LINQ and C#

Scott Allen

OdeToCode.com - @OdeToCode

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";
          });
```

LINQ today

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

Creating Extension Methods

```
public static class StringExtensions
{
    static public double ToDouble(this string data)
    {
        double result = double.Parse(data);
        return result;
    }
}
```

First parameter of an extension method uses `this` modifier

```
string text = "43.35";
double data = text.ToDouble();
```

Can invoke static method with instance syntax

Lambda Expressions

```
IEnumerable<string> filteredList =
                cities.Where(StartsWithL);

public bool StartsWithL(string name)
{
    return name.StartsWith("L");
}
```

Named Method

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

Anonymous Method

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

Lambda Expression

Implicit Typing

```
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);
```

The compiler infers the type of the variable

Query Syntax

```
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;
```

Query syntax starts with `from`

Query ends with `select` or `group`

Summary

```
var query = developers.Where(e => e.Name.Length == 5)
                      .OrderByDescending(e => e.Name)
                      .Select(e => e);

var query2 = from developer in developers
             where developer.Name.Length == 5
             orderby developer.Name descending
             select developer;
```