



**complement**

getting ideas done

# Microsoft LINQ – .NET meets data

ASQF – Fachgruppe Java / XML

29.11.2007



complement  
getting ideas done

**complement AG**

Thomas Hemmer  
CTO

thomas.hemmer@complement.de

# Microsoft LINQ – .NET meets data

## Agenda

C# 3.0 Features für LINQ

LINQ – Query Basics

LINQ for SQL, XML, ...



complement  
getting ideas done

# Microsoft LINQ – .NET meets data

## Agenda

C# 3.0 Features für LINQ

LINQ – Query Basics

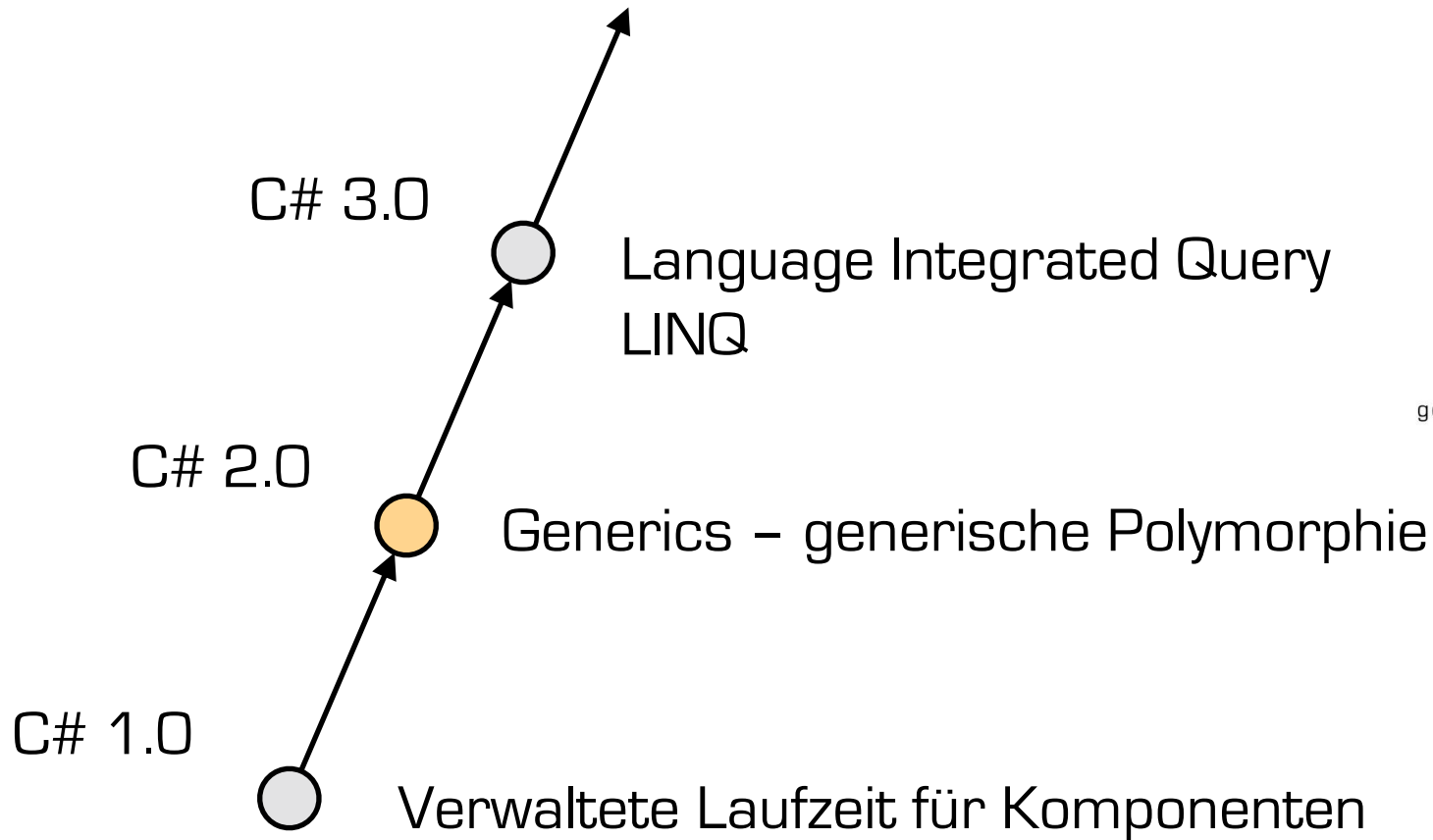
LINQ for SQL, XML, ...



