Introduction To LINQ

An Overview Of Language Integrated Query

pluralsight
see what you can learn

Overview

- **Why LINQ?**
- **What problems will LINQ solve?**
- **LINQ Query Expressions**
- **A summary of major LINQ technologies**
  - LINQ To Objects
  - LINQ To XML
  - LINQ To SQL
  - LINQ To Entities

Some C# History

Integrating data queries into C# has been a goal for years.

"LINQ" on the whiteboard

```
sequence<Employee> scotts =
    employees.where(Name == "Scott");
```

"LINQ" in C# 2.0

```
IEnumerable<Employee> scotts =
    EnumerableExtensions.Where(employees,
          delegate(Employee e)
          {
              return e.Name == "Scott";
          });
```

The Data Impedance Mismatch

- **Typical data access challenges**
  - Different data types
  - Different data sources
  - Relationships versus hierarchies
  - Limited Intellisense or language support for data access

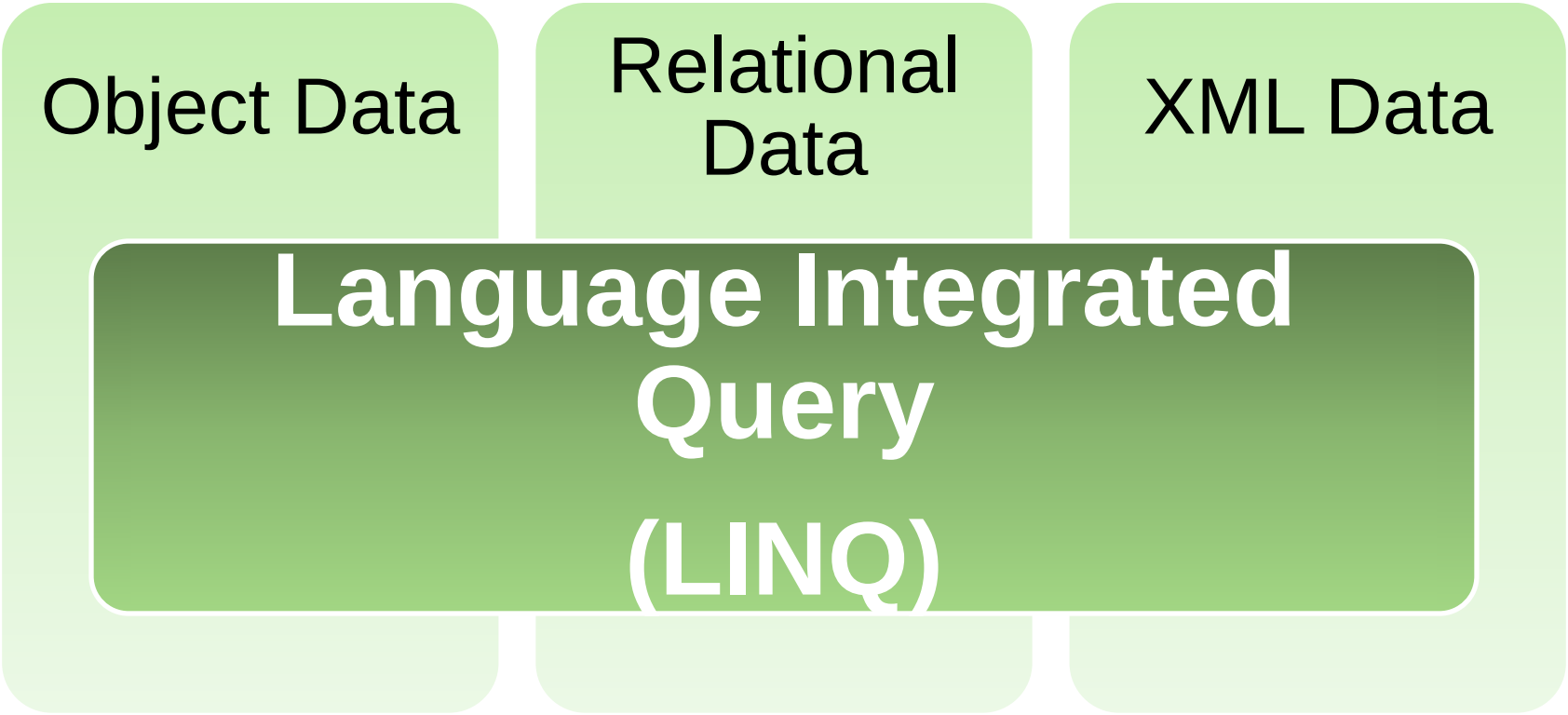
Manipulating Data

- **The data source *generally* determines the API and tools to use**

Object Data	Relational Data	XML Data
Generics	ADO.NET	XmlDocument
Algorithms	SQL	XPath / XSLT

Enter LINQ

- **LINQ provides general-purpose query facilities**



Standard Query Operators

- Defined in the System.Linq namespace
- Work on any IEnumerable<T>
- CLS compliant (generics required)

```
string[] instructors = { "Aaron", "Fritz", “Scott", “Keith" };

IEnumerable<string> query = from s in instructors
                            where s.Length == 5
                            orderby s descending
                            select s;
foreach (string name in query)
{
    Console.WriteLine(name);
}
```

LINQ Query Expressions

- **The same standard query operators work everywhere**
  - Objects
  - Relational data
  - XML data
- **Over 50 operators defined**
  - Filtering
  - Projection
  - Joining
  - Partitioning
  - Ordering
  - Aggregating
- **Similar to SQL**
  - Select, From, Where, OrderBy, GroupBy

