Praksis eksempler
Lambda expressions
Events
Fil systemet
Collections

Custom collections
LINQ
Web services
Github

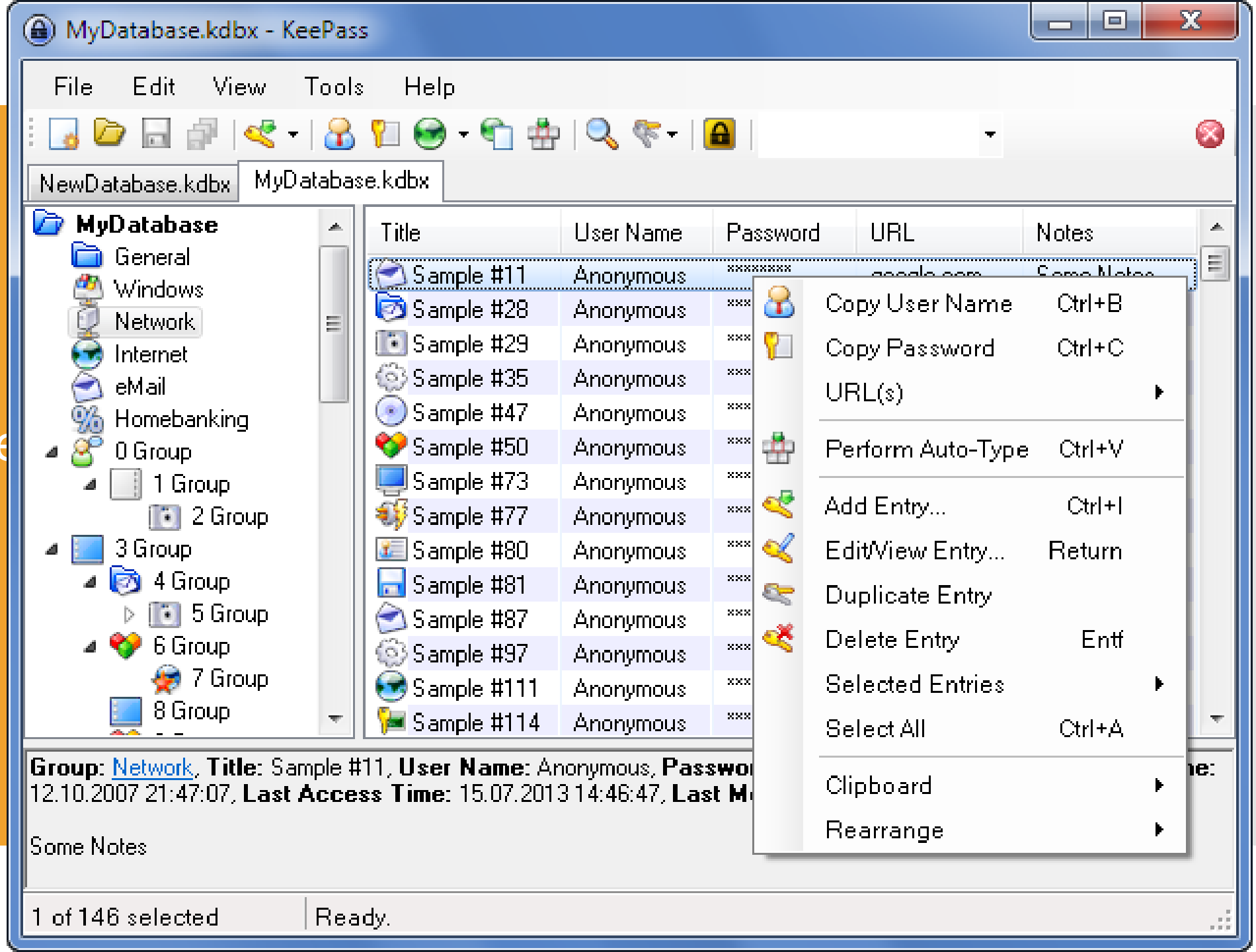
Grundlæggende programmering

Lektion 5

Praksis eksempel

KeePass, et program der husker kodeord

Praksis eksempe
KeePass



Praksis eksempel KeePass

```
87      public enum AppMessage
88      {
89          Null = 0,
90          RestoreWindow = 1,
91          Exit = 2,
92          IpcByFile = 3,
93          AutoType = 4,
94          Lock = 5,
95          Unlock = 6,
96          AutoTypeSelected = 7
97      }
98
99      public static CommandLineArgs CommandLineArgs
100     {
101         get
102         {
103             if(m_cmdLineArgs == null)
104             {
105                 Debug.Assert(false);
106                 m_cmdLineArgs = new CommandLineArgs(null);
107             }
108
109             return m_cmdLineArgs;
110         }
111     }
```

Kun en lille del af den samlede kode

Praksis eksempel Del to

- Hent bogdb.cs filen fra denne lektions mappe på Fronter.
- Dette eksempel benytter de emner vi har gennemgået hidtil til at lave et samlet biblioteks-administrerings program.
- I skal ikke blive forvirret om at det omtaler en database, data gemmes reelt internt i programmet og forsvinder ved lukning, men det vi laver i arrays er en database-lignende struktur og brug, så derfor betegnelsen.

Lambda expressions

En anonym funktion

Lambda expressions

- Et lambda udtryk er en anonym funktion, som kan bruges til at oprette delegates eller udtryk træ typer.
- Ved at bruge lambda udtryk kan man skrive lokale funktioner, der kan overføres som argumenter eller returneres som værdien af funktionskald.
- Lambda udtryk er særligt nyttigt for at skrive LINQ query udtryk.
- For at oprette et lambda udtryk angiver man inputparametre (hvis nogen) på venstre side af lambda operatoren =>, og man sætter udtryk eller udsagn blokken på den anden side
