.NET 2.0 og C# 2.0

Denne artikel viser hvordan man kan bruge nogle af de nye features i .NET 2.0 og C# 2.0. Det er ikke en komplet oversigt.

Den forudsætter et vist kendskab til .NET og C#.

Skrevet den **14. feb 2010** af **arne_v** I kategorien **Programmering / C#** | ★★★☆☆

Historie:
V1.0 - 08/05/2005 - original
V1.1 - 25/05/2005 - tilføj link til ændrings liste og appetitvækker liste
V1.2 - 24/07/2005 - tilføj release dato
V1.3 - 22/11/2207 - opdater så den reflekterer at den er udkommet i endelig udgave
V1.4 - 26/12/2008 - små ændringer
V1.5 - 14/02/2010 - smårettelser

Indledning

Jeg vil beskrive nogle af de nye features i .NET 2.0 og C# 2.0, forklare hvad de kan bruges til og vise nogle meget simple eksempler.

Mine eksempler vil typisk være simplere end dem i dokumentationen.

Jeg vil ikke vise Win Forms eller Web Forms specifikke ting. Kun generelle features. Men de kan naturligvis sagtens bruges i Win Forms og Web Forms.

Der er skrevet rigtigt meget om .NET 2.0 og C# 2.0, og hvis du har læst det meste, så finder du næppe meget nyt i denne artikel.

FTP support

En rigtig træls ting i .NET 1.1 er at WebRequest kun understøtter HTTP og ikke FTP. Det er fixet i .NET 2.0.

Ftp2.cs

```
using System;
using System.IO;
using System.Net;

class Ftp2
{
    public static void Main(string[] args)
    {
        FtpWebRequest req =
(FtpWebRequest)WebRequest.Create("ftp://192.168.1.10/utilities/utilities.zip")
```

```
;
        FtpWebResponse resp = (FtpWebResponse)req.GetResponse();
        Stream f1 = resp.GetResponseStream();
        Stream f2 = new FileStream("utilities.zip", FileMode.CreateNew,
FileAccess.Write);
        byte[] b = new byte[1000];
        int n;
        while((n = f1.Read(b, 0, b.Length)) > 0)
        {
            f2.Write(b, 0, n);
        }
        f2.Close();
        f1.Close();
        resp.Close();
    }
}
```

En feature som jeg tror rigtigt mange bliver glade for.

Ping support

.NET 2.0 har også fået en Ping klasse.

Ping2.cs

```
using System;
using System.Net.NetworkInformation;
using System.Text;

public class Ping2
{
    public static void Main(string[] args)
    {
        Ping p = new Ping();
        PingReply pr = p.Send("www.tv2.dk");
        if(pr.Status == IPStatus.Success)
        {
            Console.WriteLine("TV2 oppe (tid = " + pr.RoundtripTime + " ms)");
        }
        else
        {
            Console.WriteLine("TV2 nede");
        }
   }
}
```

Sikkert nyttigt for nogle.

Web server support

Og så er der lavet et framework for at skrive web servere i C#.

WebServer2.cs

```
using System;
using System.IO;
using System.Net;

// kaldes med:
//   http://localhost/test/show
//   http://localhost/test/exit
public class WebServer2
{
    public static void Main(string[] args)
    {
        HttpListener srv = new HttpListener();
        srv.Prefixes.Add("http://localhost/test/");
        srv.Start();
        while(true)
        {
            HttpListenerContext ctx = srv.GetContext();
            HttpListenerRequest req = ctx.Request;
            HttpListenerResponse resp = ctx.Response;
            StreamWriter sw = new StreamWriter(resp.OutputStream);
            if(req.Url.AbsolutePath == "/test/exit")
            {
                sw.WriteLine("farvel");
                sw.Close();
                break;
            }
            else if(req.Url.AbsolutePath == "/test/show")
            {
                foreach(string hdrnam in req.Headers.AllKeys)
                {
                    string hdrval = req.Headers[hdrnam];
                    sw.WriteLine(hdrnam + " = " + hdrval + "<br>");
                }
            }
            else
            {
                sw.WriteLine("ukendt kommando");
            }
            sw.Close();
        }
        srv.Stop();
        srv.Close();
    }
}
```

Man kan godt grine lidt af det. Hvem vil skrive en konkurrent
til Apache og IIS ? Men det er faktisk almindeligt idag med web

interfaces til programmer uden GUI. Så selvom man ikke har lyst til at konkurrere med Apache og IIS så kan man jo godt have et program som kører i baggrunden og som man gerne vil styre via en browser !