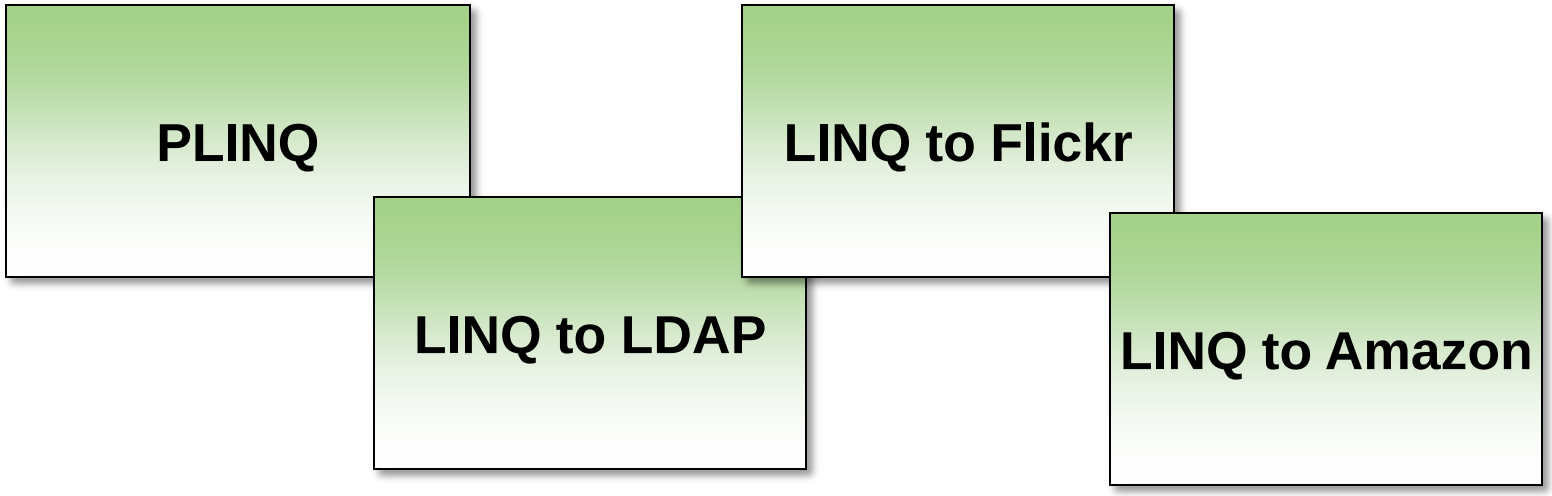
Language Integration

- **Languages and tools are aware of query expressions**
  - Rich metadata
  - Static type checking
  - Intellisense

```
string[] instructors = { "Aaron", "Fritz", "Keith", "Scott" };

// error assigning IEnumerable<int> to IEnumerable<string>
IEnumerable<string> query =
        from n in instructors
        where n.Length == 5
        orderby n descending
        select n.Length; // creating IEnumerable<int>
```

LINQ's Extensibility

- **Operator extensibility**
  - We can implement our own operators
  - We can override standard operators for our own types
- **Provider extensibility**
  - A LINQ Provider is a gateway to query-able types



LINQ to Objects

- **Replace foreach loops and other iterative code with LINQ expressions**

