Grundlagen in C# und .net

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

- TabbedPane (Register)
- ListView
- Tree
- Tabelle
- MDI-Programme
- Erweiterte Grafik
- Threads und Semaphore
- Datenbanken
- DLL
- **LINQ**

Language Integrated Query

- Sprachergänzung ab 2008, C# Version 3.0
- Abstraktionsebene für Abfragen
- Typen
  - LINQ to Objects
  - LINQ to XML
  - LINQ to SQL
  - LINQ to ADO.NET
- Vorteil
  - Eine Syntax für beliebige Datenstrukturen
- Nachteile
  - Langsam

Grundlagen für LINQ

- Neuer „Datentyp“
  - var x = 5;
  - var a = 5.1;
  - var s = "C# ist toll";
  - var feld = new [] { 0,1,2}; // Typ: int Array
  - var z = new { Name="Andrea" , Matrnr=12345; }; // Typ ?
  - var list = new List<int>(); // Typ: List<int>
- Der Compiler sucht den am besten passenden Typ aus
- Wichtige Einschränkungen:
  - Die Variable muss lokal definiert werden
  - Initialisierung bei der Deklaration
  - Der Wert darf nicht null sein

Lambda-Ausdrücke

- Lambda-Ausdrücke sind anonyme Methoden, die Ausdrücke und Anweisungen enthalten können
- Sie können für die Erstellung von Delegates verwendet werden

```
if ( i== 1) {
   process = (double x, double y) => { return x+y};
}
else {
   process = (double x, double y) => { return x-y};
}
```

- Lambda-Ausdrücke verwenden den Operator =>
- Links sind die Eingangsparameter
- Rechts ist der Anweisungsblock

1. Beispiel mit LINQ

```
// 1. Deklaration
String[] studenten = { "Müller", "Schmidt", "Meier",  "Schulze", "Abba" };

// 2. Query Abfrage erstellen
// studies is an IEnumerable<String>
var studies =
   from std in studenten
   select std;

// 3. Query ausführen, verzögert, implizit
textBox1.Text = "";
foreach (String std in studies)
{
      textBox1.AppendText("Student: "+std+"\r\n");
}
}
```

2. Beispiel mit LINQ

```
 //  1. Deklaration
String[] studenten = { "Müller", "Schmidt", "Meier",  "Schulze", "Abba" };

// 2. Query Abfrage erstellen
// studies is an IEnumerable<String>
 var studies =
   from std in studenten
   where std.EndsWith("er")
   select std;

// 3. Query ausführen, verzögert, implizit
 textBox1.Text = "";
 foreach (String std in studies)
 {
      textBox1.AppendText("Student: "+std+"\r\n");
 }
```


3. Beispiel mit LINQ

```
 //  1. Deklaration
int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

// 2. Query Abfrage erstellen
// numQuery is an IEnumerable<int>
var numQuery =
   from num in numbers
   where (num % 2) == 0
   select num;

// 3. Query ausführen, verzögert
foreach (int num in numQuery)
{
   Console.Write("{0,1} ", num);
}
}
```

Abfrage-Operatoren

- Aggregatoperationen

  Aggregate, Average, Count, LongCount, Min, Max, Sum
- Castingoperationen

  Cast, OfType, ToArray, ToDictionary, ToList, ToLookup
- Elementoperationen

  DefaultIfEmpty, ElementAt, ElementAtOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
- Gleichheitsoperationen

  EqualAll
- Sequenzoperationen

  Empty, Range, Repeat

Abfrage-Operatoren

- Gruppierungsoperationen

  GroupBy
- Joinoperationen

  Join, GroupJoin
- Sortieroperationen

  OrderBy, ThenBy, OrderByDescending, ThenByDescending, Reverse
- Aufteilungsoperationen