  - Lambda udtrykket `x => x * x` angiver en parameter, x og giver den værdien af x potens. Du kan tildele dette udtryk til en delegeret type.

Lambda expressions

```
1   using System;
2   using System.Collections.Generic;
3
    0 references
4   class Program
5   {
        0 references
6       static void Main()
7       {
8           List<int> elements = new List<int>() { 10, 20, 31, 40 };
9           // ... Find index of first odd element.
10          int oddIndex = elements.FindIndex(x => x % 2 != 0);
11          Console.WriteLine(oddIndex);
12          Console.ReadKey();
13      }
14  }
```

Lambda expressions

Events

Referencer til metoder

Events

- Events er brugerhandlinger såsom tastetryk, klik, musebevægelser, etc., eller nogle hændelser såsom system genererede meddelelser.
- Applikationer er nødt til at reagere på begivenheder, når de opstår. For eksempel interrupts.
- Events anvendes til inter-proces kommunikation.

Events

- Begivenhederne erklæres og udføres i en klasse og forbindes med event handlers der bruger delegates inden for samme klasse eller en anden klasse.
- Klassen, der indeholder eventet, bruges til at udgive eventet.
  - Dette kaldes udgiver (publisher) klassen.
  - Andre klasser, der accepterer eventet, kaldes abonnent (subscriber) klassen.
- Derfor kaldes det at egivenheder bruger en udgiver-abonnent model.

Events

- En publisher er et objekt, som indeholder definitionen af eventet og den delegerede.
  - Event-delegate associationen er også defineret i dette objekt.
  - Et publisher klasse objekt invoker begivenheden og den er anmeldt til andre objekter.
- En subscriber er et objekt der accepterer eventet og leverer en event handler. Delegate i udgiver klassen invoket metoden (event handler) i abonnent klassen.