Jeg tror nok at der er nogen som får fornøjelse af den feature.

GZip support

.NET har indbygget support for streams til og fra GZIP format.

Gzip2.cs

```
using System;
using System.IO;
using System.IO.Compression;

class GZip2
{
    private static void CopyData(Stream f1, Stream f2)
    {
        byte[] b = new byte[1000];
        int n;
        while((n = f1.Read(b, 0, b.Length)) > 0)
        {
            f2.Write(b, 0, n);
        }
        f2.Close();
        f1.Close();
    }
    public static void Compress(string fnm1, string fnm2)
    {
        CopyData(new FileStream(fnm1, FileMode.Open, FileAccess.Read),
                 new GZipStream(new FileStream(fnm2, FileMode.CreateNew,
FileAccess.Write), CompressionMode.Compress));
    }
    public static void Decompress(string fnm1, string fnm2)
    {
        CopyData(new GZipStream(new FileStream(fnm1, FileMode.Open,
FileAccess.Read), CompressionMode.Decompress),
                 new FileStream(fnm2, FileMode.CreateNew, FileAccess.Write));
    }
    public static void Main(string[] args)
    {
        Compress("test.txt", "test.txt.gz");
        Decompress("test.txt.gz", "newtest.txt");
    }
}
```

Jeg synes at det er en suveræn god ide med en compress & decompress stream. Men jeg er lidt mere skeptisk overfor valget af GZIP. Enhver

Linux mand bruger GZIP, men den er altså ikke så udbredt på Windows.
Jeg synes hellere at man skulle have puttet #ziplib ind i .NET.

Men muligheden er der ihvertfald nu.

Console farver

Og her er en ny feature til ære for alle som kodede DOS applikationer i første halvdel af 90'erne.

ConsoleColor2.cs

```
using System;

public class ConsoleColor2
{
    public static void Main(string[] args)
    {
        ConsoleColor fg = Console.ForegroundColor;
        ConsoleColor bg = Console.BackgroundColor;
        Console.WriteLine("normal");
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine("fancy");
        Console.ForegroundColor = fg;
        Console.BackgroundColor = bg;
    }
}
```

Mere kuriøst end nyttigt.

Partial classes

C# 2.0 understøtter at en klasse splittes i flere filer.

PartFrag2.cs

```
public partial class X
{
    public void m1()
    {
        m2();
    }
}
```

Part2.cs

```
using System;

public partial class X
{
    public void m2()
    {
        Console.WriteLine("Det virker");
    }
    public static void Main(string[] args)
    {
        X x = new X();
        x.m1();
    }
}
```

Ingen ved deres fulde fem vil skrive en klasse i 2 filer. Men der er en ekstrem nyttig anvendelse af denne feature. Wizards ! I .NET 1.1 fik man tit 1 fil med en sektion som blev redigeret via en Wizard og nogle sektioner som man redigerede manuelt. Og kom man til at redigere manuelt i wizardens sektion, så kunne det gå grueligt galt. Og selvom man ikke kom til det, så var wizard sektionen tit meget lang og meget grim med det resultat at hele filen blev svær at læse. Det er løst nu. Fordi nu kan wizard sektionen ryge i en fil og de manuelt redigerede sektioner i en anden fil. Pæn adskillelse af tingene.

Database connections

Et kritik punkt mod ADO.NET i .NET 1.1 sammenlignet med ADO (og f.eks. JDBC) var at man skulle instantiere database connections med database specifikke klasser, hvilket gjorde det meget nemt at skrive database afhængig kode.

Lad os først se et eksempel på en workaround til .NET 1.1:

MultiDb11.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data.Odbc;

public class MultiDb
{
    public static IDbConnection GetConnection(string constr)
    {
        if(constr.ToUpper().IndexOf("DRIVER=") >= 0)
        {
            return new OdbcConnection(constr);
```

```
        }
        else if(constr.ToUpper().IndexOf("PROVIDER=") >= 0)
        {
            return new OleDbConnection(constr);
        }
        else if(constr.ToUpper().IndexOf("TRUSTED_CONNECTION=") >= 0 ||
                constr.ToUpper().IndexOf("INTEGRATED SECURITY=") >= 0)
        {
            return new SqlConnection(constr);
        }
        else
        {
            return null;
        }
    }
}