```
IEnumerable<Process> processList =
                from p in Process.GetProcesses()
                where String.Equals(p.ProcessName, "svchost")
                orderby p.WorkingSet64 descending
                select p;
```

Deferred Execution

- **Query expression does not execute until we access the result**
  - Treat query expressions as data
  - Build composable queries

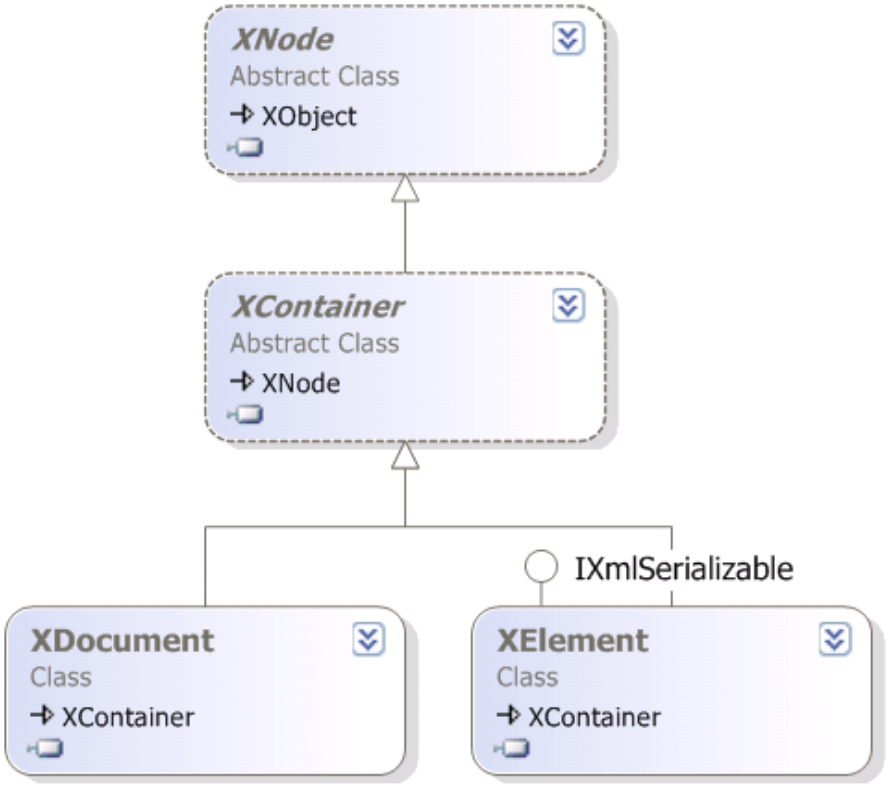
```
IEnumerable<Process> processList =
    from p in Process.GetProcesses()
    where String.Equals(p.ProcessName, "svchost")
    orderby p.WorkingSet64 descending
    select p;

foreach (Process p in processList)
{
    Console.WriteLine("{0,10} ({1,4}) : {2,15:N0}",
            p.ProcessName, p.Id, p.WorkingSet64);
}
```

