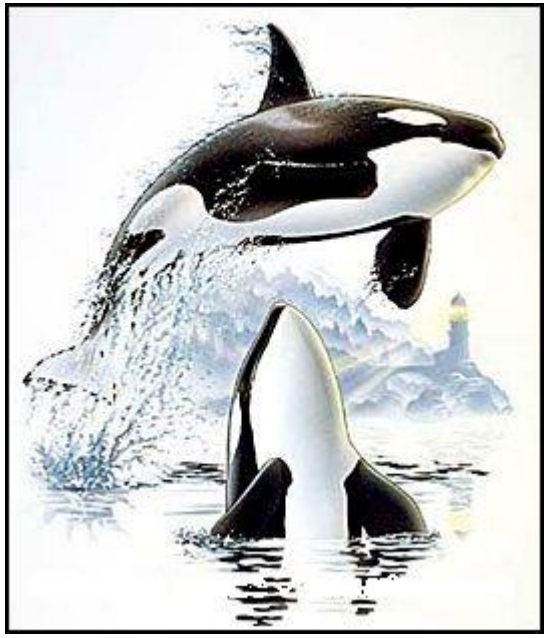
The Missing LINQ

- og relaterede features

VB version

Henrik Lykke Nielsen / Captator
www.captator.dk
Softwarearkitekt, Microsoft Regional Director for Denmark
lykke@captator.dk
+45 2237 3311
twitter.com/dothenrik
dk.linkedin.com/in/henriklykkenielsen

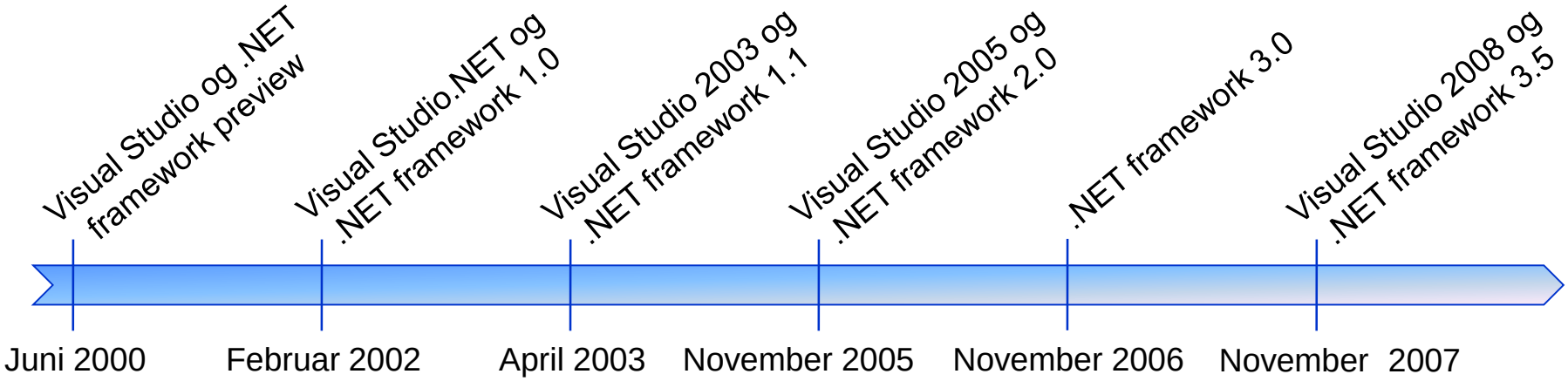
Agenda

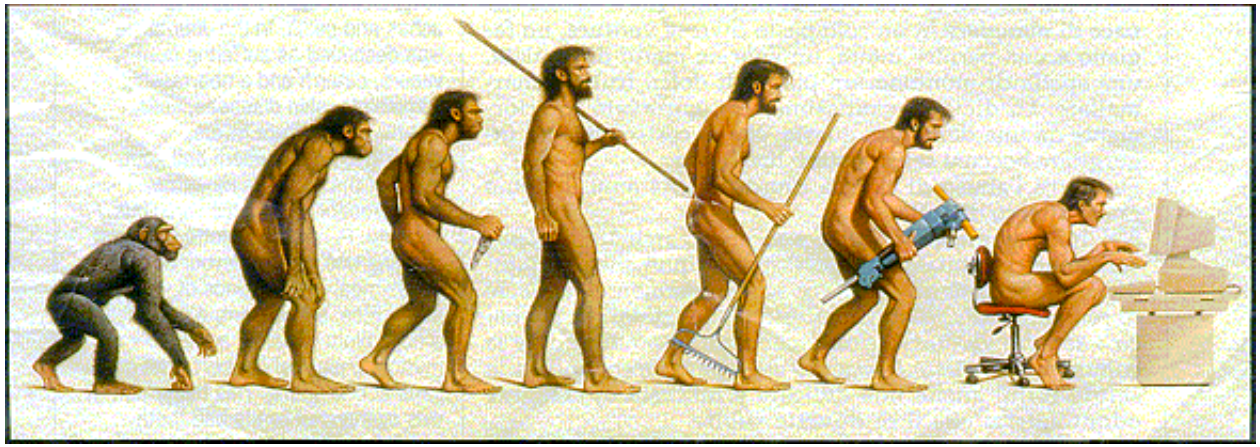


- **Sprog features**
  - Objekt og collection initializers
  - Implicit typede variable og arrays
  - Anonyme typer
  - Lambda udtryk
  - Extension metoder
- **The Missing LINQ**
  - Standard query operatorer, query expressions
  - LINQ to Objects
  - LINQ to DataSets
  - LINQ to databaser
  - LINQ to XML