Events

- For at declar en event I en klasse skal der først laves en delegate type for eventet.

```
public delegate string MyDel(string str);
```

- Derefter declares eventet selv med event keyword.

```
event MyDel MyEvent;
```

Events

```
1   using System;
2   namespace SampleApp
3   {
4       public delegate string MyDel(string str);
5
        3 references
6       class EventProgram
7       {
8           event MyDel MyEvent;
9
            1 reference
10          public EventProgram()
11          {
12              this.MyEvent += new MyDel(this.WelcomeUser);
13          }
14
            1 reference
15          public string WelcomeUser(string username)
16          {
17              return "Welcome " + username;
18          }
19
            0 references
20          static void Main(string[] args)
21          {
22              EventProgram obj1 = new EventProgram();
23              string result = obj1.MyEvent("Student");
24              Console.WriteLine(result);
25              Console.ReadKey();
26          }
27
28      }
29  }
```

Events

Events & Delegates

Fil systemet

At gemme filer på disken

Fil systemet

- En fil er en samling gemt på en disk med et specifikt navn og mappe sti. Når filen åbnes til læsning eller skrivning bliver den en stream.
- En stream er grundlæggende skevensen af bytes der går gennem kommunikations stien. Der er to hoved streams: **input stream** og the output stream.
  - Input stream bruges til at læse data fra filen (read operation)
  - og output stream bruges til at skrive til filen (write operation).

Fil systemet I/O klasser

- System.IO namespace har en række klasser der bruges til at udføre en vifte af operationer med filer
  - Lave filer
  - Slette filer
  - Læse fra eller skrive til en fil
  - Lukke en fil osv.

I/O Class	Description
BinaryReader	Reads primitive data from a binary stream.
BinaryWriter	Writes primitive data in binary format.
BufferedStream	A temporary storage for a stream of bytes.
Directory	Helps in manipulating a directory structure.
DirectoryInfo	Used for performing operations on directories.
DriveInfo	Provides information for the drives.
File	Helps in manipulating files.
FileInfo	Used for performing operations on files.
FileStream	Used to read from and write to any location in a file.
MemoryStream	Used for random access to streamed data stored in memory.
Path	Performs operations on path information.
StreamReader	Used for reading characters from a byte stream.
StreamWriter	Is used for writing characters to a stream.
StringReader	Is used for reading from a string buffer.
StringWriter	Is used for writing into a string buffer.

Fil systemet
FileStream klassen

- FileStream klassen i System.IO namespace hjælper med at skrive fra, skrive til og lukke filer. Klassen er afledt af den abstrakte klasse Stream.
- For at lave en ny fil eller åbne en eksisterende fil skal man have et **FileStream** objekt.

```
FileStream <object_name> = new FileStream( <file_name>,
<FileMode Enumerator>, <FileAccess Enumerator>,
<FileShare Enumerator>);
```

- Her laves et FileStream objekt F til at læse filen sample.txt

```
FileStream F = new FileStream("sample.txt", FileMode.Open,
FileAccess.Read, FileShare.Read);
```

Parameter	Description
FileMode	The **FileMode** enumerator defines various methods for opening files. The members of the FileMode enumerator are: ▫ **Append**: It opens an existing file and puts cursor at the end of file, or creates the file, if the file does not exist. ▫ **Create**: It creates a new file. ▫ **CreateNew**: It specifies to the operating system, that it should create a new file. ▫ **Open**: It opens an existing file. ▫ **OpenOrCreate**: It specifies to the operating system that it should open a file if it exists, otherwise it should create a new file. ▫ **Truncate**: It opens an existing file and truncates its size to zero bytes.
FileAccess	**FileAccess** enumerators have members: **Read**, **ReadWrite** and **Write**.
FileShare	**FileShare** enumerators have the following members: ▫ **Inheritable**: It allows a file handle to pass inheritance to the child processes ▫ **None**: It declines sharing of the current file ▫ **Read**: It allows opening the file for reading ▫ **ReadWrite**: It allows opening the file for reading and writing ▫ **Write**: It allows opening the file for writing

