Jim Fawcett
CSE681 – Software Modeling and Analysis
Fall 2013

LINQ

References

- Pro C# 5.0 and the .Net 4.5 Framework, Sixth Edition, Andrew Troelsen
- C# in a Nutshell, 5th Edition, Albahari & Albahari
- Programming C# 5.0, Ian Griffiths
- http://msdn.microsoft.com/en-us/library/vstudio/bb397926.aspx
- http://en.wikipedia.org/wiki/Language_Integrated_Query
- http://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b
- http://msdn.microsoft.com/en-us/library/system.xml.linq.aspx

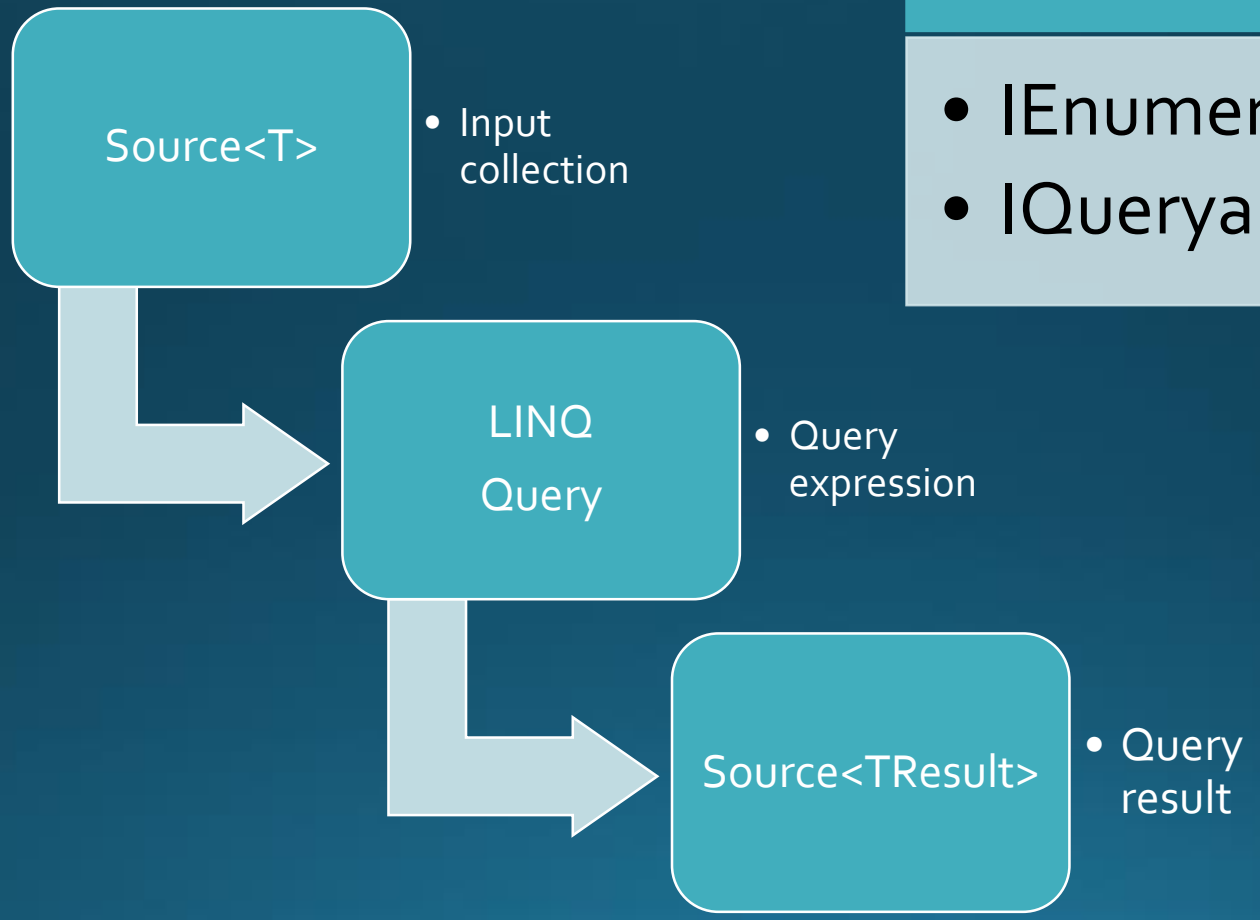
Language INtegrated Query (LINQ)

- LINQ is an extension to C#, VB, and F# and has been ported to other languages as well.
- The extension is based on:
  - LINQ Providers
    - Query Operators
      - Extension Methods
  - Language Features
    - Anonymous types using var keyword
    - Lambda expressions
    - delegates
    - Expression Trees
      - SQL Provider

LINQ Providers

- A LINQ provider publishes:
  - library of Query operators which are implemented as extension methods
    - Each kind of data requires its own implementation
- The .Net Framework has several LINQ providers and facilities:
  - LINQ to Objects
    - Works with IEnumerable<T> collections
  - LINQ to SQL
    - Works with IQueryable<T> data
- LINQ to XML is not an LINQ provider
  - It is an API for creating and parsing XML documents
  - Works with XDocument class
- Others: LINQ to Entities, WCF Data Services