Visual Studio og .NET frameworket

- **En platform i udvikling**





Objekt initializers

Captator

- **Kan direkte sætte properties og fields ved instansiering**

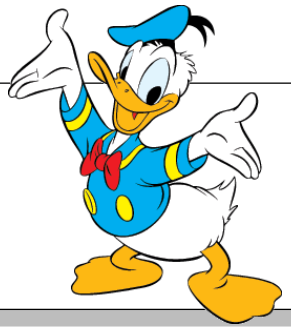
```
Class Person
  Public Navn As String
  Public CprNr As String
  Public Property Adresse As String
End Class
```

- **Klassen skal i dette første eksempel have en public, parameterløs konstruktør (paranteserne kan udelades)**

```
Dim p As New Person With { .Navn="Anders", .Adresse="Andeby" }
```

- **Objekt initializeren svarer til:**

```
Dim temp As New Person()
temp.Navn = "Anders"
temp.Adresse = "Andeby"
p = temp
```



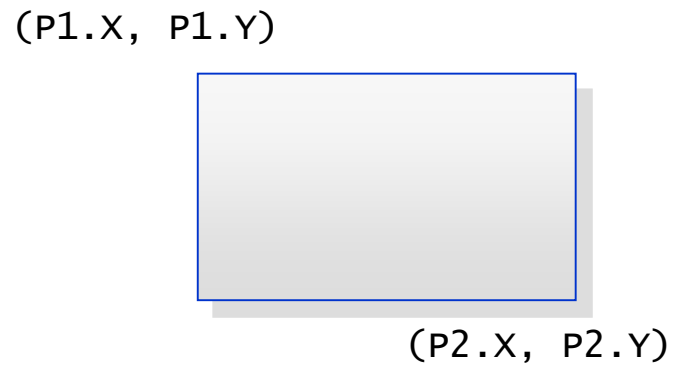
- **Kan kombineres med konstruktørparametre**

```
Dim p2 As New Person2("Andersine") With {.Adresse = "Andeby"}
```

Objekt initializers

- **Objekt initializers kan nestes**

```
Class Point
  Public X As Integer
  Public X As Integer
End Class

Class Rectangle
  Public P1 As Integer
  Public P1 As Integer
End Class
```




```
Dim a As New Point With { .X=7, .Y=42 }
Dim b As New Point With { .X=17, .Y=117 }
Dim r1 As New Rectangle With { .P1=a, .P2=b }
Dim right1X As Integer = r1.P2.X

Dim r2 As New Rectangle With
  {.P1 = New Point With { .X=10, .Y=100 }, _
   .P2 = New Point With { .X=20, .Y=200 }}
Dim right2X As Integer = r2.P2.X
```

Collection initializers

Captator

- **Allerede eksisterende array initialisering**

```
Dim primesArray As Integer() = {2, 3, 5, 7, 11, 13, 17}
```

- **Collectionen skal implementere System.Collections.Generic.ICollection<T>**
  - Kalder Add(T)

```
Dim primtal As New List(Of Integer) From {2, 3, 5, 7, 11, 13, 17}

Dim ænder As New List(Of Person) From {
  New Person() With {.Navn = "Anders And", .Adresse = "Andeby"},
  New Person() With {.Navn = "Andersine And", .Adresse = "Andeby"},
  New Person() With {.Navn = "Fætter Vims", .Adresse = "Andeby"}}

Dim primesArrayList As New ArrayList From {2, 3, 5, 7, 11, 13, 17}

Dim værdier = New Dictionary(Of Integer, String) From
  {{0, "Første"}, {1, "Anden"}}
```

Argumenter til Add-metoden

Implicitly typed local variables

Captator

- **Typen kan udledes af kompileren**

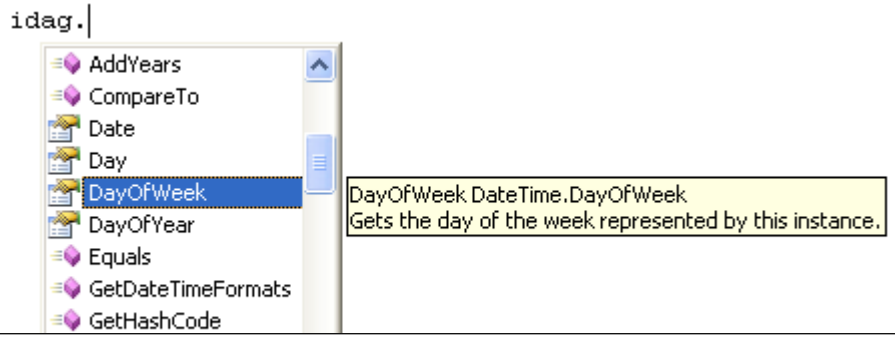
```
var pi = 3.14;
var navn = "Anders And";
var idag = System.DateTime.Today;

var tal = new int[] {10, 4, 17, 42};
var duck = new Person {Navn="Anders", Adresse="Andeby", Alder=71 };
```

Double

String

DateTime

- **Statisk typet!**
- **Fuld intellisense**
- **Kompilercheck**




```
System.Collections.Generic.List<Person> personer =
    new System.Collections.Generic.List<Person>();

var personer = new System.Collections.Generic.List<Person>();
```

