LINQ の内側

青柳 臣一
株式会社ディーバ

自己紹介

- 大阪在住（自宅…豊中、勤務先…淀屋橋）
- ブログ　http://shinichiaoyagi.blog25.fc2.com/
- Microsoft MVP (Visual Developer – Visual C#)
- Visual Studio User Group – C# フォーラムリーダー
  http://vsug.jp/
  VSUG Day 2007 Winter
  東京 12月8日（土） / 大阪 1月19日（土）
- 愛車　ANCHOR RFX8

LINQ to Object サンプル

```
var numbers = new int[] { 1, 2, 3, 4, 5 };

var q = from n in numbers
        where n % 2 == 0
        select n;

q.ToList().ForEach(Console.WriteLine);
```

LINQ to SQL サンプル

```
var db = new NorthWindDataContext();
var q = from c in db.Customers
        where c.City == "London"
        select c.CustomerID;

q.ToList().ForEach(Console.WriteLine);
```

コンパイラの気持ちになってみる LINQ to Object 編　その 1

- LINQ クエリ式はシンタックスシュガー

```
var q = numbers
      .Where(n => n % 2 == 0)
      .Select(n => n);
```

- numbers の型は？
  - int[]　…　配列は IEnumerable を実装している

コンパイラの気持ちになってみる LINQ to Object 編　その 2

- どの Where ?
  - System.Linq.Enumerable

```
public static IEnumerable<TSource> Where<TSource>(
    this IEnumerable<TSource> source, Func<TSource, bool> predicate);

public static IEnumerable<TSource> Where<TSource>(
    this IEnumerable<TSource> source, Func<TSource, int, bool> predicate);
```

  - System.Linq.Queryable

```
public static IQueryable<TSource> Where<TSource>(
    this IQueryable<TSource> source,
    Expression<Func<TSource, bool>> predicate);

public static IQueryable<TSource> Where<TSource>(
    this IQueryable<TSource> source,
    Expression<Func<TSource, int, bool>> predicate);
```

コンパイラの気持ちになってみる LINQ to Object 編　その 3

- どの Where ?　－　答え
  - int[] は IEnumerable なので当然 IEnumerable の方

- Where や Select の実装内容は ?

```
static class MyExtensions
{
	public static IEnumerable<TSource> MyWhere<TSource>(this IEnumerable<TSource> source,
	Func<TSource, bool> predicate)
	{
		foreach (var item in source)
		{
			if (predicate(item))
			{
				yield return item;
			}
		}
	}

	public static IEnumerable<TResult> MySelect<TSource, TResult>(this IEnumerable<TSource>
	source, Func<TSource, TResult> selector)
	{
		foreach (var item in source)
		{
			yield return selector(item);
		}
	}
}
```

Func<TSource, TResult> って何？

- System ネームスペースに定義されたジェネリックなデリゲート
- 引数の個数違いに 5種類の Func が定義済み

```
public delegate TResult Func<TResult>()
public delegate TResult Func<T, TResult>(T arg)
public delegate TResult Func<T1, T2, TResult>(T1 arg1, T2 arg2)
public delegate TResult Func<T1, T2, T3, TResult>(
    T1 arg1, T2 arg2, T3 arg3)
public delegate TResult Func<T1, T2, T3, T4, TResult>(
    T1 arg1, T2 arg2, T3 arg3, T4 arg4)
```

- すべてのデリゲートは Func である
  - （ただし引数無し～4つまでに限る）

匿名関数 (Anonymous Function)

- C# Language Specification 3.0 の用語
- 6.5 「Anonymous function conversions」
  - anonymous-method 式か lambda 式が匿名関数に分類される
  - 匿名関数は型は持たず、デリゲート型か Expression Tree 型に暗黙に変換することができる

```
anonymous-method 式
   deledate(int i) { return i * 2; }

lambda 式
   i => i * 2
```

Expression Tree 型　その 1

- C# Language Specification 3.0 の用語
- 型とは言っても、、、
  - System.Linq.Expressions.Expression<TDeledate> クラスのこと
- 4.6 「Expression tree types」
  - 匿名関数を実行コードの代わりにデータ構造として表現したもの

```
Func<int,int> del = x => x + 1;                    // Code
Expression<Func<int,int>> exp = x => x + 1;        // Data
```

Expression Tree 型　その 2

- Expression Tree にできるのは lambda 式のみ（だと思う）
- ExpressionTreeVisualizer
  - デバッグ時に Expression Tree を表示するツール

コンパイラの気持ちになってみる LINQ to SQL 編　その 1

- LINQ クエリ式はシンタックスシュガー

```
var db = new NorthWindDataContext();
var q = db.Customers
      .Where(c => c.City == "London")
      .Select(c => c.CustomerID);
```