complement  
getting ideas done

# C# - schnelle Evolution

## ■ Schwerpunkte



complement  
getting ideas done

# C# 3.0 - Entwurfsziele

- Integration von Objekten, relationalen Daten und XML (LINQ)
- Sowie
  - Verbesserung der Prägnanz der Sprache
  - Hinzufügen funktionaler Konstrukte
  - Vermeidung von enger Kopplung an bestimmte APIs
  - 100 % abwärtskompatibel



complement  
getting ideas done

# C# 3.0 – Erweiterungen für LINQ

- Spracherweiterungen
  - Local Variable Type Inference
  - Object Initialisers
  - Anonymous Types
  - Lambda Expressions
  - Extension Methods



complement  
getting ideas done

# Local Variable Type Inference

```
int i = 42;  
string s = "ASQF";  
double d = 3.14;  
int[] numbers = new int[] { 1, 2, 3};  
Dictionary<int,Order> orders = new Dictionary<int,Order>{};
```

```
var i = 42;  
var s = "ASQF";  
var d = 3.14;  
var numbers = new int[] { 1, 2, 3};  
var orders = new Dictionary<int,Order>{};
```

“Der Datentyp auf der  
rechten Seite der  
Zuweisung”



complement  
getting ideas done

# Object Initialisers / Collection Initialisers

```
public class Point
{
    private int x, y;

    public int X { get { return x; } set { x = value; } }
    public int Y { get { return y; } set { y = value; } }
}
```

```
Point a = new Point { X = 0, Y = 1 };
```

```
Point a = new Point();
a.X = 0;
a.Y = 1;
```



Property Zuweisung



complement  
getting ideas done

# Object Initialisers / Collection Initialisers

## Einschub: Auto-Implemented Properties

```
public class Point  
{  
    public int X { get; set; }  
    public int Y { get; set; }  
}
```

Auto implemented  
Properties



complement  
getting ideas done

```
Point a = new Point { X = 0, Y = 1 };
```

Property Zuweisung

```
Point a = new Point();  
a.X = 0;  
a.Y = 1;
```

# Anonymous Types

XYZ

```
class XYZ
{
    public string Name;
    public int Age;
}
```

```
var o = new { Name = "Jenny", Age = 31 };
```

```
var results = new[]
{
    new { Name = "P1", Age = 5},
    new { Name = "P2", Age = 6}
};
```



complement  
getting ideas done

# DEMO



complement  
getting ideas done

# Lambda Expressions

```
public delegate bool Predicate<T>(T obj);

public class List<T>
{
    public List<T> FindAll(Predicate<T> test)
    {
        List<T> result = new List<T>();
        foreach (T item in this)
            if (test(item)) result.Add(item);
        return result;
    }
    ...
}
```



complement  
getting ideas done

# Lambda Expressions

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers =
            GetCustomerList();
        List<Customer> locals =
            customers.FindAll(new
                Predicate<Customer>(CityEqualsHove)
            );
    }

    static bool CityEqualsHove(Customer c) {
        return c.City == "Hove";
    }
}
```



complement  
getting ideas done

# Lambda Expressions

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers =
            GetCustomerList();
        List<Customer> locals =
            customers.FindAll(
                delegate(Customer c)
                    { return c.City == "Hove"; }
            );
    }
}
```



complement  
getting ideas done

# Lambda Expressions

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers =
            GetCustomerList();
        List<Customer> locals =
            customers.FindAll(
                (Customer c) => {return c.City ==
                                "Hove";}
            );
    }
}
```

Lambda  
expression



complement  
getting ideas done

# Lambda Expressions

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers =
            GetCustomerList();
        List<Customer> locals =
            customers.FindAll(
                c => c.City == "Hove"
            );
    }
}
```

Lambda  
expression



complement  
getting ideas done

# Lambda Expressions

```
public class MyClass
{
    public static void Main() {
        List<Customer> customers =
            GetCustomerList();
        List<Customer> locals =
            customers.FindAll(c => c.City == "Hove");
    }
}
```



Lambda  
expression



complement  
ideas done

# Extension Methods

```
static bool MyMethod(Person p)
{
    //return MyExtensions.IsOver15(p);
    return p.IsOver15();
}

public static class MyExtensions
{
    //[System.Runtime.CompilerServices.Extension()]
    public static bool IsOver15(this Person p)
    {
        return p.Age > 15;
    }
}
```



