More Effective LINQ

DISCOVERING THE POWER OF LINQ

Mark Heath

SOFTWARE DEVELOPER

@mark_heath www.markheath.net

Why LINQ?

It’s been around a while …

C# 3 Nov 2007

… but there’s more to LINQ than you think!

Language Features

Lambda Expressions

Extension Methods

Anonymous Types

Query Expression Syntax

Generics

yield and var Keywords

```
// old syntax
customers.Where(delegate(Customer c)
                { return c.Email != null; })



customers.Where (c => c.Email != null)
```

Lambda Expressions

Easily pass anonymous functions to methods

```
static class StringExtensions
{
    public static string Shout(this string s)
    {
        return s.ToUpper() + "!!!!";
    }
}
```

Extension Methods

Extend any type with additional methods

LINQ provides extension methods on IEnumerable<T>

Connect these extension methods together into “pipelines”

Misconception: “LINQ pipelines are for show-offs”

LINQ should make your code **more** readable, not less

```
// old repetitive code
Dictionary<string,Customer> dict = new
                            Dictionary<string,Customer>();



// using var keyword
var dict = new Dictionary<string,Customer>();
```

The var Keyword

Let the compiler infer the type for you

```
var x = new { Author = "Mark Seemann", Title = "Dependency
Injection in .NET" };
var y = new { Author = "Martin Fowler", Title = "Patterns
of Enterprise Architecture" };
var z = new { Author = "Robert Martin", Title = "Clean
Code", Pages = 245 }; // NOT the same type as x & y

var books = new[] { x, y };
```

Anonymous Types

Create new types without explicitly declaring a class

```
var author = "Adam Nathan";
var title = "WPF 4";

var book = new { Author = author, Title = title };

// inferred property names:
var book = new { author, title };
```

Anonymous Types

Can infer property names

Great for passing state through LINQ pipelines

Often preferable to tuples

```
var query = from c in customers
            group c by c.Country into countryGroup
            orderby countryGroup.Key
            select countryGroup;
```

Query Expression Syntax

Similar to SQL

Many new keywords

Misconception: “LINQ is just for database queries”

There are several LINQ “providers”.
e.g. LINQ to Entities
LINQ to Objects

```
var query = from c in customers
            group c by c.Country into countryGroup
            orderby countryGroup.Key
            select countryGroup;
```

Query Expression Syntax

Can be used with any LINQ provider, including LINQ to objects

Sometimes easier to read than chained extension methods

```
public static IEnumerable<T> DoubleUp<T>(this IEnumerable<T> source)
{
    foreach (var s in source)
    {
        yield return s;
        yield return s;
    }
}
```

Generics and the yield Keyword

Create classes and methods that can work with any type

The LINQ extension methods are generic

You can create your own generic methods

Language Features

Lambda Expressions

Extension Methods

Anonymous Types

Query Expression Syntax

Generics

yield and var Keywords

Expression Trees

Collections Are Everywhere!

In memory objects

Database queries

Algorithmically generated data

LINQ applies many powerful functional programming concepts in C#

Learning LINQ will increase your understanding of functional programming

LINQ and Functional Programming

Summary

LINQ is awesome

Lots of powerful C# language features

Applicable to almost all programs

What to Expect in the Rest of This Course

- 2 Thinking in Patterns
- 3 Unleashing the Power of Pipelines
- 4 Writing Clean and Readable Code
- 5 Extending LINQ
- 6 Avoiding Unnecessary Work
- 7 Optimizing Performance
- 8 Test and Debug Effectively
- 9 Embracing a Functional Style

Put it into practice with some “LINQ Challenges”