

LINQ

(Temel Kavramlar)

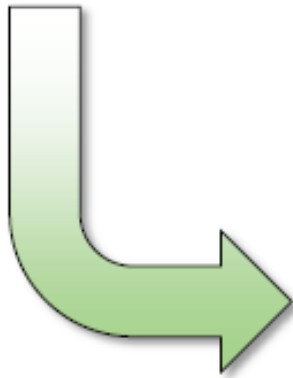
Ele Alınacak Başlıklar

- Temel Kavramlar
- Lambda İfadeleri (*Lambda Expressions)
- Query İfadeleri (*Query Expressions)
- Tür Çıkarsama (*Type Inference)
- Anonim Türler (*Anonymous Types)

Benzer Sorgular

C# 2.0'da var olan LINQ

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



Günümüzde aktif olarak kullanılan

```
var scotts =  
    from e in employees  
    where e.Name == "Scott"  
    select e;
```

Sorgu ifadelerini daha öz bir şekilde yazmak mümkündür.

Named method

```
IEnumerable<string> filteredList =  
    cities.Where(StartsWithL);  
  
public bool StartsWithL(string name)  
{  
    return name.StartsWith("L");  
}
```

Anonymous method

```
IEnumerable<string> filteredList =  
    cities.Where(delegate(string s)  
        { return s.StartsWith("L"); }));
```

Lambda Expression

```
IEnumerable<string> filteredList =  
    cities.Where(s => s.StartsWith("L"));
```

*Lambda İfadeleri

- İsimsiz metotları tanımlamak için öz bir yazım şekline sahiptir.
 - Delegate anahtar sözcüğünü barındırmaz.
 - Return anahtar sözcüğüne gereksinim duymaz.
 - Derleyici çoğu kez ifade içerisindeki türlerden çıkarsama yapar.
- Lambda operatörü => şeklinde gösterilir. (goes to)
 - Operatörün sol tarafı fonksiyonun imzasını belirtir.
 - Operatörün sağ tarafı ise ifade bloğu olarak belirtilir. (expression or statement block)

```
IEnumerable<string> filteredList =  
    cities.Where(s => s.StartsWith("L"));
```

- Lambda operatörü => şeklinde gösterilir. (goes to)
 - Operatörün sol tarafı fonksiyonun imzasını belirtir.
 - Operatörün sağ tarafı ise ifade bloğu olarak belirtilir. (expression or statement block)

Örnek

```
IEnumerable<string> filteredList =  
    cities.Where((string s) =>  
        { string temp = s.ToLower();  
          return temp.StartsWith("L");  
        }));
```

Bellek mi, VeriTabanı mı?

```
IEnumerable<string> filteredList =  
    cities.Where(s => s.StartsWith("L"));
```

- Sorgulanmak istenen veri topluluğunun her daim bellek üzerinde olma zorunluluğu yoktur.
- LINQ sorguları veritabanları üzerinde de rahatlıkla işletilebilir.

▪ IEnumerable<T> ve IQueryable<T>

LINQ
to
Objects

```
namespace System.Linq {  
    public static class Enumerable  
    {  
        public static IEnumerable<TSource> Where<TSource>(  
            this IEnumerable<TSource> source,  
            delegate Func<TSource, bool> predicate) ...  
    }  
    ...  
}
```

LINQ
to
SQL

```
namespace System.Linq {  
    public static class Queryable  
    {  
        public static IQueryable<TSource> Where<TSource>(  
            this IQueryable<TSource> source,  
            expression Expression<Func<TSource, bool>> predicate)  
            ...  
    }  
    ...  
}
```

- Extension metotlar programcıya standart operatörleri sunmaktadır.
 - Eğer istenirse standart operatörler yeniden tanımlanabilirler.
- Lambda ifadeleri,
 - Delegate şeklinde kullanılabileceği gibi
 - Expression ağacı şeklinde de kullanılabilir.

Yeterince
okunabilir/anlaşılabilir değil !

```
string[] cities = { "Boston", "Los Angeles",  
                    "Seattle", "London", "Hyderabad" };  
  
IEnumerable<string> filteredList =  
    cities.Where(s => s.StartsWith("L"));
```

*Query İfadeleri

```
string[] cities = { "Boston", "Los Angeles",  
                    "Seattle", "London", "Hyderabad" };  
  
IEnumerable<string> filteredCities =  
    from city in cities  
    where city.StartsWith("L") && city.Length < 15  
    orderby city  
    select city;
```

- Extension from sözcüğü ile başlayıp select veya group sözcüğü ile son bulur.
- Sözdizimi SQL sorgusuna oldukça benzer.