**complement**  
getting ideas done

# Extension Methods

Extension Method

```
namespace MyExt
{
    public static class Extensions
    {
        public static string Concatenate(this IEnumerable<string> strings,
            string separator) {...}
    }
}
```

```
using MyExt;
```

Extensions im aktuellen Scope  
bekanntmachen

```
string[] names = new string[] { "Jenny", "Daniel", "Rita" };
string s = names.Concatenate(", ");
```

IntelliSense!

obj.Foo(x, y)  
↓  
XXX.Foo(obj, x, y)



complement  
getting ideas done

# Microsoft LINQ – .NET meets data

## Agenda

C# 3.0 Features für LINQ

LINQ – Query Basics

LINQ for SQL, XML, ...



complement  
getting ideas done

# Query Expressions

- Queries bestehen aus Methodenaufrufen



```
from c in customers  
where c.City == "Hove"  
select new { c.Name, c.Phone };
```

```
customers  
.Where(c => c.City == "Hove")  
.Select(c => new { c.Name, c.Phone });
```



complement  
getting ideas done

# Query Expressions

```
var contacts =  
  from c in customers  
  where c.City == "Hove"  
  select new { c.Name, c.Phone };
```

Query expressions

Local variable type inference

Lambda expressions

```
var contacts =  
  customers  
  .Where(c => c.City == "Hove")  
  .Select(c => new { c.Name, c.Phone });
```

Extension methods

Anonymous types

Object initializers



complement  
getting ideas done

```

static void Main()
{
    int[] numbers = { 5, 10, 8, 3, 6, 12};

    //Query syntax:
    IEnumerable<int> numQuery1 =
        from num in numbers
        where num % 2 == 0
        orderby num
        select num;

    //Method syntax:
    IEnumerable<int> numQuery2 = numbers.Where(num => num % 2 == 0).OrderBy(n => n);

    foreach (int i in numQuery1)
    {
        Console.Write(i + " ");
    }
    Console.WriteLine(System.Environment.NewLine);
    foreach (int i in numQuery2)
    {
        Console.Write(i + " ");
    }

    Console.ReadKey();
}

```



complement  
getting ideas done

# DEMO



complement  
getting ideas done

# Query Expressions

```
from c in Customers
where c.City == "Erlangen"
select new { c.Name, c.Address };
```



complement  
getting ideas done

# Query Expressions

Beginn mit *from*

Null oder mehrere  
*from, join, let, where, or*  
*orderby*

```
from id in source  
{ from id in source |  
join id in source on expr equals expr [ into id ] |  
let id = expr |  
where condition |  
orderby ordering, ordering, ... }  
select expr | group expr by key  
[ into id query ]
```

Endet mit  
*select* or *group by*

Optionale *into*  
Weiteführung  
(nesting)



complement  
getting ideas done

# Query Expressions

Project	<b>Select</b> <i>&lt;expr&gt;</i>
Filter	<b>Where</b> <i>&lt;expr&gt;</i> , <b>Distinct</b>
Test	<b>Any</b> ( <i>&lt;expr&gt;</i> ), <b>All</b> ( <i>&lt;expr&gt;</i> )
Join	<i>&lt;expr&gt;</i> <b>Join</b> <i>&lt;expr&gt;</i> <b>On</b> <i>&lt;expr&gt;</i> <b>Equals</b> <i>&lt;expr&gt;</i>
Group	<b>Group By</b> <i>&lt;expr&gt;</i> , <i>&lt;expr&gt;</i> <b>Into</b> <i>&lt;expr&gt;</i> , <i>&lt;expr&gt;</i> <b>Group Join</b> <i>&lt;decl&gt;</i> <b>On</b> <i>&lt;expr&gt;</i> <b>Equals</b> <i>&lt;expr&gt;</i> <b>Into</b> <i>&lt;expr&gt;</i>
Aggregate	<b>Count</b> ( <i>&lt;expr&gt;</i> ), <b>Sum</b> ( <i>&lt;expr&gt;</i> ), <b>Min</b> ( <i>&lt;expr&gt;</i> ), <b>Max</b> ( <i>&lt;expr&gt;</i> ), <b>Avg</b> ( <i>&lt;expr&gt;</i> )
Partition	<b>Skip</b> [ <b>While</b> ] <i>&lt;expr&gt;</i> , <b>Take</b> [ <b>While</b> ] <i>&lt;expr&gt;</i>
Set	<b>Union</b> , <b>Intersect</b> , <b>Except</b>
Order	<b>Order By</b> <i>&lt;expr&gt;</i> , <i>&lt;expr&gt;</i> [ <b>Ascending</b>   <b>Descending</b> ]