Fil systemet
FileStream klassen

```
1   using System;
2   using System.IO;
3
4   namespace FileIOApplication
5   {
        0 references
6       class Program
7       {
            0 references
8           static void Main(string[] args)
9           {
10              FileStream F = new FileStream("test.dat", FileMode.OpenOrCreate, FileAccess.ReadWrite);
11              for (int i = 1; i <= 20; i++)
12              {
13                  F.WriteByte((byte)i);
14              }
15
16              F.Position = 0;
17              for (int i = 0; i <= 20; i++)
18              {
19                  Console.Write(F.ReadByte() + " ");
20              }
21              F.Close();
22              Console.ReadKey();
23          }
24      }
25  }
```

Collections

Klasser til at gemme og hente data

Collections

- Collection klasser er specialiseret i data opbevaring og hentning.
- Disse klasser giver support for stacks, queues, lists og hash tables.
- De fleste collection klasser implementerer de samme interfaces.
- Collection klasser tjener flere formal:
  - At allokere hukommelse dynamisk til elementer
  - At tilgå en liste af items på basis af et indeks osv.
- Disse klasse laver collections af objekter af Object klassen, der er basis klassen for alle data typer i C#.

Her en tabel over de mest brugte klasser i System.Collection namespace.

Class	Description and Useage
ArrayList	It represents ordered collection of an object that can be **indexed** individually. It is basically an alternative to an array. However, unlike array you can add and remove items from a list at a specified position using an **index** and the array resizes itself automatically. It also allows dynamic memory allocation, adding, searching and sorting items in the list.
Hashtable	It uses a **key** to access the elements in the collection. A hash table is used when you need to access elements by using key, and you can identify a useful key value. Each item in the hash table has a **key/value** pair. The key is used to access the items in the collection.
SortedList	It uses a **key** as well as an **index** to access the items in a list. A sorted list is a combination of an array and a hash table. It contains a list of items that can be accessed using a key or an index. If you access items using an index, it is an ArrayList, and if you access items using a key , it is a Hashtable. The collection of items is always sorted by the key value.
Stack	It represents a **last-in, first out** collection of object. It is used when you need a last-in, first-out access of items. When you add an item in the list, it is called **pushing** the item and when you remove it, it is called **popping** the item.
Queue	It represents a **first-in, first out** collection of object. It is used when you need a first-in, first-out access of items. When you add an item in the list, it is called **enqueue** and when you remove an item, it is called **deque**.
BitArray	It represents an array of the **binary representation** using the values 1 and 0. It is used when you need to store the bits but do not know the number of bits in advance. You can access items from the BitArray collection by using an **integer index**, which starts from zero.