Implicitly typed local variables

Captator

- **Typen kan udledes af kompileren**

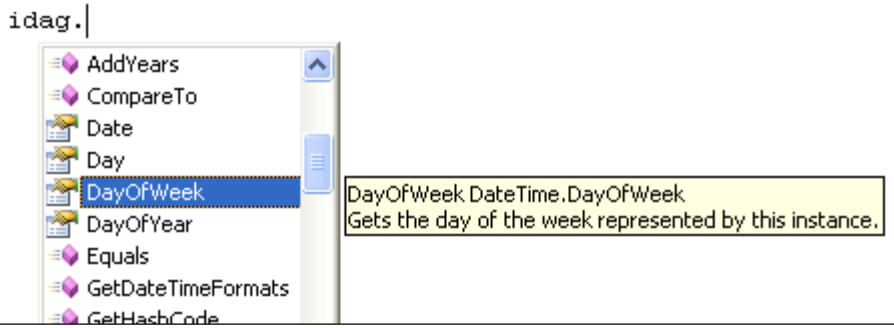
```
Dim pi = 3.14
Dim navn = "Anders And"
Dim idag = System.DateTime.Today

Dim tal = New Integer() {10, 4, 17, 42}
Dim duck As
    New Person With {.Navn="Anders", .Adresse="Andeby", .Alder=71}
```

double

string

DateTime

- **Statisk typet!**
- **Fuld intellisense**
- **Kompilercheck**




```
Dim personer As System.Collections.Generic.List(Of Person) = _
    New System.Collections.Generic.List(Of Person)()
```

Har man hele tiden kunnet i VB

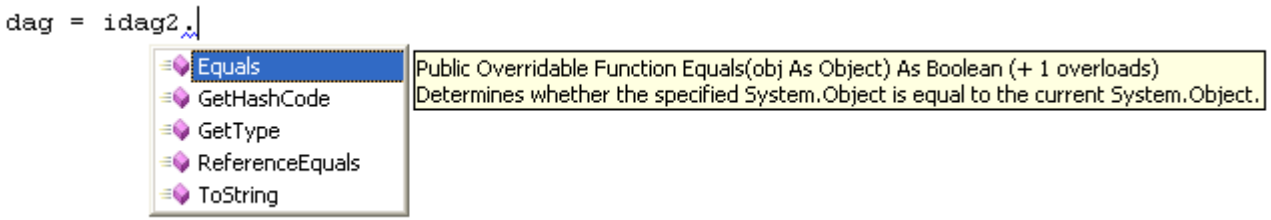
```
Dim personer As New System.Collections.Generic.List(Of Person)()
```

Implicitly typed local variables ⇔ late binding

- Type inferens er <u>ikke</u> det samme som late binding
- Late binding:

```
Option Strict Off
Dim idag2 As Object = System.DateTime.Today
dag = idag2.Day
```

- Meget fleksibelt
- Ikke typestærkt
- Ingen kompilercheck
- (Næsten) ingen intellisense



Implicitly typed arrays

Captator

- **Arrayets type fastlægges ud fra højre siden**

```
Dim a = New Integer() {2, 4, 17, 42}
Dim b = New Double() {1, 1.5, 2, 2.5}
Dim c = New String() {"hello", Nothing, "world”}
Dim d = New Object() {1, "one", 2, "two"}
```

- **Alle elementer skal implicit kunne castes til typen**
- **Kan kombineres med fx anonyme objekt initializers**

```
Dim anonymeÆnder = New Person() _
  {New Person With {.Navn = "Anders And", .Adresse = "Andeby"}, _
   New Person With {.Navn = "Andersine And", .Adresse = "Andeby"}, _
   New Person With {.Navn = "Fætter Vims", .Adresse = "Andeby"}}

int andTal = anonymeÆnder.Length;
string andeNavn = anonymeÆnder[1].Navn;
```

Implicitly typed arrays

- **Kan udlede for-variables type**

```
For Each x In New Integer() { 2, 4, 17, 42 }
  '...
Next
```

```
Dim a = New Integer() {2, 4, 17, 42}

'...

For Each x In a
  '...
Next
```

Anonyme typer

Captator

- **En type behøver ikke at have noget (kendt) navn**

```
Dim b1 = New With { .Titel=".NET for hackere", .Pris = 17.42 }
Dim b2 = New With { .Titel=".NET for plattenslagere", .Pris = 42.17}

b2 = b1
Dim b3 = b2
Dim bogensTitel As String = b1.Titel

MessageBox.Show("Første bogs titel = " + b1.Titel)
MessageBox.Show("Tredje bogs titel = " + b3.Titel)
```

```
Dim p As New Person() With
 { .Navn="Anders",.Adresse="Andeby",.Alder=71,.CprNr="123456-7890" }

Dim addressInfo1 = New With { p.Navn, p.Adresse }
Dim addressInfo2 = New With { .Navn = p.Navn, .Adresse = p.Adresse }
Dim addressInfo3 = New With { .Name = p.Navn, .Address = p.Adresse }

addressInfo2 = addressInfo1

addressInfo3 = addressInfo2
```

✘ Compiler fejl

Anonyme typer – casting by example

Captator

- **Følgende funktion returnerer en anonym type**

```
Public Function ReturnAnonymous() As Object
  Dim anonymTypeInstans = New With {.Navn = "Anders And", .Adresse = "Andeby"}
  Return anonymTypeInstans
End Function
```

- **Inferens af generiske typer gør, at ved kald af "Cast" kan typen T fastlægges ud fra parameteren "type"**

```
Public Function Cast(Of T)(ByVal obj As Object, ByVal type As T) As T
  Return CType(obj, T)
End Function
```

- **Parameteren "type" til "Cast" metoden angives som et eksempel på den anonyme type, der castes til**

```
Dim obj As Object = ReturnAnonymous()
Dim typed = Cast(obj, New With {.Navn = "", .Adresse = ""})

MessageBox.Show(typed.Navn + " - " + typed.Adresse, "Data fra funktion")
```

Anonyme typer – casting by example

- ”Option Strict Off” tillader assignment mellem ikke-lokale anonyme typer

```
Dim b1 = New With {.Titel=".NET for hackere", .Pris = 17.42}

Util.AnonymTypeModtager(b1)
```