Query Operations



Source

- IEnumerable<T>
- IQueryable<T>

LINQ Queries – Extension Syntax

- Here is a typical query using extension method syntax:

```
string[] cars = { "Boxter", "Mini Cooper", "Mustang", "Camaro", "Miata", "Z4" };

IEnumerable<string> query = cars
    .Where(n => n.StartsWith("M"))
    .OrderBy(n => n)
   .Select(n => n);
```

LINQ Queries – Query Syntax

- Here's the same query using Query Syntax:

```
var query =
    from n in cars
    where n.StartsWith("M")
    orderby n

    select n;
```

- Note the difference in capitalization and use of . notation

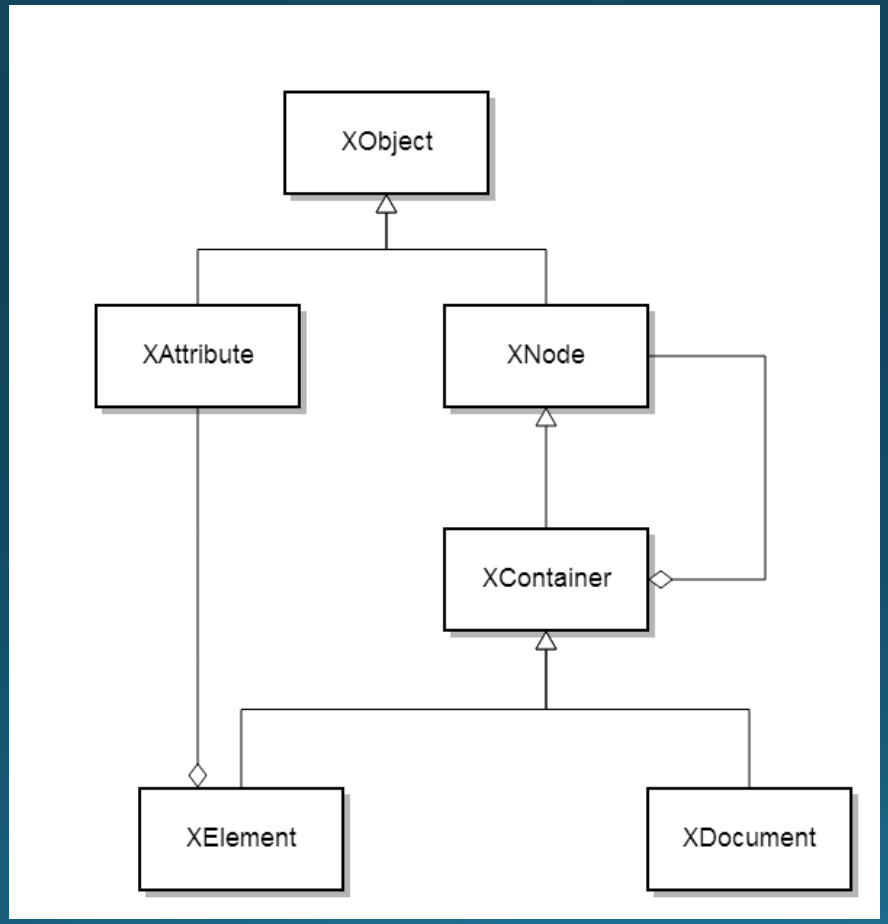
Query Operators

- Restriction: Where
- Projection: Select, SelectMany
- Partitioning: Take, Skip, TakeWhile, SkipWhile
- Ordering: OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse
- Grouping: Groupby
- Set: Distinct, Union, Intersect, Except
- Conversion: ToArray, ToList, ToDictionary, OfType
- Element: First, FirstOrDefault, ElementAt
- Generation: Range, Repeat,
- Quantifiers: Any, All
- Aggregate: Count, Sum, Min, Max, Average, Aggregate
- Miscel: Concat, EqualAll
- Join: Cross Join, Group Join, Cross Join with Group Join, Left Outer Join
- Execution: Deferred Execution, immediate Execution, Query Reuse

LINQ to XML

- LINQ to XML is not based on a LINQ Provider of extension methods.
- LINQ to XML uses the XDocument class that builds and interrogates an XML parse tree.
  - Many of its methods return IEnumerable<T> objects
  - That supports LINQ to Objects queries
- The most important classes are:
  - XDocument
  - XElement
  - XAttribute

LINQ to XML Class Diagram



XElement and XDocument

- We can build an XElement instance from an XML string:

```
XElement elem = XElement.Parse(xmlString);
```

- We can build an XDocument instance from a file containing XML:

```
XDocument doc = XDocument.Load(@"..\aFile.xml");
```

- You can serialize an Xdocument to a File, Stream, TextWriter, or XmlWriter:

```
doc.save("../anXmlFile.xml");
```

Query Functions

• XElement.Elements()	returns child XElements
• XElement.Elements(“name”)	returns child XElements with tag “name”
• XElement.Decendents()	returns decendent Xelements
• XElement.DecendentNodes()	returns decendent XNodes
• XNode.Parent	returns parent Xelement
• XNode.Document	returns the XDocument element
• XNode.Ancestors()	returns XElement ancestors