Define Query

Execute Query

LINQ to XML

- Not just another XML API
- XElement is the core class in the System.Xml.Linq namespace



Functional Construction with XElement

- **Concise, readable XML creation**

```
XElement instructors =
    new XElement("instructors",
        new XElement("instructor", "Aaron"),
        new XElement("instructor", "Fritz"),
        new XElement("instructor", "Keith"),
        new XElement("instructor", "Scott")
    );
```

instructors.toString();

```
<instructors>
  <instructor>Aaron</instructor>
  <instructor>Fritz</instructor>
  <instructor>Keith</instructor>
  <instructor>Scott</instructor>
</instructors>
```

Queries with LINQ to XML

- **Same standard query operators**
- **XML specific extensions applied**

```
XElement instructors =
    new XElement("instructors",
        new XElement("instructor", "Aaron"),
        new XElement("instructor", "Fritz"),
        new XElement("instructor", "Keith"),
        new XElement("instructor", "Scott")
    );
```

```
int numberOfScotts =
    (from i in instructors.Elements("instructor")
     where i.Value == "Scott"
     select i).Count();
```

LINQ to SQL

- LINQ to SQL is a simple object-relational mapper (OR/M)
- Works with SQL Server 2000 / 2005 / 2008
- Translates query expression into T-SQL

```
IEnumerable<Customer> customers =
    from c in context.Customers
    where c.Country == "France"
    orderby c.CustomerID ascending
    select c;
```

```
SELECT [t0].[CustomerID], [t0].[CompanyName], [t0].[ContactName],
       [t0].[ContactTitle], [t0].[Address], [t0].[City],
       [t0].[Region], [t0].[PostalCode], [t0].[Country],
       [t0].[Phone], [t0].[Fax]
FROM [dbo].[Customers] AS [t0]
WHERE [t0].[Country] = @p0
ORDER BY [t0].[CustomerID]
```