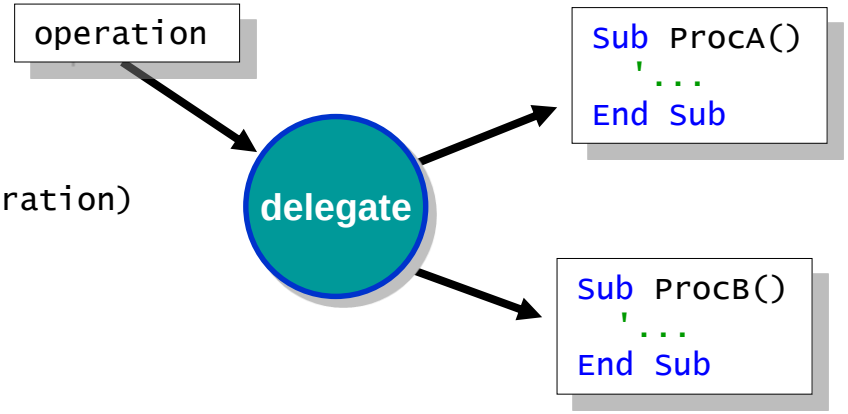
```
Option Strict Off

Public Class Util

  Public Shared Sub AnonymTypeModtager(ByVal obj As Object)

    Dim b2 = New With {.Titel = "Dummy", .Pris = 42.17}

    b2 = obj       ' Option Strict On disallows implicit conversions
                   ' from 'Object' to '<anonymous type>
  End Sub

End Class
```

- Casting by example virker kun indenfor samme assembly – muligheden forsvinder muligvis!

Delegates

```
Private Delegate Function MathOperation(ByVal a As Integer, ByVal b As Integer) As Integer

Private Function Add(ByVal a As Integer, ByVal b As Integer) As Integer
  Return a + b
End Function

Private Function Subtract(ByVal a As Integer, ByVal b As Integer) As Integer
  Return a - b
End Function

Private Function Multiply(ByVal a As Integer, ByVal b As Integer) As Integer
  Return a * b
End Function

Private Sub Calculate(ByVal operation As MathOperation, ByVal a As Integer, _
    ByVal b As Integer)
  Dim result As Integer = operation(a, b)
  ' ...
End Sub
```

```
Dim operation As MathOperation

If radAdd.Checked Then
  operation = New MathOperation(AddressOf Add)
ElseIf radSubtract.Checked Then
  operation = CType(AddressOf Subtract, MathOperation)
ElseIf radMultiply.Checked Then
  operation = AddressOf Multiply
End If

Calculate(operation, 42, 17)
```



Lamba udtryk

- **Komprimerede anonyme metoder**

```
Delegate Function MinDelegateType(ByVal a As Integer) As Integer

Sub ShowResult(ByVal del As MinDelegateType, ByVal i As Integer)
  MessageBox.Show(del(i).ToString())
End Sub
```

```
Dim delCalc As MyDelegateType = Function(x As Integer) x + 42

ShowResult(delCalc, 17)

ShowResult(Function(x As Integer) x + 42, 18)

ShowResult(Function(x) x + 42, 20)
```

x er eksplicit typet

x er implicit typet

- **Parametre til lambda udtryk kan være eksplicit eller implicit typede**

Lamba udtryk

- **Predefinerede generiske delegate typer: Func**
- **Ligger i System namespacet (System.Core.dll)**

```
Delegate Function Func(Of TResult)() As TResult
Delegate Function Func(Of T, TResult)(ByVal arg As T) As TResult
Delegate Function Func(Of T1, T2, TResult)(ByVal arg1 As T1, ByVal arg2 As T2) As TResult
Delegate Function Func(Of T1, T2, T3, TResult)
  (ByVal arg1 As T1, ByVal arg2 As T2, ByVal arg3 As T3) As TResult
Delegate Function Func(Of T1, T2, T3, T4, TResult)
  (ByVal arg1 As T1, ByVal arg2 As T2 , ByVal arg3 As T3, ByVal arg3 As T4) As TResult
```

- **Eksempel – f2 svarer til f1:**

```
Private Function NavnGivetFunktion(ByVal x As Integer) As Integer
  Return x + 42
End Function
```

```
Dim f1 As System.Func(Of Integer, Integer) = AddressOf NavnGivetFunktion

Dim f2 As System.Func(Of Integer, Integer) = Function(x) x + 42
```

- **Action delegate typerne svarer til Func blot "void"**

Lamba udtryk

- **Predefineret generisk predikats-funktion**

```
Delegate Function Predicate(Of T)(ByVal obj As T) As Boolean
```

- **Eksempel:**

```
Dim selector As System.Predicate(Of Integer) = Function(x) x > 0

If selector(42) Then
  MessageBox.Show("Tallet kommer med.")
Else
  MessageBox.Show("Tallet kommer ikke med.")
End If
```

- **Expression trees kan repræsentere lambda udtryk som data**

Extension metoder

Captator

- **Kan udvide udvalgte typer med ekstra metoder**
- **Lav en extension metode**
  - Statisk metode i statisk klasse (C#), metode i module (VB)
  - Det første argument til metoden udpeger typen, som extension metode virker på
- **Importer namespacet (using (C#) / Imports (VB))**
  - Extension metoderne i det importerede namespace kan nu kaldes
- **Metoden kaldes som en instansmetode**
  - Kan naturligvis også kaldes via typen som normalt
- **Kan være almindelige såvel som generiske metoder**
- **Instansmetoder ”vinder over” extension metoder**

Extension metoder - eksempel

```
Namespace SuperExtensions
  Public Module UserInterface
    <System.Runtime.CompilerServices.Extension()> _
    Public Sub Show(ByVal s As String)
      System.Windows.Forms.MessageBox.Show(s)
    End Sub
  End Module
End Namespace
```