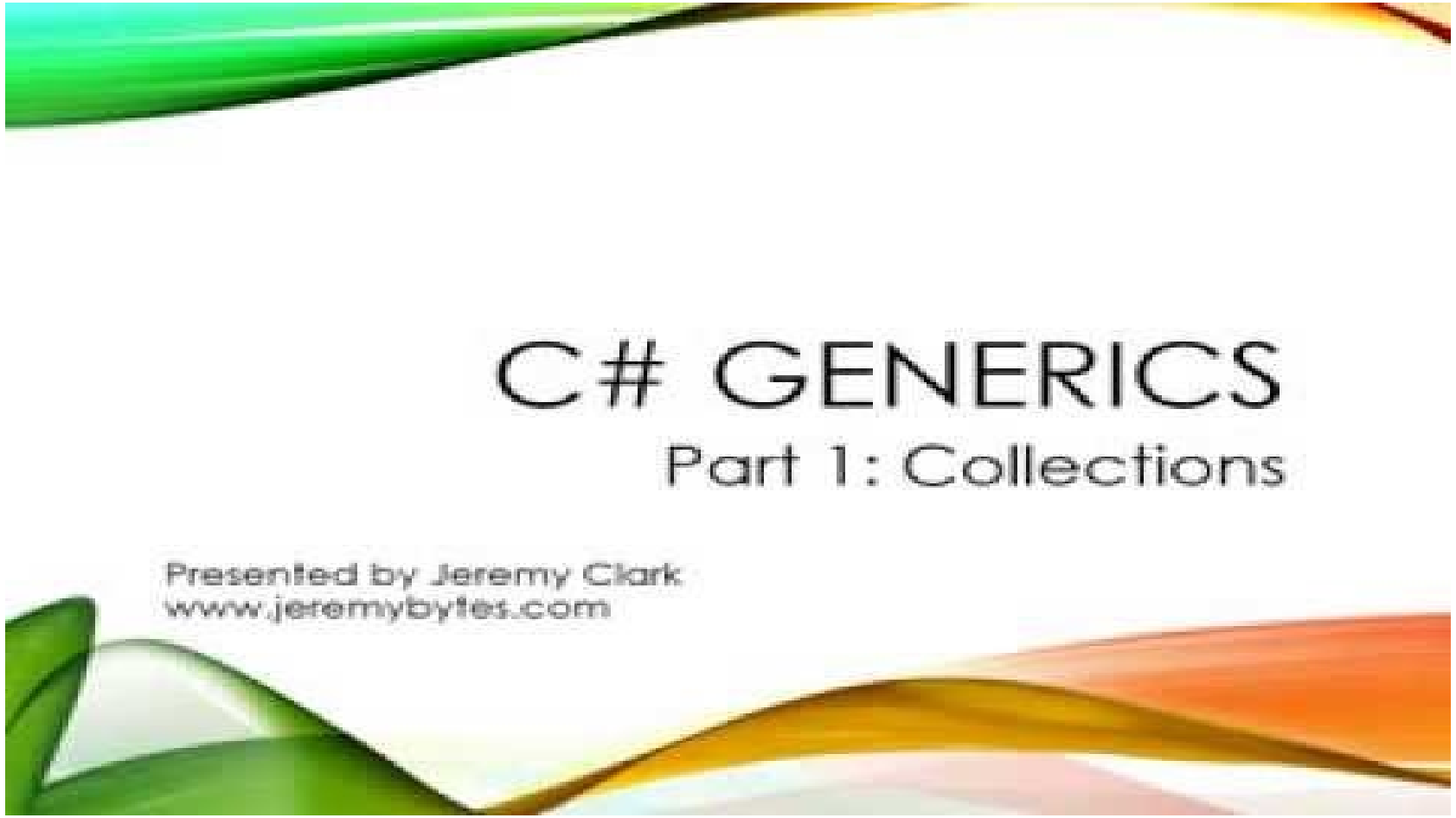
Custom Collections

Ens egne collections

Custom Collections

- Man kan lave ens egen collection klasse ved at arve fra en af de mange.NET Framework collection klasser og føje kode til for at implementere ens egen burger-definerede funktionalitet.
- Ligegyldigt hvor begrænsede funktioner man vil give ens collection er der nogle funktioner man skal give den.
  - En collection er nødt til at støtte behandlingen alle sine poster med et For … While loop.
  - Det er meget usædvanligt at en collection ikke understøtter hentning af enkelte poster i collectionen ved en position (en indeksering).
- Nogle funktioner vil dog føre til andre.
  - Hvis man f.eks. vil understøtte at tilføje elementer til ens collection vil man også ønske at understøtte initializers (at tilføjer flere elementer til samlingen ved hjælp af tuborg-klammer ({…}).

Collections



LINQ

Adgang til databaser, XML og andre data kilder

LINQ

- Akronymet LINQ star for Language INtegrated Query.
- Gennem LINQ queries får man let data adgang til in-memory objekter, databaser, XML documenter og mere til.
- LINQ blev indført i Visual Studio 2008 og er designet af Anders Hejlsberg.
- LINQ tillader en at skrive queries selv uden kendskab til query sprog som SQL, XML osv.
- LINQ queries kan skrives til forskellige datatyper.