class MultiDb11
{
    private static void test(string constr)
    {
        IDbConnection con = MultiDb.GetConnection(constr);
        con.Open();
        IDbCommand cmd = con.CreateCommand();
        cmd.CommandText = "SELECT * FROM T1";
        IDataReader rdr = cmd.ExecuteReader();
        while(rdr.Read())
        {
            int f1 = (int)rdr[0];
            string f2 = (string)rdr[1];
            Console.WriteLine(f1 + " " + f2);
        }
        con.Close();
    }
    public static void Main(string[] args)
    {
        test(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Databases\MSAccess\Test.mdb");
        test("server=ARNEPC3;Integrated Security=SSPI;database=Test");
    }
}
```

Lidt besværligt. Og de fleste opgav også og brugte bare new SqlConnection eller new OleDbConnection i koden.

Men når man var startet på det så var det jo naturligt at fortsætte med new SqlCommand eller new OleDbCommand etc..

.NET 2.0 har en indbygget måde at gøre det på.

MultiDb2.cs

```
using System;
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data.Odbc;

class MultiDb2
{
    private static void test(string provider, string constr)
    {
        DbProviderFactory dbf = DbProviderFactories.GetFactory(provider);
        IDbConnection con = dbf.CreateConnection();
        con.ConnectionString = constr;
        con.Open();
        IDbCommand cmd = con.CreateCommand();
        cmd.CommandText = "SELECT * FROM T1";
        IDataReader rdr = cmd.ExecuteReader();
        while(rdr.Read())
        {
            int f1 = (int)rdr[0];
            string f2 = (string)rdr[1];
            Console.WriteLine(f1 + " " + f2);
        }
        con.Close();
    }
    public static void Main(string[] args)
    {
        test("System.Data.OleDb", @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Databases\MSAccess\Test.mdb");
        test("System.Data.SqlClient", "server=ARNEPC3;Integrated
Security=SSPI;database=Test");
    }
}
```

Meget bedre. Om folk så vil bruge den her måde (og IDbCommand, IDataReader etc.) må tiden så vise. Det burde de medmindre de har meget specielle behov for database specifikke features. Men helt klart en meget nyttig feature.

Hvis du skal have tilføjet nye databaser (som understøtter .NET 2.0) så skal de ligges ind i machine.config eller app.config configuration/system.data/DbProviderFactories.

Eksempel på app.config for FireBird, SQLServer CE og SQLite:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.data>
    <DbProviderFactories>
      <add name="SQLite Data Provider"
```

```
           invariant="System.Data.SQLite"
           description=".Net Framework Data Provider for SQLite"
           type="System.Data.SQLite.SQLiteFactory, System.Data.SQLite" />
      <add name="Microsoft SQL Server Compact Data Provider"
           invariant="System.Data.SqlServerCe"
           description=".NET Framework Data Provider for Microsoft SQL Server
Compact"
           type="System.Data.SqlServerCe.SqlCeProviderFactory,
System.Data.SqlServerCe" />
      <add name="FirebirdClient Data Provider"
           invariant="FirebirdSql.Data.FirebirdClient"
           description=".Net Framework Data Provider for Firebird"
           type="FirebirdSql.Data.FirebirdClient.FirebirdClientFactory,
FirebirdSql.Data.FirebirdClient" />
    </DbProviderFactories>
  </system.data>
</configuration>
```

Generics

Generics er en feature kendt fra C++ (siden sidst i 1980'erne) og
i Java (siden version 1.5 som blev releaset i 2004).

Den meste kendte anvendelse er type safe collections.

Hvor man i .NET 1.1 enten lavede en helt custom wrapper class eller
extendede den dertil beregnede CollectionBase class.

TypSafCol11.cs

```
using System;
using System.Collections;

public class MyData
{
    private string s;
    public MyData() : this("")
    {
    }
    public MyData(string s)
    {
        this.s = s;
    }
    public override string ToString()
    {
        return s;
    }
    public string S
    {
        get
        {
```

```
            return s;
        }
        set
        {
            s = value;
        }
    }
}

public class MyCollection : CollectionBase
{
    public MyData this[int index]
    {
        get
        {
           return (MyData)List[index];
        }
    }
    public int Add(MyData value)
    {
        return List.Add(value);
    }
}

class TypSafCol11
{
    public static void Main(string[] args)
    {
        MyCollection mycol = new MyCollection();
        mycol.Add(new MyData("A"));
        mycol.Add(new MyData("BB"));
        mycol.Add(new MyData("CCC"));
        for(int i = 0; i < mycol.Count; i++)
        {
            Console.WriteLine(mycol[i]);
        }
    }
}
```

Så kan man i .NET 2.0 bruge generics til at lave det samme med.