- **I VB angiver System.Runtime.CompilerServices.Extension-attributten, at dette er en extension metode**

```
Imports SuperExtensions
```

```
SuperExtensions.UserInterface.Show("Hej statiske verden");

Dim str As String = "Hej verden"
str.Show()
```

Et problem … der måske eksisterer

```
Dim sql As String = "SELECT * FROM Authors"

Dim cmd As New System.Data.SqlClient.SqlCommand(sql, conn)
Dim adap As New System.Data.SqlClient.SqlDataAdapter(cmd)
Dim tbl As New System.Data.DataTable
adap.Fill(tbl)

Dim navn As String = CStr(tbl.Rows(0)("Name"))
```

- **Query, parametre og resultat er ikke stærkt typet**
  - Queryen er kun en streng
  - Parametre er svagt typede
  - Resultatet (f.eks. DataTable) er en collection af svagt typede objekter
- **Query sproget er typisk koblet til databaser**
  - Ikke-trivielt at lave querysprog

LINQ arkitektur

C#

VB

Andre sprog …

.NET Language Integrated Query (LINQ)

Data sources tilgængelige fra LINQ

LINQ enabled ADO.NET

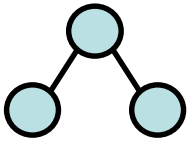
LINQ to Objects

LINQ to DataSets

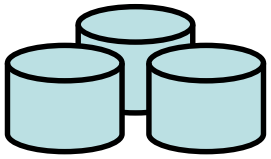
LINQ to SQL

LINQ to Entities

LINQ to XML



Objekter

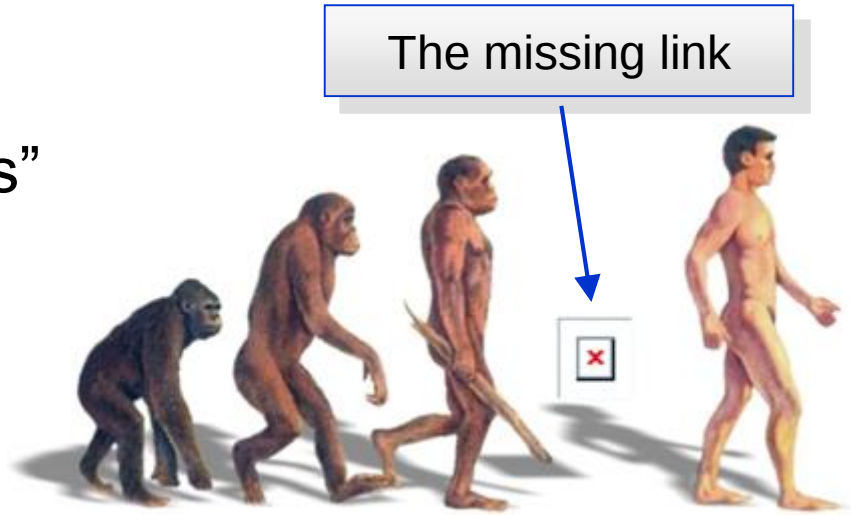


Relationel

XML

The Missing LINQ

- **LINQ danner bro mellem .NET sprog og ”LINQ to *”**
- **Generel query facilitet i .NET frameworket**
  - Definerer en række standard query operatorer
    - Gennemløb, filtrering, projektion, ...
  - Et API til at query ethvert .NET array eller collection
  - Opererer på og returnerer IEnumerable<T> (C#) / IEnumerable(Of T) (VB) objekter (inklusive arrays)
  - Metoderne eksekveres først ved gennemløb
  - De fleste metoder kan ”pipes”
  - Koden kan modulariseres
- **Deklarativ model**



Standard operatorer

- **Extension metoder defineret i System.Linq.Enumerable klassen**
- **De nuværende standard operatorer:**
  - Aggregate, All, Any, AsEnumerable, Average, Cast, Concat, Contains, Count, DefaultIfEmpty, Distinct, ElementAt, ElementAtOrDefault, Empty, Except, First, FirstOrDefault, GroupBy, GroupJoin, Intersect, Join, Last, LastOrDefault, LongCount, Max, Min, OfType, OrderBy OrderByDescending, Range, Repeat, Reverse, Select, SelectMany, SequenceEqual, Single, SingleOrDefault, Skip, SkipWhile, Sum, Take, TakeWhile, ThenBy, ThenByDescending, ToArray, ToDictionary, ToList, ToLookup, Union, Where

Standard query operatorer - Where

Captator

- **Filtrerer en sekvens udfra et predikat**

Extension metode

```
Public Shared Function Where(Of TSource) _
    (ByVal source As IEnumerable(Of TSource), ByVal predicate As _
    Func(Of TSource, Boolean)) As IEnumerable(Of TSource)
```

```
Dim ænder As New List(Of Person)
ænder.Add(New Person With {.Navn = "Fætter Vims", .Adresse = "Andeby"})
ænder.Add(New Person With {.Navn = "Andersine And", .Adresse = "Andeby"})
ænder.Add(New Person With {.Navn = "Anders And", .Adresse = "Andeby"})

Dim res1 = System.Linq.Enumerable.Where(ænder, _
    Function(obj) obj.Navn.StartsWith("Anders"))

For Each p As Person In res1
  MessageBox.Show(p.Navn, "Kaldt som statisk metode")
Next

Dim res2 = ænder.Where(Function(obj) obj.Navn.StartsWith("Anders"))

For Each p As Person In res2
  MessageBox.Show(p.Navn, "Kaldt som instans metode")
Next
```