LINQ

```
1   using System;
2   using System.Linq;
3
    0 references
4   class Program
5   {
        0 references
6       static void Main()
7       {
8           string[] words = { "hello", "wonderful", "LINQ", "beautiful", "world" };
9           //Get only short words
10          var shortWords = from word in words
11                           where word.Length <= 5
12                           select word;
13
14          //Print each word out
15          foreach (var word in shortWords)
16          {
17              Console.WriteLine(word);
18          }
19          Console.ReadLine();
20      }
21  }
```

LINQ Syntaks

Der er to syntakser I LINQ.

- Lamda (Method) Syntax

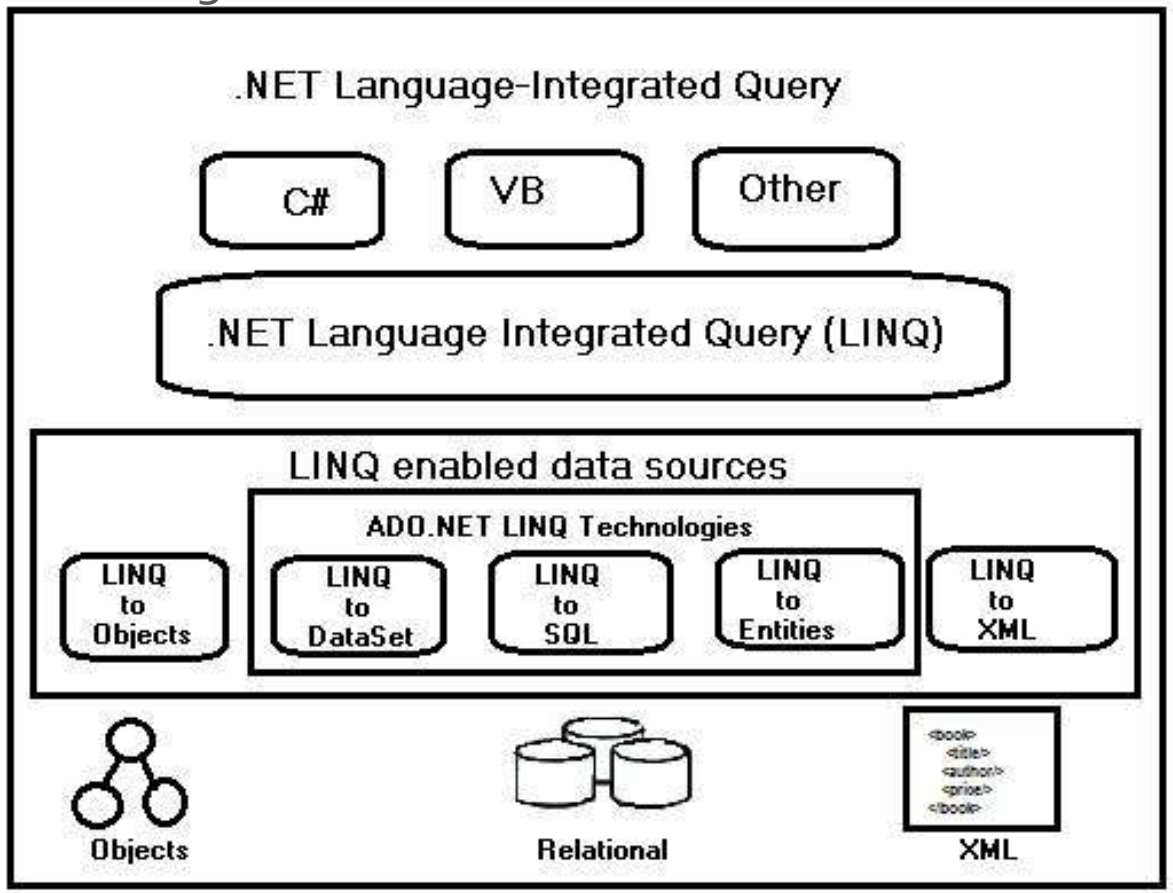
```
var longWords = words.Where( w => w.length > 10);
Dim longWords = words.Where(Function(w) w.length > 10)
```

- Query (Comprehension) Syntax

```
var longwords = from w in words where w.length > 10;
Dim longwords = from w in words where w.length > 10
```

LINQ
Arkitektur i .NET

- LINQ har en 3-laget arkitektur i hvilken det øverste lag består af sprog (language) extensions og det nederste lag består af data kilder typisk er objekter der implementerer IEnumerable<T> eller IQueryable<T> generiske interfaces.



