Zip filer med .NET

Jeg præsentere nogle muligheder for pakning/udpakning af filer i .NET, så man nemt kan komme i gang med det:)
(primært komprimerede filer, men også ét eksempel på zip filer!!)

Skrevet den **02. Feb 2009** af **kalp** I kategorien **Programmering / .NET** | ★★☆☆☆

Kodestumper bør kopires over i et udviklingsværktøj, som Visual Studio, så det er til, at læse!

Denne artikel er skrevet på et mildt overfladisk niveau og kræver derfor af læseren:

1. Erfaring indenfor programmering med .NET (C#, VB)
2. Eller et lærenemt hoved
3. Eller en copy & paste ekspert :)

Du kan i mange sammenhænge få behov for, at skulle komprimere data.
Derfor har jeg skrevet denne artikel, så du nemt kan komme i gang med dette.
Der er mange kilder om emnet på engelsk, men denne er på dansk til dem, som ikke er så skrape til det engelske sprog.
Der er i .NET 2 klasser, som man kan komprimere og dekomprimere data med.

GZipStream
DeflateStream

De kan findes i namespace'et:
System.IO.Compression

Begge har samme begrænsninger.
Hvad er så forskellen? GZipStream kan udpakke filer der er pakket med værktøjet "gzip".
Den kan selvfølgelig også pakke en fil, som ligeledes kan udpakkes af værktøjet gzip.
Hvis pakning og udpakning af data sker i det system man selv arbejder med, så bør man benytte DeflateStream.
Den giver også en mindre fil når man pakker med DeflateStream end hvad GZipStream giver.
Det er ikke fordi DeflateStream pakker bedre (de benytter samme algoritme).
Det skyldes, at GZipStream tillader en "Header", som kan have ekstra information der nyttigt når der skal udpakkes.
Da vi taler om de 2 klasser benytter samme algoritme, så er det også bagateller man sparer når man pakker med DeflateStream.
Jeg mener dog stadig man bør anvende den hvis man alligevel ikke skal benytte sine pakkede filer uden for sit system.

De 2 største begrænsninger disse klasse besider er:

1. De kan kun pakke én fil og dermed kun udpakke én fil.
2. De kan kun pakke data, som ikke er størrer end 4GB (men det bør vidst også være nok for de fleste).

Eftersom klasserne fungerer på samme måde vil jeg vise hvorledes man pakker og udpakker filer med klassen DeflateStream.
Du er dog velkommen til, at benytte GZipStream i din kode.

Vi får behov for 2 FileStream objekter.
Den første er kilden vi vil pakke og den anden er destinationen/navnet på filen efter den er blevet pakket.
Den første linje er kilden man vil pakke.
Den anden linje er placeringen og navnet den pakkede fil skal have.

```
FileStream kilde    = File.OpenRead(@"c:\enfil.txt");
FileStream pakketFil = File.Create(@"c:\pakket.zip");
```

Herefter kan man oprette DeflateStream objektet.

```
DeflateStream zipper = new DeflateStream(pakketFil, CompressionMode.Compress);
```

Første parameter i DeflateStream konstruktøren er vores FileStream fra før.
Anden parameter er en enum struktur, som giver 2 valgmuligheder. Enten Compress eller Decompress.
Det vi skal benytte indtil videre er Compress og derfor vælges denne.
Dette er det eneste der skal til for, at pakke en fil.
Vores DeflatedStream er klar til, at tage imod én byte af gangen, så vi til sidst ender op med en komprimeret fil.
Koden til denne sidste bid kan se ud som følgende.

```
int enByte = kilde.ReadByte();  // indlæs første byte fra fil
while(enByte != -1){  //  hvis enByte er -1 har vi nået sidste byte i filen
   zipper.WriteByte((byte)enByte);  // type cast til byte fra int
   enByte = kilde.ReadByte();  // læs næste byte i filen
}
```

Samlet er koden:

```
FileStream kilde    = File.OpenRead(@"c:\enfil.txt");
FileStream pakketFil = File.Create(@"c:\pakket.zip");

DeflateStream zipper = new DeflateStream(pakketFil, CompressionMode.Compress);

int enByte = kilde.ReadByte();  // indlæs første byte fra fil
while(enByte != -1){  //  hvis enByte er -1 har vi nået sidste byte i filen
   zipper.WriteByte((byte)enByte);  // type cast til byte fra int
   enByte = kilde.ReadByte();  // læs næste byte i filen
}
```