Derleyici query expressionları lambda ifadelerini içeren bir dizi metot eşleniklerine dönüştürür.

Type Inference

```
string[] cities = { "Boston", "Los Angeles",  
var "Seattle", "London", "Hyderabad" };
```

```
IEnumerable<string> filteredCities =  
    from city in cities  
    where city.StartsWith("L") && city.Length < 15  
    orderby city  
    select city;
```

```
IEnumerable<string> filteredCities =  
    cities.Where(c => c.StartsWith("L") && c.Length < 15)  
           .OrderBy(c => c)  
           .Select(c => c);
```

- Tür çıkarsamak için kullanılan var anahtar sözcüğü JavaScript'tekinden farklı olarak tip güvenlidir.

```
var name = "Scott";  
var x = 3.0;  
var y = 2;  
var z = x * y;  
  
// all lines print "True"  
Console.WriteLine(name is string);  
Console.WriteLine(x is double);  
Console.WriteLine(y is int);  
Console.WriteLine(z is double);
```

```
// ERROR: implicitly typed local variables must be initialized
```

```
var i;
```

```
// ERROR: Implicitly-typed local variables cannot have multiple  
declarators
```

```
var j, k = 0;
```

```
// ERROR: Cannot assign <null> to an implicitly-typed local variable
```

```
var n = null;
```

```
var number = "42";
```

```
// ERROR: Cannot implicitly convert type 'string' to 'int'
```

```
int x = number + 1;
```

*Anonim Türler

- İsimsiz sınıflar oluşturmak mümkündür.
- Oluşturulan isimsiz sınıflarda özellik ve başlangıç değerleri belirtilebilir.
- Oluşturulan isimsiz sınıflar, geri dönüş değeri veya parametre olarak kullanılamaz.

```
var employee = new {  
    Name = "Scott",  
    Department = "Engineering"  
};  
Console.WriteLine("{0}:{1}", employee.Name, employee.Department);
```

```
var processList =  
    from process in Process.GetProcesses()  
    orderby process.Threads.Count descending,  
             process.ProcessName ascending  
    select new  
    {  
        process.ProcessName,  
        ThreadCount = process.Threads.Count  
    };
```

```
Console.WriteLine("Process List");  
foreach (var process in processList)  
{  
    Console.WriteLine("{0,25} {1,4:D}",  
        process.ProcessName,  
        process.ThreadCount);  
}
```


LINQ
(Giriş)

Ele Alınacak Başlıklar

- LINQ to Objects
- LINQ to XML
- LINQ to SQL
- Entity Framework

Başlıkların detaylı bir incelemesi takip eden haftalarda gerçekleşecektir. !!!

- Kullanılan veri türü genellikle kullanılacak olan araçları ve API'yi de belirler.

Object Data

Generics

Algorithms

Relational Data

ADO.NET

SQL

XML Data

XmlDocument

XPath / XSLT

LINQ teknolojisi genel amaçlı sorgulama olanağı sağlar.

Object Data

Relational
Data

XML Data

**Language Integrated
Query
(LINQ)**

LINQ Operatörleri

```
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 Sorgu İfadeleri

- Bahsi geçen standart sorgu operatörleri bir çok ortamda geçerlidir.
 - Nesneler üzerinde
 - İlişkisel veritabanları üzerinde
 - XML üzerinde
- 50'nin üzerinde operator tanımlanmıştır.
 - Filtering
 - Projection
 - Joining
 - Partitioning
 - Ordering
 - Aggregating

```
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 to Objects

- Genellikle bellekteki nesneleri sorgulamak amacıyla kullanılır. (objects in memory)

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


Ertelenmiş Yürütme (Deferred Execution)

- Sorgu ifadeleri, sonuca ulaşıncaya kadar icra edilmezler.
- Sorgu ifadelerine (query expressions) veri gibi işlem görür.
- Değiştirilebilen sorgular oluşturmak mümkündür.