complement  
getting ideas done

# Expression Trees

```
public delegate bool Predicate<T>(T item);
```

```
Predicate<Customer> test = c => c.City == "Hove";
```



```
Predicate<Customer> test = new Predicate<Customer>(XXX);
```

```
private static bool XXX(Customer c) {  
    return c.City == "Hove";  
}
```



complement  
getting ideas done

# Expression Trees

```
public delegate bool Predicate<T>(T item);
```

```
Expression<Predicate<Customer>> test = c => c.City == "Hove";
```



```
ParameterExpression c =  
    Expression.Parameter(typeof(Customer), "c");  
Expression expr =  
    Expression.Equal(  
        Expression.Property(c, typeof(Customer).GetProperty("City")),  
        Expression.Constant("Hove")  
    );  
Expression<Predicate<Customer>> test =  
    Expression.Lambda<Predicate<Customer>>(expr, c);
```



complement  
making ideas done

# LINQ Architektur

```
var query = from c in customers where c.City == "Hove" select c.Name;
```

```
var query = customers.Where(c => c.City == "Hove").Select(c => c.Name);
```

Source implements  
IEnumerable<T>

Source implements  
IQueryable<T>

System.Linq.Enumerable  
Delegate based

System.Linq.Queryable  
Expression tree based

Objects

SQL

DataSets

Weitere...



complement  
getting ideas done

# Zwischenfazit

LINQ = Spracherweiterungen + Query Expressions

- LINQ kann mit allen erdenklichen Datenquellen arbeiten
  - Objekte, SQL, XML, Filesystem, SharePoint, LDAP
- LINQ baut auf anderen neuen Features auf
  - Type inference, object initialisers, anonymous types, extension methods, lambda expressions
  - IEnumerable<T> and IQueryable<T> from System.Linq
  - Query Expressions
- Lambdas können an Expression Trees gebunden werden



