BASTA!

.NET, VISUAL STUDIO & MORE

Oliver Sturm | Developer Express

Extensible LINQing

Microsoft Most Valuable Professional

Wer ist eigentlich dieser Oliver Sturm?

Ich bin

- Interessiert an Programmiersprachen, Datenbanken und vielen anderen Sachen
- Microsoft MVP für C#
- Autor der C# Corner im dot.net magazin

Ich

- Arbeite für Developer Express als Technical Evangelist und Lead Program Manager
- Blogge unter http://www.sturmnet.org/blog

Sie sollten

- Mir Emails schicken an oliver@sturmnet.org

Agenda

- Wie funktioniert LINQ?
- Erweiterungspunkte
- LINQ mit remote Datenquellen
- IQueryable, IQueryProvider

W

```
from s in myList
where s.Name == “ASpecificS”
select s.ID;
```

- Diese Hilfsmethoden sind als Extension Methods implementiert.
- Spezielle sprachspezifische Where werden vom Co
- E
  g
  verwend

Mithilfe von Extension Methods kann der Programmierer neue Methoden für existierende Typen aufrufen

Erweiterungspunkte

- Eigene Query-Operatoren
  - Andere Datentypen oder Business-Logik
- Standard-Operatoren neu implementieren
  - “Query expression pattern” ist Bestandteil der C# 3.0 Specs
  - Basiert auf Standardmechanismen für Extension Methods
- Implementation von Interfaces

Einfache Erweiterungen

Demo

LINQ vs. LINQ to Something

Demo

Das Sammeln der

Viele Daten,
viel Zeit…

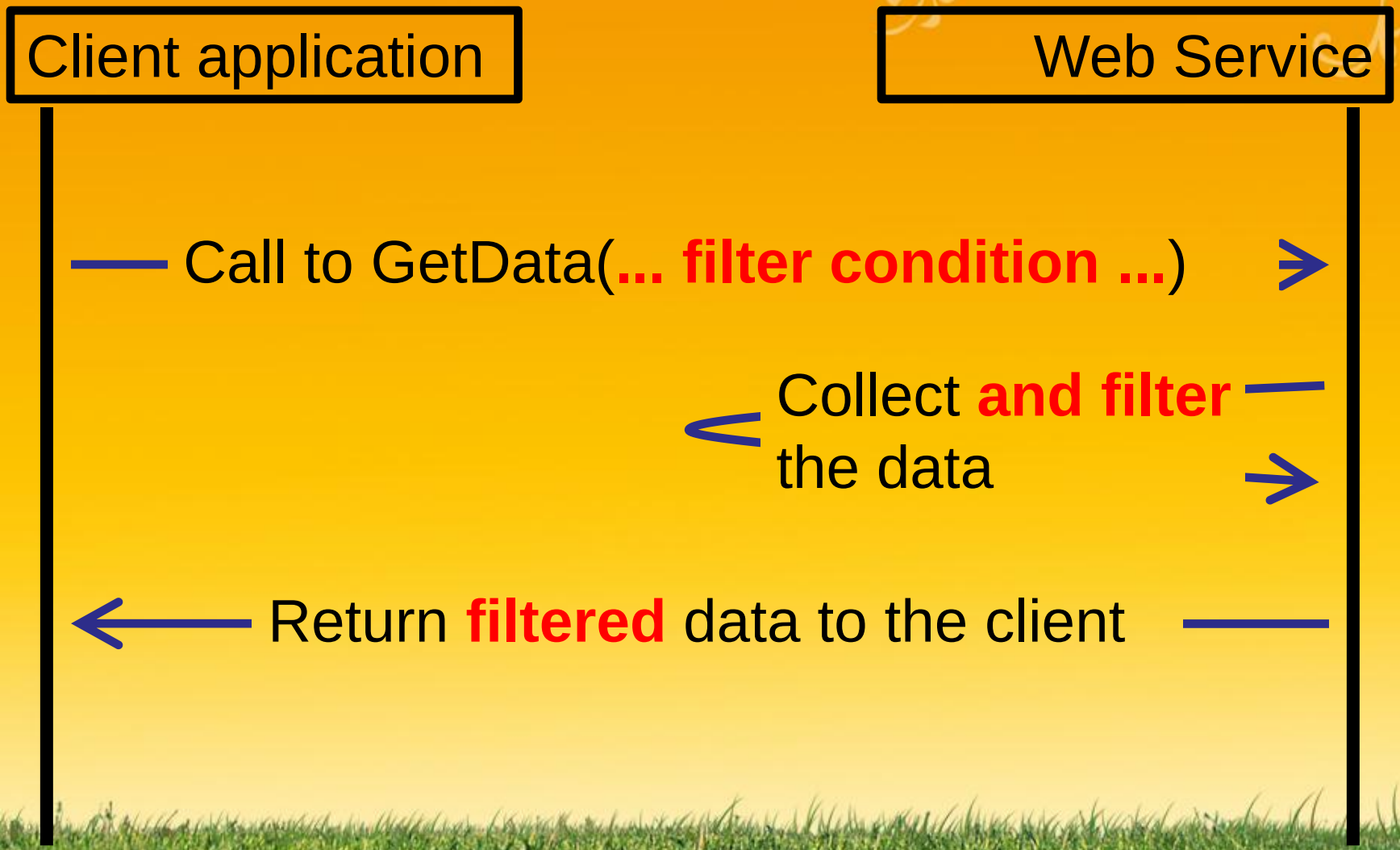
rtsWith("T")