- Customers の型は？
  - System.Data.Linq.Table<Customer>
    … IQueryable<TEntity> を実装している

コンパイラの気持ちになってみる LINQ to Object 編　その 2

- どの Where ?　－　答え
  - 当然 IQueyable の方
- Where や Select の実装内容は ?
  - Func<TSource, bool> ではなく Expression<Func<TSource, bool>> なのがミソ

```
public static IQueryable<TSource> Where<TSource>(
   this IQueryable<TSource> source,
   Expression<Func<TSource, bool>> predicate);
```

コンパイラの気持ちになってみる LINQ to SQL 編　その 3

- Where や Select の実装内容は？　（続き）
  - Where や Select は source が持っているプロバイダの CreateQuery() メソッドを呼び出す
    - このときに predicate （これが Expression Tree） を渡している
  - コンパイル時に行われるのはここまで
- 実行時
  - GetEnumerator() メソッドが呼び出された時、プロバイダの Execute() メソッドが呼ばれる

動的に Where 式を作る

```
var db = new NorthWindDataContext();
var q = from c in db.Customers
        select c;

if (mode == 0)
{
    q = q.Where(c => c.City == "London");
}
else
{
    q = q.Where(c => c.Country == "USA");
}
```

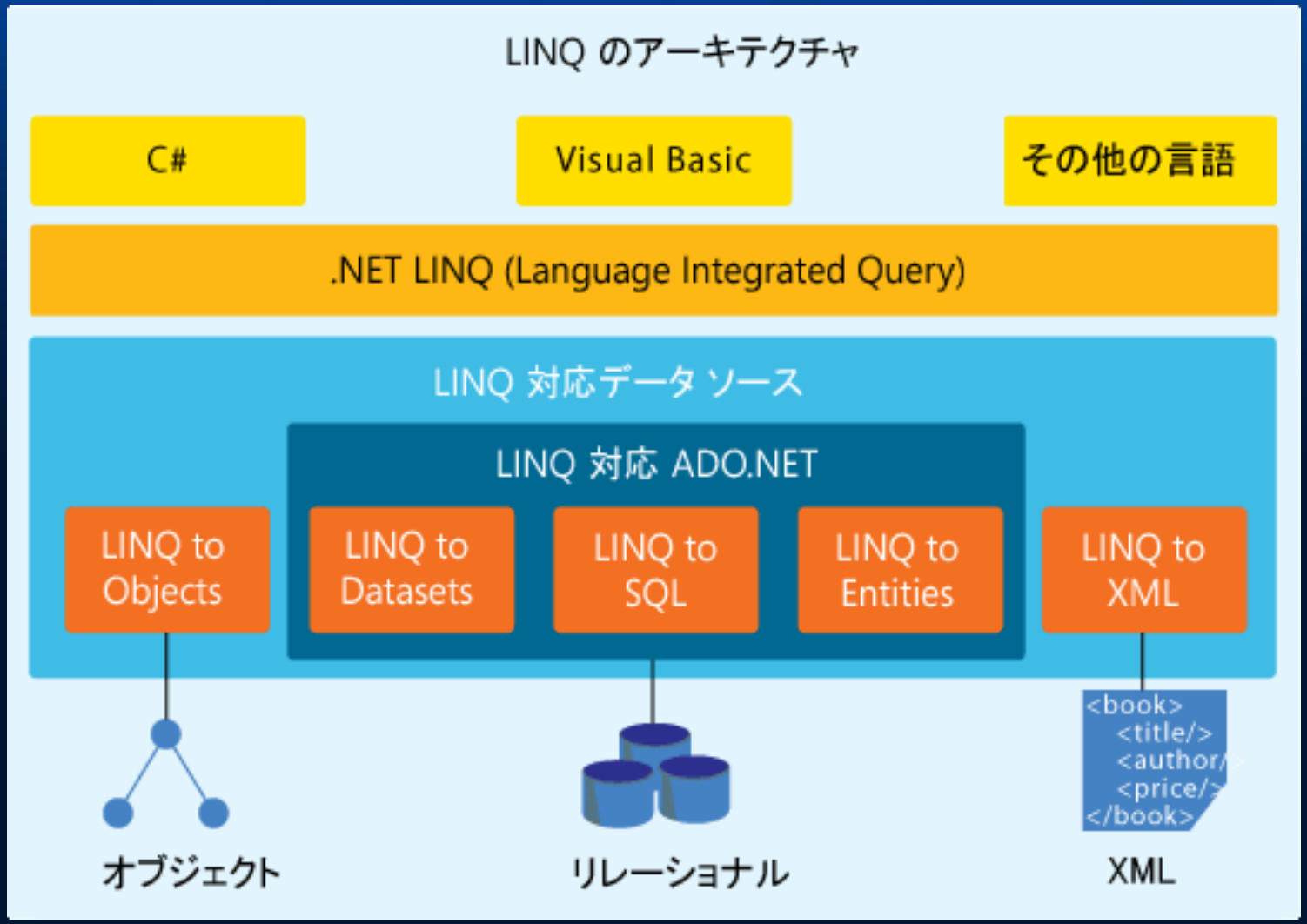
もっと動的に Where 式を作る

```
string city = “London”;
string country = “UK”;

ParameterExpression param = Expression.Parameter(typeof(Customer), "");
Expression condition1 =  Expression.Equal(
        Expression.Property(param, typeof(Customer).GetProperty("City")),
        Expression.Constant(city)
);
Expression condition2 = Expression.Equal(
        Expression.Property(param, typeof(Customer).GetProperty("Country")),
        Expression.Constant(country)
);
Expression condition = Expression.And(condition1, condition2);
var predicate =  Expression.Lambda<Func<Customer, bool>>(
        condition,
        new ParameterExpression[] { param }
);

var db = new NorthWindDataContext();
var filteredCustomers = db.Customers.Where(predicate);
var q = from c in filteredCustomers
        orderby c.CustomerID
        select c.CustomerID;
```