Mapping Objects to Tables

Mapping with attributes

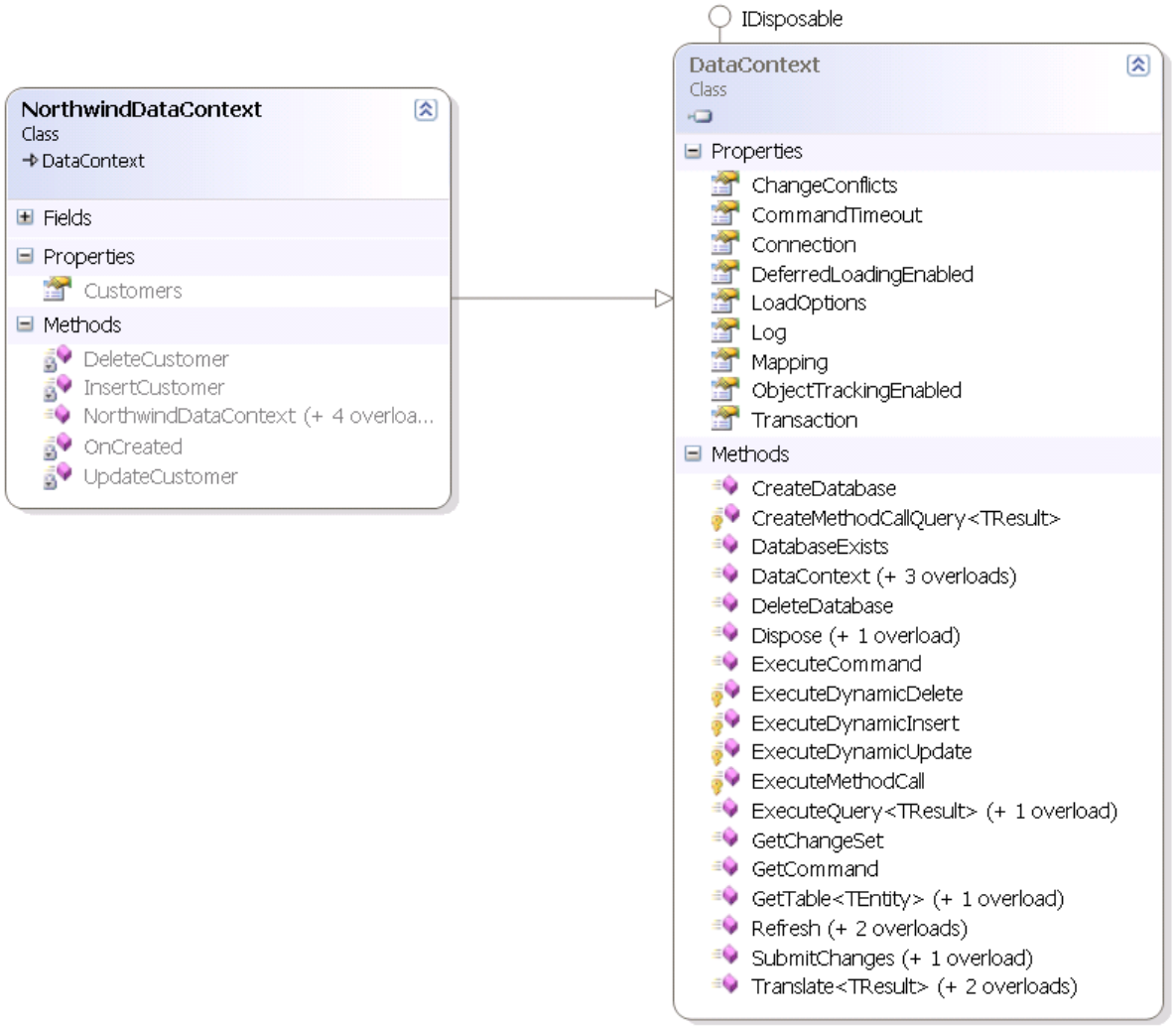
```
[Table(Name = "dbo.Customers")]
public partial class Customer
{
    [Column]
    public string CustomerID
    {
        // ...
    }
    // ...
}
```