LINQ
Query udtryk

- Query udtryk er ikke andet end en LINQ forespørgsel, udtrykt i en form, der svarer til SQL med query operatører som Select, Where og OrderBy.
- Query udtryk starter som regel med nøgleordet ”From".

LINQ
Query udtryk

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5
6   namespace Operators
7   {
        0 references
8       class LINQQueryExpressions
9       {
            0 references
10          static void Main()
11          {
12              // Specify the data source.
13              int[] scores = new int[] { 97, 92, 81, 60 };
14
15              // Define the query expression.
16              IEnumerable<int> scoreQuery = from score in scores
17                                            where score > 80
18                                            select score;
19
20              // Execute the query.
21              foreach (int i in scoreQuery)
22              {
23                  Console.Write(i + " ");
24              }
25              Console.ReadKey();
26          }
27      }
28  }
```

LINQ
vs Stored Procedures

Der er en række forskelle på LINQ og Stored procedures.

- Stored procedures er meget hurtigere end en LINQ forespørgsel da de følger en forventet udførelses plan.
- Det er nemmere at undgå run-time fejl under udførelse af en LINQ query i forhold til en stored procedure.
  - LINQ har Visual Studio Intellisense support samt full-type checking ved kompilerings-tid.
- LINQ tillader debugging igennem .NET debugger hvilket ikke er tilfældet for stored procedures.
- LINQ tilbyder understøttelse af flere databaser i modsætning til stored procedures, hvor det er vigtigt at omskrive koden til forskellige typer af databaser.
- Implementering af LINQ baserede løsninger er nemt og enkelt i forhold til indsættelsen af en række af stored procedures.

LINQ
Hvorfor – før og nu

Før LINQ

```
SqlConnection sqlConnection = new SqlConnection(connectString);
SqlConnection.Open();
System.Data.SqlClient.SqlCommand sqlCommand = new SqlCommand();
sqlCommand.Connection = sqlConnection;
sqlCommand.CommandText = "Select * from Customer";
return sqlCommand.ExecuteReader (CommandBehavior.CloseConnection)
```

Med LINQ