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

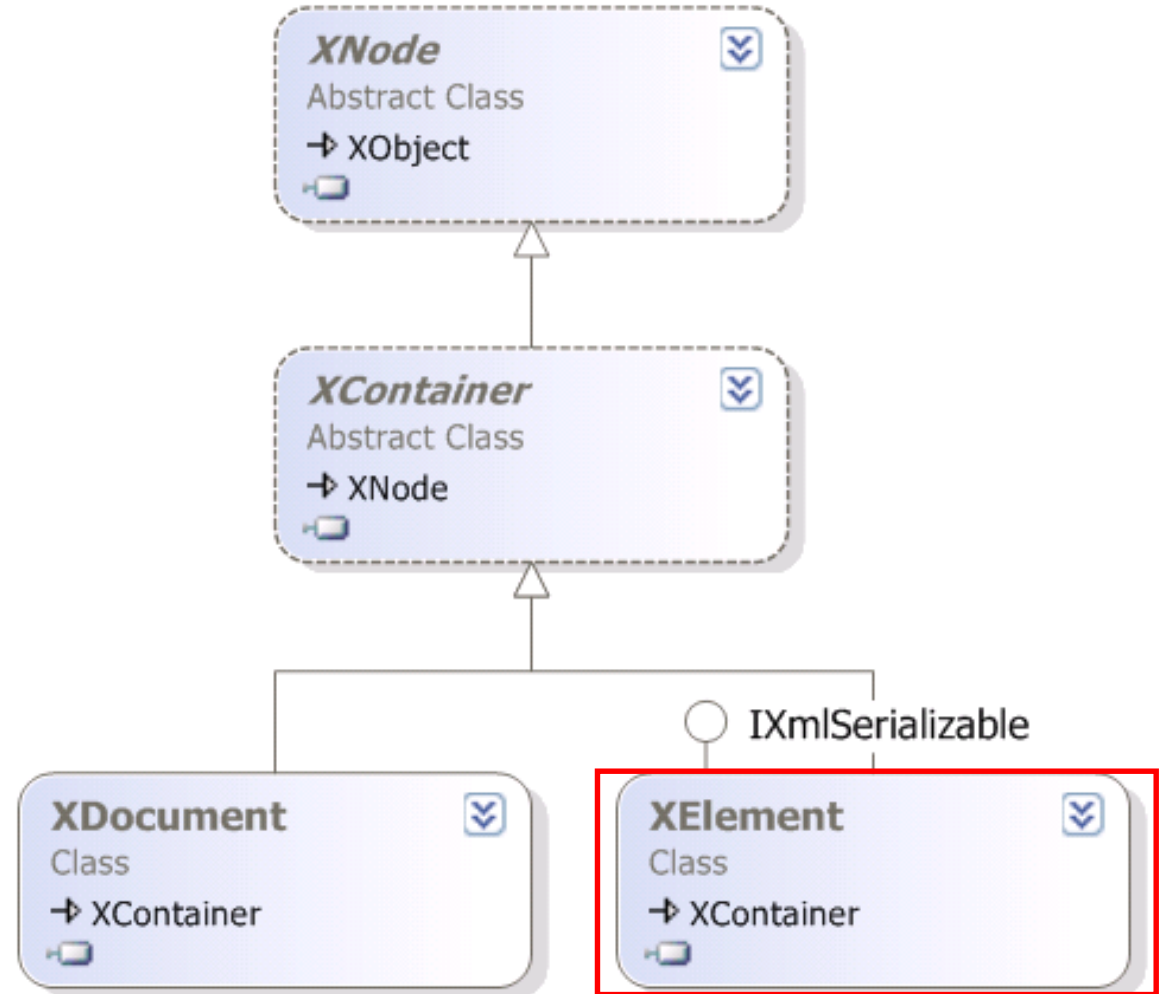
Define Query

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

Execute Query

*LINQ TO XML

- XML için yeni bir API değildir.
- XElement, bu yapı içerisinde kullanılan en temel sınıftır.
- XElement yardımıyla öz ve okunabilir XML dosyaları oluşturulabilir.