1:1 mapping from class to table

Mapping with external XML

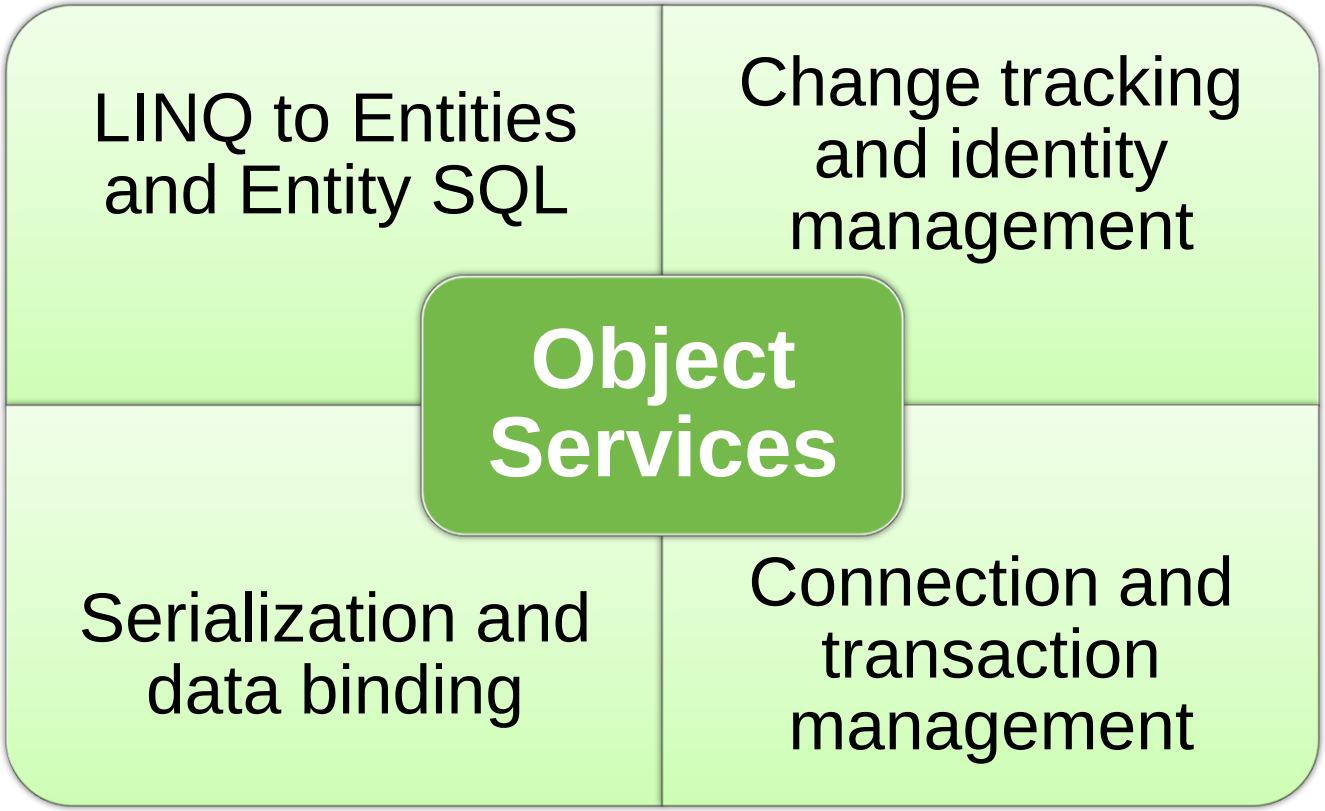
```
<Table Name="dbo.Customers" Member="Customers">
  <Type Name="Customers">
    <Column Name="CustomerID" Member="CustomerID"  />
    <Column Name="CompanyName" Member="CompanyName"  />
    <Column Name="ContactName" Member="ContactName"  />
    <Column Name="ContactTitle" Member="ContactTitle"
/>
    <Column Name="Address" Member="Address"  />
    <!-- .. -->
  </Type>
</Table>
```

LINQ to SQL – The DataContext



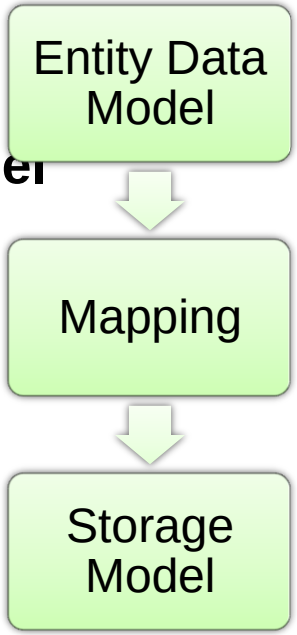
Entity Framework

- **Entity Framework provides a rich layer of *object data services***



Entity Framework Conceptual Model

- **Program against a conceptual entity data model**
- **Entity model abstracted from storage model (database schema) by a mapping layer**
  - Supports m:n mapping
  - Supports several inheritance schemes
- **LINQ to Entities works against conceptual model**
  - ObjectContext is the gateway to entities



Summary

- LINQ – new query capabilities for the .NET Platform
- LINQ standard operators work against IEnumerable<T>
- LINQ providers open a wide variety of disparate data sources
- LINQ to XML introduces a new XML API for .NET
- LINQ to SQL / LINQ to Entities provide object/relational mapping features

References

- LINQ: .NET Language Integrated Query – Don Box, Anders Hejlsberg (http://msdn2.microsoft.com/en-us/library/bb308959.aspx)
- .NET Language Integrated Query For XML Data – Michael Champion (http://msdn2.microsoft.com/en-us/library/bb308960.aspx)