LINQ query expressions

Captator

```
Dim ænder As New List(Of Person)
ænder.Add(New Person With {.Navn = "Fætter Vims", .Adresse = ... })
ænder.Add(New Person With {.Navn = "Anders And", .Adresse = ... })
```

- **Sprogunderstøttelse for query udtryk**

```
Dim andeborgere = From duck In ænder _
                  Where duck.Adresse = "Andeby" _
                  Order By duck.Alder, duck.Navn _
                  Select New With {duck.Alder, duck.Navn}
```

Eksekveres som

```
Dim andeborgere = ænder.Where(Function(duck) duck.Adresse = "Andeby") _
  .OrderBy(Function(duck) duck.Alder) _
  .ThenBy(Function(duck) duck.Navn) _
  .Select(Function(duck) New With {duck.Alder, duck.Navn})
```

```
For Each duck In andeborgere
  MessageBox.Show(duck.Navn + "  " + duck.Alder.ToString())
Next
```

- **Fleksible projektioner**

```
Select New With {duck.Navn, .Født = System.DateTime.Now.Year - duck.Alder}
```

LINQ query operatorer

- **Clauses indbygget i VB (keywords)**
  - From, Select, Where, Order By, Join, Group By, Group Join, Aggregate, Let, Distinct, Skip, Skip While, Take, Take While
  - En query skal begynde med enten en *From* eller en *Aggregate* clause
- **Clauses indbygget i C# (keywords)**
  - from, where, select, group, into, orderby, join, let
  - En query skal begynde med en *From* clause
- **Øvrige LINQ operatorer kan kaldes som extension metoder (placeret I klasserne Enumerable og Queryable i System.Linq namespacet)**

LINQ query expressions

Captator

- **Når man definerer en LINQ query opbygges et expression tree**
- **Deferred (udskudt) eksekvering**
  - Normalt evalueres queryen først, når elementerne i resultatet tilgås (i eksempelvis en For Each-løkke)
  - Gør det muligt at kombinere queries
- **Immediate (øjeblikkelig) eksekvering**
  - Metoderne Count, Sum, Any etc. returnerer beregnet værdi
  - Metoderne First, Last, Single etc. returnerer et enkelt element
  - Metoderne ToList, ToArray etc. returnerer mængder - kan bruges til caching

LINQ to Objects

Captator

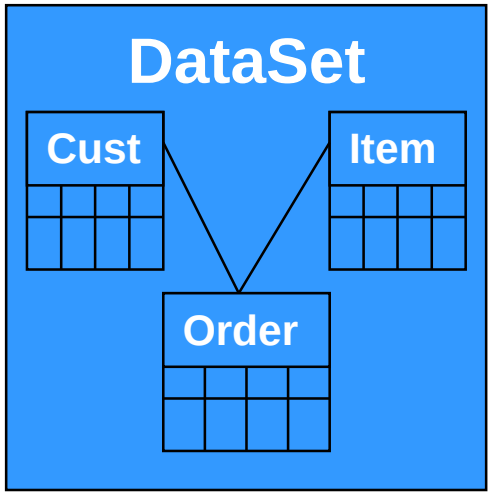
```
Dim dirPaths As String() = Directory.GetDirectories("C:\CaptatorVss\Eifos2")

Dim foundDirs = From dirPath In dirPaths _
                Let dir = New System.IO.DirectoryInfo(dirPath) _
                Let fileCount = dir.GetDirectories().Count() _
                Where dir.Name.StartsWith("C") And fileCount > 3 _
                Select dir

For Each directory In foundDirs
  txtData.Text += directory.Name + vbCrLf
Next
```

```
Dim dirPaths As String() = Directory.GetDirectories("C:\CaptatorVss\Eifos2")

Dim foundDirs1 = From dirPath In dirPaths _
                 Let dir = New System.IO.DirectoryInfo(dirPath) _
                 Where dir.Name.StartsWith("C") _
                 Select dir

Dim foundDirs2 = From dir In foundDirs1 _
                 Let fileCount = dir.GetDirectories().Count() _
                 Where fileCount > 3 _
                 Select dir

For Each directory In foundDirs2
  txtData.Text += directory.Name + vbCrLf
Next
```

Queries kan kombineres, hvilket letter genbrug mærkbart

LINQ to DataSet

- DataSets er disconnectede datastrukturer
- Har en struktur der minder om relationelle databaser
- Standard LINQ syntax understøttelse
- Filtrering, projektion, joins
- Kan kombineres med andre in-memory datastrukturer
- Understøtter aggregering



LINQ to DataSet

Captator

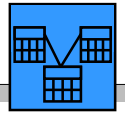
- **DataSets indeholder svagt typede data**
- **De svagt typede data skal gøres typestærke**
  - Kald System.Data.DataTableExtensions.AsEnumerable extension-metoden på DataTable-objekter

```
<ExtensionAttribute> _
Public Shared Function AsEnumerable (source As DataTable) As _
     System.Data.EnumerableRowCollection(Of DataRow)
```

  - System.Data.DataRowExtensions.Field extension-metoderne (en del overloads) giver typestærk tilgang til de enkelte felter