Örnek:

```
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>
```

```
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, object-relational mapper (OR/M)'dir.
- SQL Server ile çalışmaktadır.
- LINQ sorgularını SQL Server üzerinde çalıştırılacak olan T-SQL sorgularına dönüştürür.

LINQ Sorgusu

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

TSQL Sorgusu



```
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]
```

Bellekte bulunan nesnelerin veritabanındaki tablolarla eşleştirilmesi iki türlü olur.

- Attribute'lerin kullanımı veya
- Harici bir XML dosyası kullanılarak

Attribute Kullanarak Gerçekleştirilen Eşleştirme

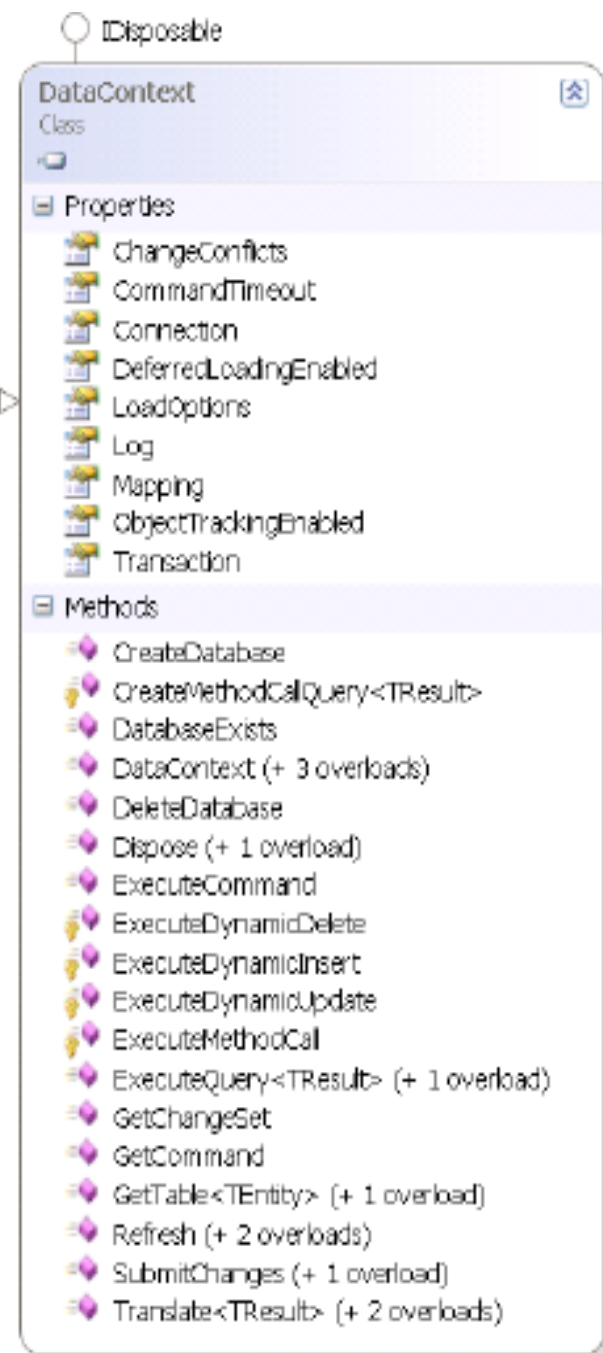
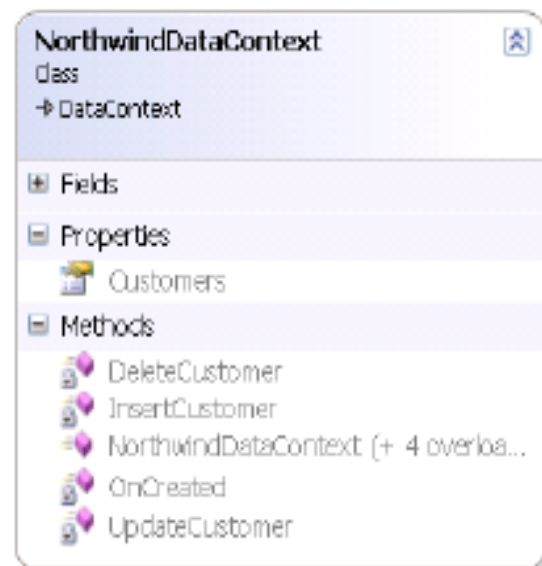
```
[Table(Name = "dbo.Customers")]  
public partial class Customer  
{  
    [Column]  
    public string CustomerID  
    {  
        // ...  
    }  
    // ...  
}
```

1:1 mapping from class to table

XML Dosyası Kullanılarak Gerçekleştirilen Eşleştirme

```
<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>
```

DataContext



Entity Framework

- Entity Framework, LINQ to SQL'den daha esnek bir yapıya sahiptir.
- Sadece SQL Server ile değil pek çok farklı veritabanı ile çalışabilir.
- Yine, LINQ sorgularını işletmek mümkündür.