Objekte
ndere

from er y in
dataWe Service.GetData()

Filter the data

Viel effizienter…



Wichtige Interfaces

```
from entry in dataWebService.GetData()
     where entry.Name.StartsWith("T")
     select entry;
```

- Die Datenquelly muss `IEnumerable<T>` implementieren
- Wenn eine `where` clause vorhanden ist und die Datenquelle `IQueryable<T>` implementiert, werden die Methoden dieses Interfaces aufgerufen
- IQueryable<T> liefert IQueryProvider, mit dessen Methoden die Query verarbeitet wird

Wichtige Methoden/Properties

```
public interface IQueryable : IEnumerable {
        Type ElementType { get; }
        Expression Expression { get; }
        IQueryProvider Provider { get; }
}
```

Wichtige Methoden/Properties

```
public interface IQueryProvider {
        IQueryable CreateQuery(Expression
                expression);
        IQueryable<TElement>
                CreateQuery<TElement>(Expression
                expression);
        object Execute(Expression expression);
        TResult Execute<TResult>(Expression
                expression);
}
```

Wichtige Methoden/Properties

```
public interface IEnumerable<T> :
        IEnumerable {
        IEnumerator<T> GetEnumerator( );
}
```

Ablauf einer Abfrage

1. IQueryable<T> wird von der Datenquelle abgefragt
2. IQueryable.Provider liefert den Provider
3. IQueryable.Expression liefert die Basis-Expression
4. IQueryProvider.CreateQuery<T> bekommt die neue Expression und liefert IQueryable<T> zurück
5. IEnumerable<T>.GetEnumerator liefert die Result-Collection

Wozu all die anderen Methoden in den Interfaces?

- Microsoft hat offenbar diese Interfaces extrahiert, nachdem die Implementation bereits existierte.
- Hier ist ein Pattern beschrieben, das diese Methoden nutzt:
  http://msdn2.microsoft.com/en-us/library/bb546158.aspx
  (Reminder: find and ~~kill~~ tickle inventor of these URLs)
- Implementation in mehreren Klassen ist möglich, im Beispiel ist alles in einer Klasse

Implementation der Interfaces

Demo

Zusammenfassung

- LINQ ist ein erweiterbares System
- Abhängig von den Anforderungen können verschiedene Erweiterungspunkte genutzt werden
- Es ist möglich, praktisch jede Datenquelle anzubinden

Vielen Dank

Bitte kontaktieren Sie mich jederzeit zum Inhalt dieser Session

oliver@sturmnet.org