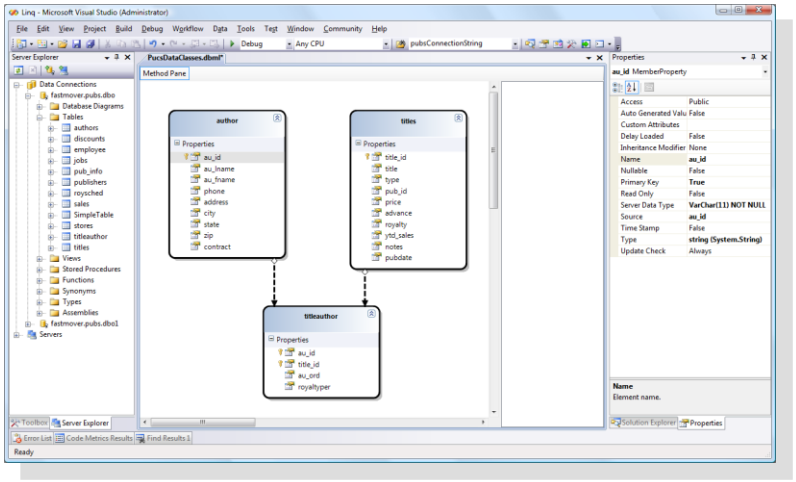
```
<ExtensionAttribute> _
Public Shared Function Field(Of T) (row As DataRow, columnName As String) As T
```

```
Dim kundeNavne = _
    From kunde In _ds.Tables("Customer").AsEnumerable() _
    Where kunde.Field(Of String)("City") = "London" _
    Select New With {.CompanyName = kunde.Field(Of String)("CompanyName")}
```



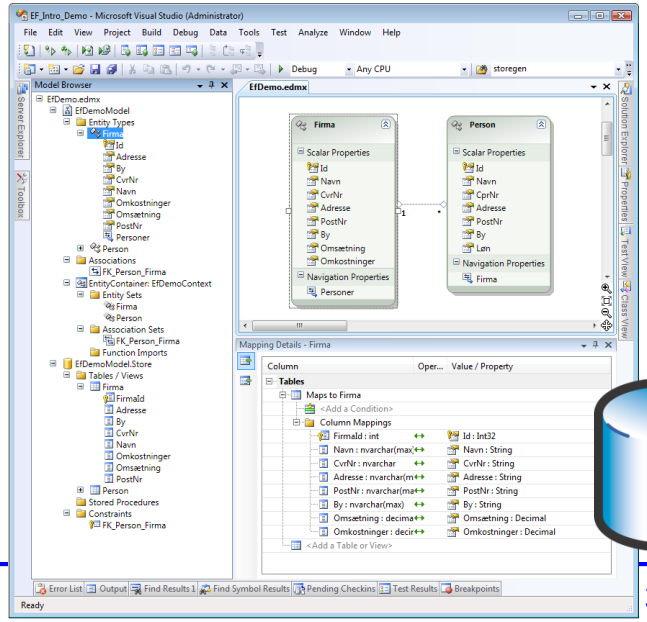
LINQ to SQL

- Supporterer Microsoft SQL Server
- En simpel mappingmekanisme mellem relationel database og objektmodel
- Tager udgangspunkt i en én-til-én mapping mellem SQL Server database og objektmodel
- Mappinger kan foretages via attributter eller XML fil
- Designer til Visual Studio
- Udnytter deferred execution



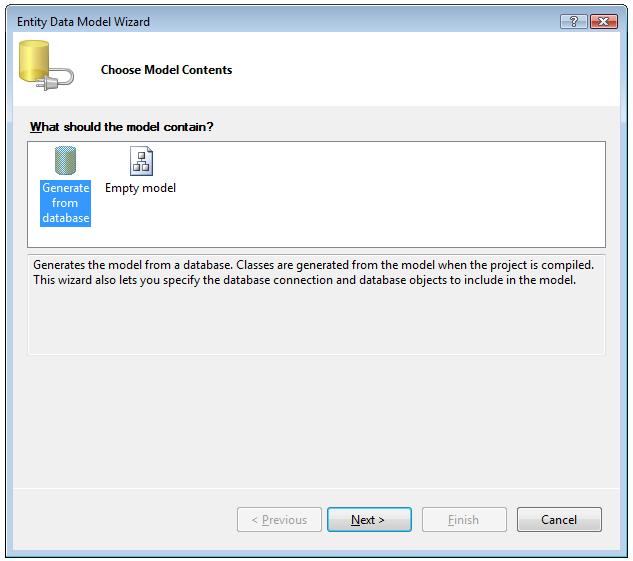
Entity frameworket og LINQ to Entities

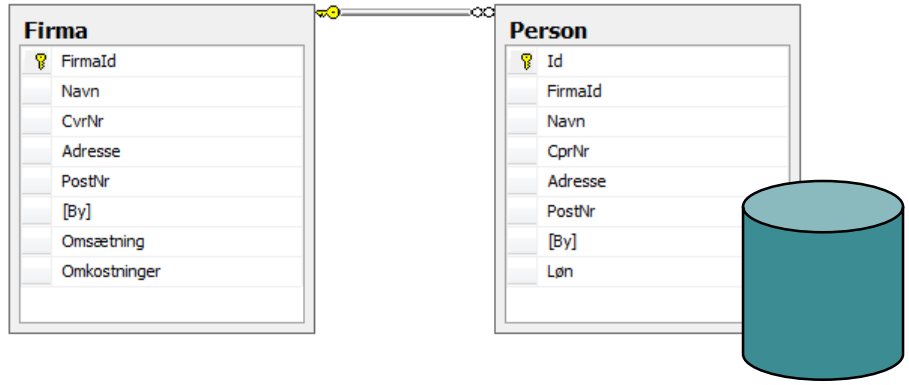
- **Database support via providermodel**
- **Entity Data Model (EDM)**
  - SSDL - Storage Schema Definition Language
    - Beskriver hvordan databasen ser ud
  - CSDL - Conceptual Schema Definition Language)
    - Definerer entiteter der anvendes i applikationen
  - MSL - Mapping Specification Language
    - Mapper SSDL laget til CSDL laget
- **Designer til Visual Studio**
- **Udnytter deferred execution**

