 5 Minuten

Quellcode

// logger.snippet –über „logger“ erreichbar in der IDE

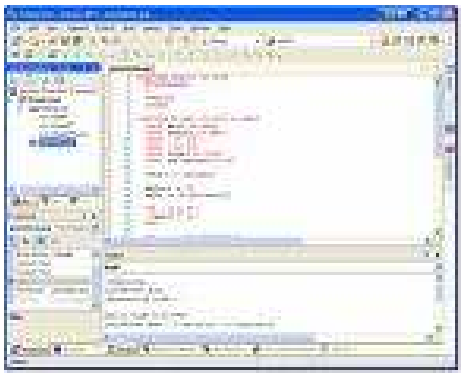
```
#region Log4Net

        /// <summary>
        /// log4net Klassen-Logger
        /// </summary>
        private static readonly log4net.ILog log =
            log4net.LogManager.GetLogger(System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);

#endregion
```

// log.snippet –über „log“ erreichbar in der IDE

```
if (log.IsErrorEnabled) log.Error("...");
if (log.IsDebugEnabled) log.Debug("...", exception);
```

Demo 3: log4net Integration

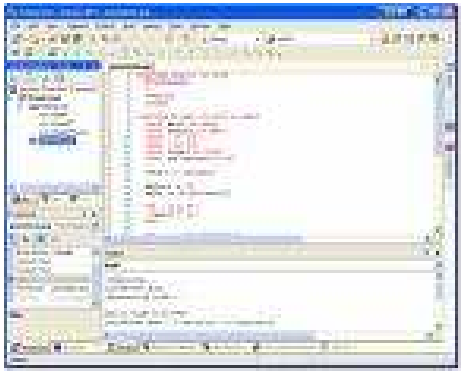
- Logging mit log4net
  - Integration
    - Math.cs
    - AssemblyInfo.cs
  - Konfiguration
    - app.config



Logger Hierarchie

- Kinder erben die Konfiguration ihrer Väter
- Namespace („.“) bildet Hierarchie
- Konfiguration der Kinder überschreibbar

```
<logger name="BonnToCode.Logging.Demo.Math">
    <level value="Info" />
</logger>
```



Erweiterbarkeit

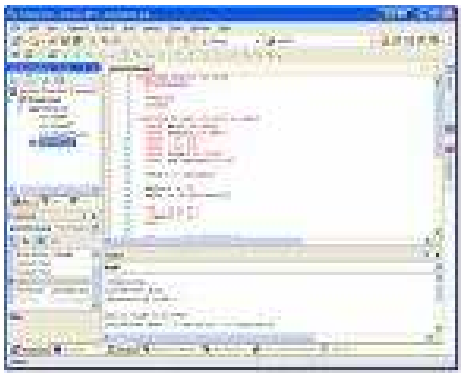
- IAppender
  - Ausgabeziel
- ILayout
  - Aussehen der Ausgabe

Toolbox Logger

- Ziel: Logging in eine optionale Toolbox
- IAppender: ToolboxAppender-Klasse
  - Inherits AppenderSkeleton
  - Static Instanz der Logging Form
- Formular: LoggingForm-Klasse
  - Formular mit TextBox für die Meldungen
  - Methode „Append“ für Fehlermeldungen
- XML Konfiguration (ToolboxAppender)

Demo 3: log4net Erweiterung I

- log4net Erweiterung
  - ToolBox Appender
  - Einbinden in Konfiguration



Toolbox Logger Properties

- Ziel: Breite und Höhe der Toolbox über die Konfiguration festlegen

- ToolboxAppender: Properties einfügen

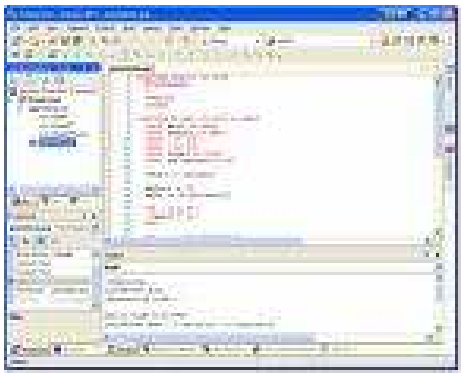
```
public int Height
    { get; set; }

public int Width
    { get; set; }
```

- XML Appender Konfiguration erweitern

```
<appender [...]>
        <height value="300" />
        <width value="200" />
        <layout [...]>
                [...]
        </layout>
</appender>
```

Demo 3: log4net Erweiterung II

- log4net Erweiterung
  - Toolbox Appender Properties
  - Properties in Konfiguration



Fragen?