Når man kan pakke en fil vil man sikkert også gerne kunne udpakke den igen.
Det kan gøres på følgende måde.
Vi skal igen benytte 2 FileStream objekter.

```
```

```
FileStream kilde      = File.OpenRead(@"c:\pakket.zip ");
FileStream udPakketFil = File.Create(@"c:\enfil.txt");
```

Denne gang er kilden en pakket fil.
Før var det filen der skulle pakkes.
Herefter kan man oprette DeflateStream objektet.

```
DeflateStream unZipper = new DeflateStream(kilde, CompressionMode.Decompress);
```

Og CompressionMode er selvfølgelig sat til Decompress.
Selve udpakningen sker sådan her.

```
int enByte = unZipper.ReadByte();
while(enByte != -1){
   udPakketFil.WriteByte((byte)enByte);
   enByte = unZipper.ReadByte();
}
```

samlet er koden:

```
FileStream kilde      = File.OpenRead(@"c:\pakket.zip ");
FileStream udPakketFil = File.Create(@"c:\enfil.txt");

DeflateStream unZipper = new DeflateStream(kilde, CompressionMode.Decompress);

int enByte = unZipper.ReadByte();
while(enByte != -1){
   udPakketFil.WriteByte((byte)enByte);
   enByte = unZipper.ReadByte();
}
```

Disse .NET klasser fungere egentlig udmærket, men hvad hvis man gerne vil pakke/udpakke flere filer?
Der kan man hente SharpZipLib fra http://www.icsharpcode.com/OpenSource/SharpZipLib/Download.aspx
Når man har refereret dll'en i sit projekt er det meget samme fremgangsmåde, som det allerede præsenterede kode.
SharpZipLib er gratis!

Jeg viser et eksempel på udpakning af filer med SharpZibLib og vil mene man bør være i stand, at pakke filer med det efterfølgende.

```
ZipInputStream zippedFile = new ZipInputStream(File.OpenRead(@"c:\pakket.zip ")); //filer der skal udpakkes.

ZipEntry theEntry = zippedFile.GetNextEntry(); //finder først fil I arkivet.
try {
```

```
while (theEntry != null) {
      FileStream currentFile =
      File.Create(File.Create(string.Format(@"c:\{0}.txt"), theEntry.Name));
         try  {
           byte[] length = new byte[2048];
             while (true)  {
               int size = zippedFile.Read(length, 0, length.Length);
               if (size > 0) {
                 currentFile.Write(length, 0, size);
               } else break;
             }
           }
        finally  {
          currentFile.Close();
        }
        theEntry = zippedFile.GetNextEntry();
      }
      }
      finally {
        zippedFile.Close();
      }
```

Jeg har lavet mine eksempler med txt filer, men det er for, at illustrere det.
Jeg håber du via. Lidt copy & paste og lidt roderi med det kan få pakket og udpakket dine filer med success;)
Mangler du svar på noget, så kan du spørge på Eksperten eller se hvad en søgning eller to giver dig på google:)

Kommentar af casualty d. 15. Apr 2008 | 1

Tak for det...

Kommentar af mysitesolution d. 18. Apr 2008 | 2

Gør det klart inden man køber artiklen om vi snakker .Zip filer eller komprimerede filer.

Kommentar af arne_v d. 04. May 2008 | 3

OK.

Der er et par småting:
- det er ikke pænt at kalde en pakket fil for .zip hvis det ikke er en ZIP fil
- deflate eksemplerne får ikke kaldt Close
- jeg tror der er en File.Create for meget i #ZipLib eksemplet

Og så er der et stort problem, men det fremgår ikke af dokumentationen, så hvis ikke man ved det, så har man ikke mange chancer for at finde ud af det. Deflate eksemplerne bruger enkelt byte læsning. Det er hurtigere med en stor buffer, men det er kun et performance spørgsmål. Værre er det at DeflateStream komprimerer meget dårligt med små buffere. Et lille eksempel: original 6.594.469 bytes, komprimeret med enkelt byte læsning og skrivning 6.418.407 bytes, komprimering med 100000 byte buffer for læsning og skrivning 726.475 !