TypSafCol2.cs

```
using System;
using System.Collections.Generic;

public class MyData
{
    private string s;
    public MyData() : this("")
    {
```

```
    }
    public MyData(string s)
    {
        this.s = s;
    }
    public override string ToString()
    {
        return s;
    }
    public string S
    {
        get
        {
            return s;
        }
        set
        {
            s = value;
        }
    }
}

class TypSafCol2
{
    public static void Main(string[] args)
    {
        List<MyData> mycol = new List<MyData>();
        mycol.Add(new MyData("A"));
        mycol.Add(new MyData("BB"));
        mycol.Add(new MyData("CCC"));
        for(int i = 0; i < mycol.Count; i++)
        {
            Console.WriteLine(mycol[i]);
        }
    }
}
```

Men generics kan bruges til andet end type safe collections.

Her er et lille eksempel.

Generic2.cs

```
using System;

public class MyComparer<T> where T : IComparable
{
    public static T Max(T a, T b)
    {
        if(a.CompareTo(b) > 0)
        {
```

```
            return a;
        }
        else
        {
            return b;
        }
    }
    public static T Min(T a, T b)
    {
        if(a.CompareTo(b) < 0)
        {
            return a;
        }
        else
        {
            return b;
        }
    }
}

public class Test
{
    public static void Main(string[] args)
    {
        Console.WriteLine(MyComparer<int>.Max(11,2));
        Console.WriteLine(MyComparer<int>.Min(11,2));
        Console.WriteLine(MyComparer<string>.Max("11","2"));
        Console.WriteLine(MyComparer<string>.Min("11","2"));
    }
}
```

En meget nyttig feature.

Andet

Resten af de nye features må du selv slå op i docs.

Som appetitvækker nævner jeg lige:
- ASP.NET master pages
- C# nullable types
- C# anonyme metoder
- C# static classes
- C# seperat access på get og set for properties

God kode lyst.

.NET 3.5 og C# 3.0

Se også artikel:
http://www.eksperten.dk/guide/1153
omkring .NET 3.5 og C# 3.0!

Kommentar af frankeman d. 09. maj 2005 | 1

Udmærket artikel, ganske konkret.

Kommentar af skwat d. 31. maj 2005 | 2

Kommentar af dr_chaos d. 30. maj 2005 | 3

Kommentar af websam d. 03. mar 2006 | 4

Alletiders artikel, den dækker ikke alt men den giver da inspiration til videre tankegang ;o)

Kommentar af nielsbrinch d. 10. maj 2005 | 5

Lækkert med en simpel oversigt over de mest interessante ændringer.

Kommentar af digitalsoul d. 01. nov 2005 | 6

Giver et rigtigt godt overblik over nogen af de nye ting der er i .net 2.0 :)

Kunne være rart med en ligende artikel over nye ting i asp.net 2.0

Kommentar af webcreator d. 29. maj 2005 | 7

Mange tak for en fin oversigt over de mest interessante udvidelser i C# .NET 2.0.
Specielt WebServer og FTP var meget interessant. Gode kodeeksempler medfølger - fint :)

Kommentar af davidfossil d. 01. jun 2005 | 8

Kommentar af brummelum d. 24. maj 2005 | 9

Ikke så meget uden-om-snak, fine eksempler

Kommentar af lifo d. 07. jul 2005 | 10

Artiklen er ok
MEN (der er altid et men)
som du selv siger "Det er ikke en komplet oversigt" der vil jeg sige langt fra
der er et hav af andre spændende nye ting.
Nu da release datoen for 2.0 er kendt så skulle du måske opdatere det punkt

Kommentar af phvass d. 11. maj 2005 | 11

Kommentar af anthonsen d. 01. aug 2006 | 12

Kommentar af over-load d. 13. maj 2005 | 13

Ikke så vildt forklarende vil jeg sige - lever ikke op min generelle Arne_v score!

Kommentar af mmbn d. 09. jan 2006 | 14

God informativ artikel

Kommentar af visualdeveloper d. 02. okt 2005 | 15

God artikel...Igamle dage tog det år at lave farver i et konsolvindue men det kan jeg se at net 2.0 har lavet om på !

Kommentar af jimgordon d. 23. maj 2005 | 16

Super artikel. Syntes sjovt nok bedre om den gamle måde med en factory method pattern til at hente en ADO connection. Der kan man jo se hvad der sker, plus lave sine egne connections.

Kommentar af claus_joergensen d. 08. maj 2005 | 17

Viser en masse features, også værd at læse for nybegyndere. Til webdelen kan også siges at ASP.NET 2 endelig undersøtter templates ordenligt. Gælder mig til v.final