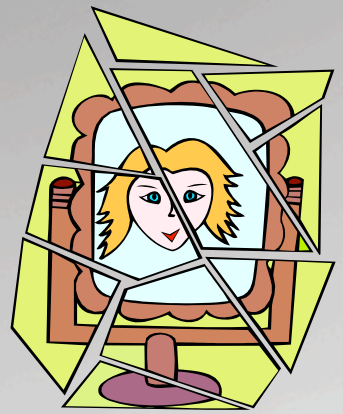
Oliver Sturm, DevExpress

Extensible LINQing

Who's that Oliver Sturm guy, anyway?



I am

- Interested in programming languages, databases and a whole bunch of other things
- Microsoft MVP for C#

I do

- Work for Developer Express as a Technical Evangelist and Lead Program Manager
- Blog at http://www.sturmnet.org/blog

You should

- Email me at oliver@sturmnet.org

Agenda

- How does LINQ work?
- Extensibility points
- LINQ with remote data sources
- IQueryable, IQueryProvider

How

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

- These
  Methods
- Special language
  method calls by th
- There

Through extension methods,
new functionality can be invoked
on existing types

Extensibility points

- Custom query operators
  - New data types or business logic
- Re-implement standard operators
  - “Query expression pattern” is part of the C# 3.0 Specs
  - Based on standard Extension Methods mechanisms
- Implementation of Interfaces

Simple extensions

Demo

LINQ vs. LINQ to Something

Demo

Oliver Sturm, DevExpress

W

Collecting data

Lots of data,
lots of time..

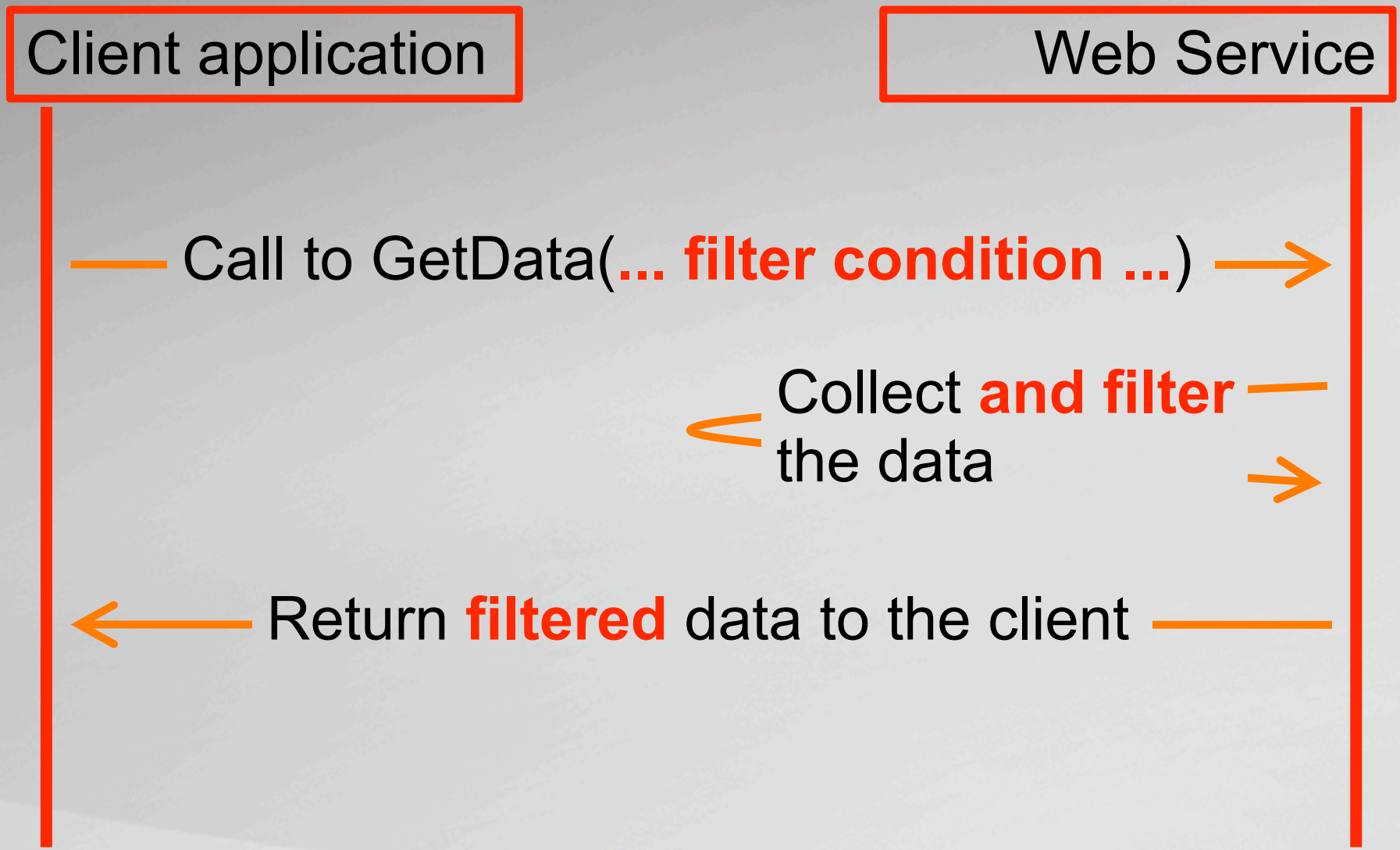
rtsWith("T")

ting objects,
st

from e y in
dataWe Service.GetData()

Filter the data

Much more efficient…



Important interfaces

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

- The data source must implement `IEnumerable<T>`
- If there is a `where` clause and the data source implements `IQueryable<T>`, `methods from that interface are called`
- IQueryable<T> returns IQueryProvider, which is used to process the query

Important methods/properties

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

Important methods/properties

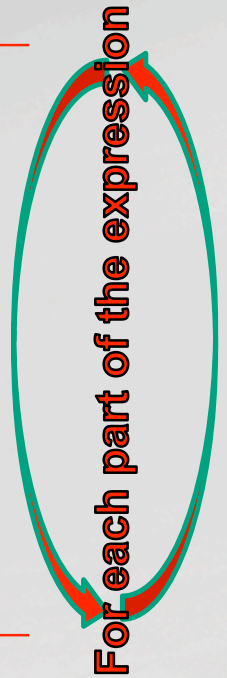
```
public interface IQueryProvider {
   IQueryable CreateQuery(Expression
     expression);
   IQueryable<TElement>
     CreateQuery<TElement>(Expression
     expression);
   object Execute(Expression expression);
   TResult Execute<TResult>(Expression
     expression);
}
```

Important methods/properties

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

Running a query

1. IQueryable<T> is queried from the data source
2. IQueryable.Provider returns a provider
3. IQueryable.Expression returns the base expression
4. IQueryProvider.CreateQuery<T> is passed the new expression and returns IQueryable<T>
5. IEnumerable<T>.GetEnumerator returns the result collection



What are all the other interface methods good for?

- Apparently, these interfaces have been extracted after implementation
- Here's a description of a pattern that uses these methods:
  http://msdn2.microsoft.com/en-us/library/bb546158.aspx
  (Reminder: find and ~~kill~~ tickle inventor of these URLs)
- Implementation in multiple classes is possible, in the examples everything is in one class

Implementation of the interfaces

Demo

Summary

- LINQ is an extensible system
- Different extensibility points are available depending on requirements
- It is possible to attach almost any data source

Thank you!

Please feel free to contact me about this session anytime!

oliver@sturmnet.org