complement  
getting ideas done

# Microsoft LINQ – .NET meets data

## Agenda

C# 3.0 Features für LINQ

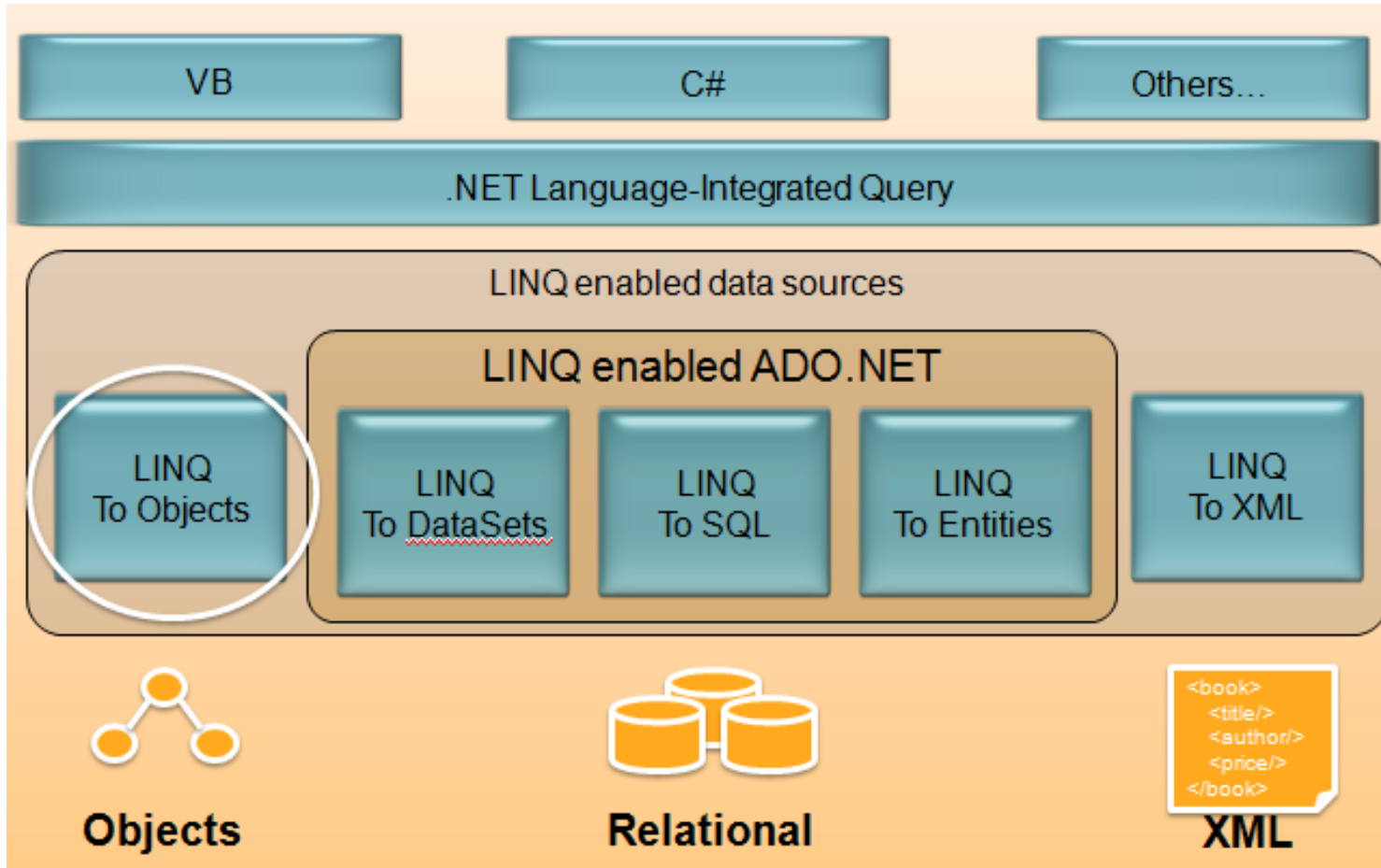
LINQ – Query Basics

LINQ for SQL, XML, ...



complement  
getting ideas done

# LINQ - Solution Domain



complement  
getting ideas done

# DEMO



complement  
getting ideas done

# LINQ to SQL im Zusammenspiel mit LINQ to XML

```
XElement londonCustomers =  
    new XElement("Customers", from c in db.Customers where  
        c.City == "London" select  
            new XElement("Customer",  
                new XAttribute("CustomerID", c.CustomerID),  
                new XElement("Name", c.ContactName),  
                new XElement("Phone", c.Phone)  
            );  
Console.WriteLine(londonCustomers);
```

```
<Customers>  
  <Customer CustomerID="AROUT">  
    <Name>Mark Harrington</Name>  
    <Phone>{171} 555-0188</Phone>  
  </Customer>  
  <Customer CustomerID="BSBEV">  
    <Name>Michelle Alexander</Name>  
    <Phone>{171} 555-0112</Phone>  
  </Customer>  
  <Customer CustomerID="CONSH">  
    <Name>Nicole Holliday</Name>  
    <Phone>{171} 555-0182</Phone>  
  </Customer>  
</Customers>
```



Complanet  
has done

# LINQ to SQL im Zusammenspiel mit LINQ to XML

```
<customerUpdates>
  <customerUpdate>
    <custid>ALFKI</custid>
    <phone>206-555-0103</phone>
  </customerUpdate>
  <customerUpdate>
    <custid>EASTC</custid>
    <phone>425-555-0143</phone>
  </customerUpdate>
</customerUpdates>
```

```
</
foreach (var cu in customerUpdates.Elements("customerUpdate"))
{
    Customer cust = db.Customers.First
        (c => c.CustomerID == (string)cu.Element("custid"));
    cust.Phone = (string)cu.Element("phone");
}

db.SubmitChanges();
```



complement

getting ideas done

# Referenzen, weiterführende Links

The LINQ Project @ Microsoft:

- <http://msdn2.microsoft.com/en-us/netframework/aa904594.aspx>

LINQ auf MSDN Library:

- [http://msdn2.microsoft.com/en-us/library/bb397926\(VS.90\).aspx](http://msdn2.microsoft.com/en-us/library/bb397926(VS.90).aspx)

Daniel Moth Blog (Microsoft):

- <http://www.danielmoth.com/Blog/>

Virtual Labs zu LINQ auf msdn.microsoft.com

- <http://msevents.microsoft.com/CUI/SearchDisplay.aspx?culture=en-US&tgtAudHero=2#eventType=4;culture=en-US;sortKey=;sortOrder=;pageEvent=false;hdnInitialCount=;kwdAny=linq;searchcontrol=yes>



complement  
getting ideas done

# Fragen ?



## Kontakt:

**Thomas Hemmer**

complement AG

Südwestpark 92

90449 Nürnberg

+49 911 255097620

Thomas.hemmer@complement.de



**complement**  
getting ideas done