

LINQ – Teil 1

Ingo Köster

Diplom-Informatiker (FH)

LINQ

- › Language Integrated Query
- › LINQ verwendet Syntax ähnlich zu SQL
- › Verwendet
 - › Lambda-Ausdrücke
 - › Typinferenz
 - › Objektinitialisierer
 - › Anonyme Typen
 - › Erweiterungsmethoden
 - › Nur bei Extension Method Syntax

Typinferenz – Schlüsselwort var

- › Bei der Deklaration von Variablen muss der Datentyp angegeben werden
 - › z.B. `int zahl;`
- › Erlaubt es, eine Variable mit dem Schlüsselwort `var` zu deklarieren
- › Datentyp muss nicht angegeben werden
 - `var zahl = 5;`
 - `var person = new Person();`

Typinferenz – Schlüsselwort var

- › Variablen mit dem Schlüsselwort var sind implizit typisierte Variablen
- › Der Compiler wählt den am besten passenden Datentyp aus dem Ausdruck rechts vom Gleichheitszeichen aus
- › Beim Anlegen von Variablen mit dem Schlüsselwort var müssen diesen ein Wert zugewiesen werden
 - › Die Anweisung `var einWert;` ist daher nicht zulässig
- › Implizit typisierte Variablen können nicht als Übergabeparameter verwendet werden

Objektinitialisierer

- › Objekt erzeugen und anschließend Werte setzen

```
Circle einKreis = new Circle();  
einKreis.PositionX = 23;  
einKreis.PositionY = 42;  
einKreis.Radius = 7;
```

- › Mit Objektinitialisierer

```
Circle einKreis = new Circle()  
{  
    PositionX = 23,  
    PositionY = 42,  
    Radius = 7  
};
```

Anonymer Typ

- › Objekte erstellen, ohne den Typ explizit anzugeben
- › Dabei wird eine neue Klasse erstellt
 - › Ein anonymer Typ

```
var einObj = new { Name = "Franz", Ort = "Bochum" };
```

- › Anonyme Typen können nicht verändert werden

Anonymer Typ

- › Array mit anonymen Typen anlegen

```
var personen = new[  
{  
    new { Name = "Tim", Ort = "Essen" },  
    new { Name = "Tom", Ort = "Dortmund" }  
};
```

- › Namen und Anzahl der Properties müssen gleich sein

LINQ Beispiel

- › Alle Personen Objekte mit einem Alter über 30
- › Ergebnis ist eine Liste mit anonymen Typen

```
var pers = from einePerson in allePersonen
            where einePerson.Alter > 30
            select new
            {
                einePerson.Nachname,
                einePerson.Alter
            };
```


LINQ Aufbau

```
var pers = from einePerson in allePersonen  
            where einePerson.Alter > 30  
            select new { einePerson.Nachname, einePerson.Alter };
```

› Typinferenz

› Lambda-Ausdruck

› Anonymer Typ mit Objektinitialisierer

LINQ Aufbau

```
var pers = from einePerson in allePersonen
            where einePerson.Alter > 30
            select new { einePerson.Nachname,
                          einePerson.Alter };
```



```
var pers = allePersonen.Where(einePerson =>
                              einePerson.Alter > 30).Select(einePerson => new {
                              einePerson.Nachname, einePerson.Alter });
```

LINQ Abfrageoperatoren Teil 1

Operatortyp	Operator
Aggregatoperatoren	Aggregate, Average, Count, LongCount, Min, Max, Sum
Casting-Operatoren	Cast, OfType, ToArray, ToDictionary, ToList, ToLookup, ToSequence
Elementoperatoren	DefaultIfEmpty, ElementAt, ElementAtOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
Gleichheitsoperatoren	EqualAll
Sequenzoperatoren	Empty, Range, Repeat
Gruppierungsoperatoren	GroupBy
Join-Operatoren	Join, GroupJoin

LINQ Abfrageoperatoren Teil 2

Operatortyp	Operator
Sortieroperatoren	OrderBy, ThenBy, OrderByDescending, ThenByDescending, Reverse
Aufteilungsoperatoren	Skip, SkipWhile, Take, TakeWhile
Quantifizierungsoperatoren	All, Any, Contains
Restriktionsoperatoren	Where
Projektionsoperatoren	Select, SelectMany
Set-Operatoren	Concat, Distinct, Except, Intersect, Union