Introduction To LINQ

Data Access

Data != Objects

(Object Relational Mapping)

Build database-like search functionality into the .NET Framework that works on any data: object, XML, SQL, custom types…

LINQ: Language INtegrated Query
C# 3.0 Language Innovations

```csharp
var contacts = 
    from c in customers
    where c.State == "WA"
    select new { c.Name, c.Phone };

var contacts =
    customers
    .Where(c => c.State == "WA")
    .Select(c => new { c.Name, c.Phone });
```

LINQ To Objects

<table>
<thead>
<tr>
<th>Restriction</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projection</td>
<td>Select, SelectMany</td>
</tr>
<tr>
<td>Ordering</td>
<td>OrderBy, ThenBy</td>
</tr>
<tr>
<td>Grouping</td>
<td>GroupBy</td>
</tr>
<tr>
<td>Joins</td>
<td>Join, GroupJoin</td>
</tr>
<tr>
<td>Quantifiers</td>
<td>Any, All</td>
</tr>
<tr>
<td>Partitioning</td>
<td>Take, Skip, TakeWhile, SkipWhile</td>
</tr>
<tr>
<td>Sets</td>
<td>Distinct, Union, Intersect, Except</td>
</tr>
<tr>
<td>Elements</td>
<td>First, Last, Single, ElementAt</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Count, Sum, Min, Max, Average</td>
</tr>
<tr>
<td>Conversion</td>
<td>ToArray, ToList, ToDictionary</td>
</tr>
<tr>
<td>Casting</td>
<td>OfType&lt;T&gt;, Cast&lt;T&gt;</td>
</tr>
</tbody>
</table>
LINQ Basics

- **Query Operators** can be used against any .NET collection (IEnumerable<T>)
  - Built-in examples: Select, Where, GroupBy, Join, etc.
  - Extensibility model supports adding/replacing them

- **Query Expressions** can operate on information sources and apply query operators against them to return IEnumerable<T> sequences

LINQ to SQL

- Language integrated data access
  - Maps tables and rows to classes and objects
  - Builds on ADO.NET and .NET Transactions

- **Mapping**
  - Encoded in attributes or external XML file
  - Relationships map to properties

- **Persistence**
  - Automatic change tracking
  - Updates through SQL or stored procedures
LINQ to SQL
Accessing data today

SqlConnection c = new SqlConnection(…);
c.Open();
SqlCommand cmd = new SqlCommand(@"SELECT c.Name, c.Phone
FROM Customers c
WHERE c.City = @p0");
cmd.Parameters.AddWithValue("@p0", "London");
Datareader dr = c.ExecuteReader(cmd);
while (dr.Read()) {
  string name = dr.GetString(0);
  string phone = dr.GetString(1);
  DateTime date = dr.GetDateTime(2);
}
dr.Close();

LINQ to SQL
Accessing data with LINQ

public class Customer { … }

public class Northwind : DataContext
{
  public Table<Customer> Customers;
}

Northwind db = new Northwind(…);
var contacts =
from c in db.Customers
where c.City == "London"
select new { c.Name, c.Phone };
LINQ to SQL Mapping

Database → DataContext
Table → Class
View → Class
Column → Field / Property
Relationship → Field / Property
Stored Procedure → Method

LINQ to SQL Architecture

```
from c in db.Customers
    where c.City == "London"
select c.CompanyName
```

```
db.Customers.Add(c1);
c2.City = "Seattle";
db.Customers.Remove(c3);
```

```
SELECT CompanyName
FROM Cust
WHERE City = 'London'
```

```
INSERT INTO Cust ...
UPDATE Cust ...
DELETE FROM Cust ...
```
Key Points

- Flexible mapping
  - “Classes first” or “data first”, attributes or mapping file
- DataContext
  - Strongly typed database connection
- Entity classes
  - Identity mapping and change tracking
- Relationships
  - One-to-one, one-to-many

LINQ to SQL

- Access relational data as strongly typed objects
- Language integrated query
- Works with existing infrastructure
- Unified query and transform of objects, relational, XML
Resources

- The LINQ Project on MSDN (http://msdn2.microsoft.com/en-us/netframework/aa904594.aspx)
- Charlie Calvert’s Blog (http://blogs.msdn.com/charlie/)
- Mike Taulty’s Blog (http://mtaulty.com/)
- LINQ in Action (http://linqinaction.net/)
- Scott Gu’s Blog (http://weblogs.asp.net/scottgu/default.aspx)