LINQ の概念図


LINQ のアーキテクチャ
C#
Visual Basic
その他の言語
.NET LINQ (Language Integrated Query)
LINQ 対応データ ソース
LINQ 対応 ADO.NET
LINQ to Objects
LINQ to Datasets
LINQ to SQL
LINQ to Entities
LINQ to XML
<book>
<title/>
<author/>
<price/>
</book>
オブジェクト
リレーショナル
XML

LINQ to ほげほげ

- http://oakleafblog.blogspot.com/2007/03/third-party-linq-providers.html より
  - LINQ to WebQueries
  - LINQ to Amazon
  - LINQ to RDF Files
  - LINQ to MySQL
  - LINQ to NHibernate
  - LINQ to LDAP
  - LINQ to Flickr
  - LINQ to Google Desktop
  - LINQ to SharePoint
  - LINQ to Streams (SLinq, Streaming LINQ)
  - LINQ to Expressions

まとめ

- LINQ はデリゲート、もしくは、Expression Tree を作って Extension Method に渡すだけのものである

[参考] ExpressionTreeVisualizer

- サンプルに含まれる
  - C# 版　http://download.microsoft.com/download/8/0/e/80e3f901-9bcc-4b8b-ba66-c49e24559fbc/vcs_samples_vs2008_beta2_02.exe
  - VB 版　http://download.microsoft.com/download/1/5/6/1569370a-a54d-4b0f-999f-ebad37a0aeea/vb%20samples%20beta2.msi
- 実行して解凍
- dll をコピー
  - ExpressionTreeVisualizer.dll と LinqToSqlQueryVisualizer.dll を
    My Documents¥Visual Studio 2008¥Visualizers にコピーする
  - デフォルトでは
    My Documents¥Visual Studio Samples¥
    LinqSamples¥ExpressionTreeVisualizer¥ExpressionTreeVisualizer
    My Documents¥Visual Studio Samples¥
    LinqSamples¥QueryVisualizer¥SqlServerQueryVisualizer

参考資料

C# Language Specification 3.0（英語）
http://msdn2.microsoft.com/en-us/vcsharp/Aa336809.aspx

MSDN Library - Language-Integrated Query (LINQ)（英語）
http://msdn2.microsoft.com/en-us/library/bb397926(VS.90).aspx

The Wayward WebLog： LINQ: Building an IQueryable Provider - Part I（英語）
http://blogs.msdn.com/mattwar/archive/2007/07/30/linq-building-an-iqueryable-provider-part-i.aspx
（現在 Part VII まで公開）

Kevin's VB Adventures : How to implement IQueryable (Part 1), (Part 2)（英語）
http://blogs.msdn.com/kevin_halverson/archive/2007/07/10/how-to-implement-iqueryable.aspx
http://blogs.msdn.com/kevin_halverson/archive/2007/07/11/how-to-implement-iqueryable-part-2.aspx

LukeH's WebLog : Taking LINQ to Objects to Extremes: A fully LINQified RayTracer
http://blogs.msdn.com/lukeh/archive/2007/10/01/taking-linq-to-objects-to-extremes-a-fully-linqified-raytracer.aspx