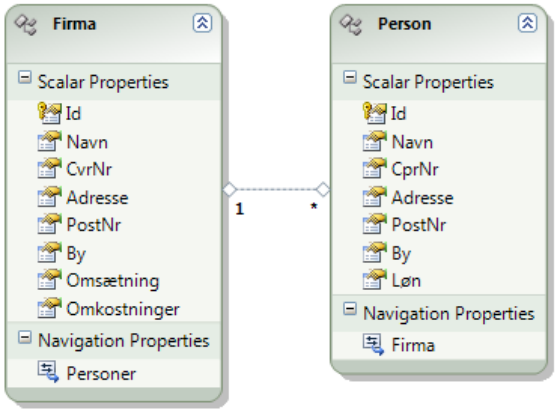


Træk og slip en Entity Framework model

- **Ud fra et database schema genereres en model og en række klasser**








```
var context = new EfDemoContext();

var firmaQuery = from f in context.Firma
                 select f;
Person and =
    firmaQuery.First().Personer.First(i=>i.Navn.StartsWith("And"));

string andensNavn = and.Navn;
```

LINQ to XML

- **System.Linq.Xml namespacet indeholder XML klasser til brug for LINQ to XML**
  - XML typer: XElement, XAttribute, …
  - System.Linq.Xml.Extensions indeholder extension metoder til brug for LINQ queries
- **Opbygning af XML**
- **Forespørgsler på og transformation af XML**
- **Manipulation af XML**

LINQ to XML

Captator

- **Læsning og transformation**

Source xml

```
<CustomerList>
  <Customer>
    <First>Anders</First>
    <Last>And</Last>
  </Customer>
  <Customer>
    <First>Joakim</First>
    <Last>von And</Last>
  </Customer>
</CustomerList>
```

Target xml

```
<Contacts>
  <Contact>
    <FirstName>Anders</FirstName>
    <LastName>And</LastName>
  </Contact>
  <Contact>
    <FirstName>Joakim</FirstName>
    <LastName>von And</LastName>
  </Contact>
</Contacts>
```

```
Dim kundeListe As XElement = XElement.Parse(xml)

Dim kontakter As New XElement("Contacts", _
    From c In kundeListe.Elements("Customer") _
    Select New XElement("Contact", _
        New XElement("FirstName", CStr(c.Element("First"))), _
        New XElement("LastName", CStr(c.Element("Last")))))
```

LINQ to XML

Captator

- **Opbygning af XML vha. funktionel konstruktion**

```
Dim foundDirs As System.IO.DirectoryInfo() = ...

Dim snippet As System.Xml.Linq.XElement = _
    New System.Xml.Linq.XElement("directories", _
        From dir In foundDirs _
        Select New System.Xml.Linq.XElement("directory", _
            New System.Xml.Linq.XAttribute("fileCount", _
                dir.GetDirectories().Count().ToString()), _
            New System.Xml.Linq.XElement("fullName", dir.FullName)))

txtData.Text = snippet.ToString()
```

- **Opbygning af XML vha. VBs integrerede syntaks**

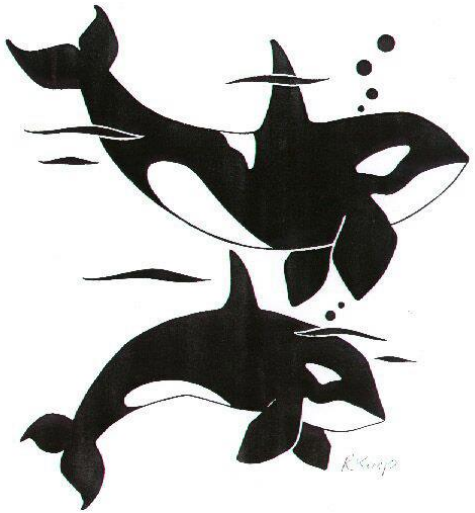
```
Dim foundDirs = ...

Dim snippet As System.Xml.Linq.XElement = _
<directories>
  <%= From dir In foundDirs _
    Select <directory fileCount=<%= dir.GetDirectories().Count().ToString() %>>
             <fullName><%= dir.FullName %></fullName>
           </directory> %>
</directories>

txtData.Text = snippet.ToString()
```

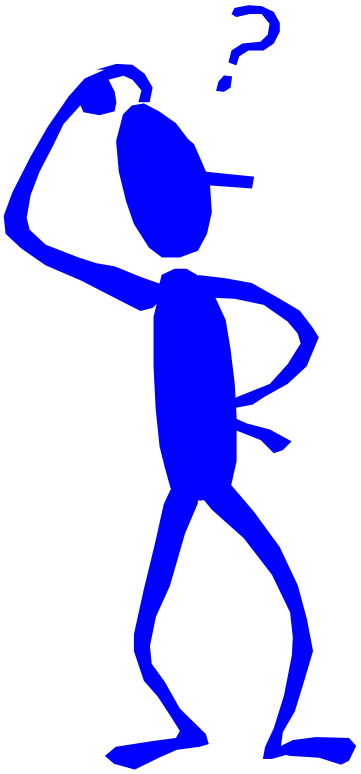
Indsætning af værdier

XML literals og latebinding i VB

```
Dim kundeliste As XElement = <CustomerList>
                               <Customer type="Privat">
                                 <First>Anders</First>
                                 <Last>And</Last>
                               </Customer>
                               <Customer type="Erhverv">
                                 <First>Joakim</First>
                                 <Last>von And</Last>
                               </Customer>
                             </CustomerList>

For Each Dim kunde In kundeliste.Customer
  MessageBox.Show(CStr(kunde.@type))
Next

Dim elementName = "vare"
Dim price = 42
Dim snippet As XElement = _
    <<%= elementName %>><pris><%= price %></pris></>

MessageBox.Show(snippet.ToString())
```

Spørgsmål





www.captator.dk

kurser, rådgivning og softwareudvikling