```
Northwind db = new Northwind(@"C:\Data\Northwnd.mdf");
var query = from c in db.Customers
            select c;
```

LINQ

Web services

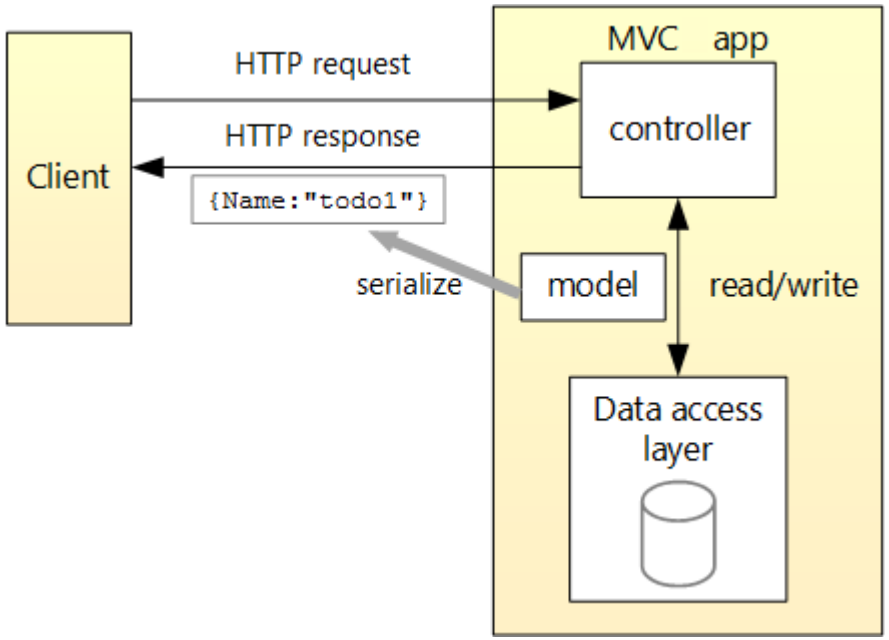
Funktioner over nettet

Web services

- En web service er en web-baseret funktionalitet der tilgås ved hjælp af internet protokoller for at blive anvendt af web-applikationer.
  - En web service er en web-applikation, som dybest set er en klasse, der består af metoder, der kan anvendes af andre programmer.
  - Den følger også en code-behindarkitektur såsom ASP.NET websider, selv om den ikke har en brugergrænseflade.
- Der er tre dele af webservice udvikling:
  - Front-end udseende (view) i HTML og CSS
  - Behandling af dataen i back end (model)
  - Afsendelse af denne data til front-end (controller)

Web services

ASP.NET Core har indbygget support for MVC bygning af Web APIs.



- Klienten er hvad der benytter web APIen (browser, mobil app osv.)
- *Model* er et objekt, der repræsenterer dataene i din ansøgning.
- *Controller* er et objekt, der håndterer HTTP-anmodninger og skaber HTTP-svaret.

Følgende tutorial anbefales hvis man vil snuse nærmere til det at lave web-services

- https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api

Github

Versionsstyring og deling af kode

Github

Opgave

Jeg står til rådighed til svar på spørgsmål og forklaring af fejl

Opgave

OPGAVE

- Begynd at tage prøverne inde på https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169 fra en ende af.
- I behøves ikke se videoerne, men brug dem endelig som reference / genopfriskning.

Kilder

Materiale benyttet i denne lektion
Noget af det er udover pensum-listen!

Kilder

Praksis eksempel

- http://keepass.info/
- https://sourceforge.net/projects/keepass/files/KeePass%202.x/

Lambda expressions

- https://msdn.microsoft.com/en-us/library/bb397687.aspx
- https://msdn.microsoft.com/en-us/library/bb397675.aspx
- https://www.dotnetperls.com/lambda
- https://youtu.be/xev-kNmz_ao

Events

- https://www.tutorialspoint.com/csharp/csharp_events.htm
- https://youtu.be/jQgwEsJISyo

Kilder

Fil system

- https://www.tutorialspoint.com/csharp/csharp_file_io.htm

Collections

- https://www.tutorialspoint.com/csharp/csharp_collections.htm
- https://youtu.be/J9Cwi45UtZU

Custom collections

- https://support.microsoft.com/da-dk/kb/307484
- https://msdn.microsoft.com/en-us/library/xth2y6ft(v=vs.71).aspx
- http://www.c-sharpcorner.com/uploadfile/skumaar_mca/custom-collection-class-in-c-sharp/

Kilder

Custom collections

- https://www.codeproject.com/articles/265692/having-fun-with-custom-collections
- https://visualstudiomagazine.com/articles/2015/02/01/creating-a-simple-collection-class.aspx

LINQ

- http://www.tutorialspoint.com/linq/
- https://kevinlawry.wordpress.com/2012/08/07/why-i-avoid-stored-procedures-and-you-should-too/
- https://youtu.be/1UqjUgcdq6g

Kilder

Web services

- https://www.tutorialspoint.com/asp.net/asp.net_web_services.htm
- https://blog.tonysneed.com/2016/01/06/wcf-is-dead-long-live-mvc-6/
- https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api

Github

- https://www.pluralsight.com/blog/software-development/github-tutorial
- https://youtu.be/ofKg7e37bQE