  Skip, SkipWhile, Take, TakeWhile
- Quantifizierungsoperationen

  All, Any, Contains

Abfrage-Operatoren

- Restriktionsoperationen

  Where
- Projektionsoperationen

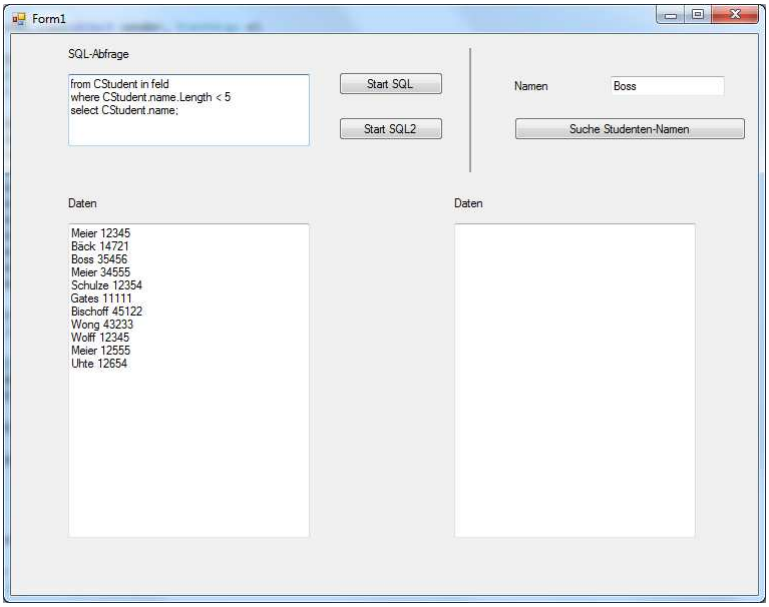
  Select, SelectMany
- Set-Operationen

  Concat, Distinct, Except, Intersect, Union

4. Beispiel mit LINQ

```
String[] studenten = { "Müller", "Schmidt", "Meier", "Schulze", "Abba" };
  var studies =
  from std in studenten
  where std.EndsWith("er")
  select std;

  int numCount1 = studies.Count();

  // 3. Query execution.
  textBox1.Text = "";
  textBox1.AppendText("Anzahl der Studenten: " + numCount1 + "\r\n");

  numCount1 = (from std in studenten
          where std.EndsWith("er")
          select std
          ).Count();

  // 3. Query execution.
  textBox1.AppendText("Anzahl der Studenten: " + numCount1 + "\r\n");
```

2. Dialog-Beispiel mit LINQ



Form1_load

```
Student std;
studenten = new List<Student>();

    studenten.Add(new Student("Meier",   12345));
    studenten.Add(new Student("Bäck", 14721));
    studenten.Add(new Student("Boss", 35456));
    studenten.Add(new Student("Meier", 34555));
    studenten.Add(new Student("Schulze", 12354));
    studenten.Add(new Student("Gates", 11111));
    studenten.Add(new Student("Bischoff", 45122));
    studenten.Add(new Student("Wong",     43233));
    studenten.Add(new Student("Wolff", 12345));
    studenten.Add(new Student("Meier", 12555));
    studenten.Add(new Student("Uhte", 12654));
;
```

Form1_load

```
string sStr="";
for (int i = 0; i < studenten.Count; i++)
{
   std=(Student)studenten[i];
   if (i==0)
      sStr=std.ToString();
   else
      sStr=sStr+"\r\n"+std.ToString();
}
EQuelle.Text=sStr;
```


Bn1_Click

```
string sStr = Esql.Text.Trim();
//if (sStr.Equals("")) return;
int n = 5;

var studies = from Student in studenten
              where Student.name.Length > n
              select Student.name;
sStr="";
foreach (var std in studies)
{
   sStr = sStr + std.ToString() + "\r\n";
}
EZiel.Text = sStr;
```

Bn2_Click

```
 string sStr = Esql.Text.Trim();
//if (sStr.Equals("")) return;
int max = 30000;  // Geht auch mit JSpinner !

var studies =   studenten
            .Where  (Student => Student.matrnr>max )
            .Select (c => c.name);

sStr = "";
foreach (var std in studies)
{
   sStr = sStr + std.ToString() + "\r\n";
}
EZiel.Text = sStr;
```


Bn3_Click

```
string sStr = EName.Text.Trim();
if (sStr.Equals("")) return;
Student std = studenten.Find(c => c.name.Equals( sStr ));

EZiel.Text = std.ToString();
```