Reactive Extensions入门(1)：LINQ和Rx简单介绍

yycoding.xyz/post/2012/4/17/introducting-linq-and-reactiveextensions

相信大家都用过Language Integrated Query (LINQ)，他是一种强大的工具能够从集合中提取数据。Reactive Extensions（Rx）是对LINQ的一种扩展，他的目标是对异步的集合进行操作，也就是说，集合中的元素是异步填充的，比如说从webservice或者云端获取数据然后对集合进行填充。

LINQ是从C#3.0开始引入的语言特性，而Rx起源于Microsoft DevLabs小组的研究，他扩展了LINQ的一些特性，目前Rx支持多种平台如JavaScript，Windows Phone。随着数据处理变得复杂，LINQ使得我们的处理逻辑变得简单清晰，同样地，随着越来越多的数据通过从云端异步获取，Rx使得这种异步数据处理操作变得简单和容易维护。

本文简要介绍LINQ和Rx的重要性，并用代码展示了LINQ和Rx的一些基本操作。

1. 什么是LINQ和Rx

1.1 什么是LINQ

LINQ是.NET的一个扩展，专门用来对数据集合，关系数据，XML文件等对象进行查询和提取数据的技术，他提供了统一的类似SQL的语法来对数据进行查询，而不用关心数据源的不同。

LINQ使得我们的代码风格更加声明化和函数化，在我之前的一篇文章 查询表达式和循环控制 读者可以明显的感受到这一点。声明式编程风格相信大家都使用过，比如说C#中的特性(AttributeClass)，WPF中的XAML以及F#语言，大家可以从中体会到，声明式编程风格强调的是实现的逻辑和需求，而不是具体的执行步骤。LINQ和Rx是C#中声明式和函数式编码风格的一部分，他强调如何能够完成想要的功能，而不是专注于怎么样去实现。函数式编程(Function Programming)和传统的命令式编程(Imperative Programing)风格相比，代码更加简洁明了，容易维护。

命令是编程中，程序通常有几个步骤组成，在某一步骤中通常有循环，循环体内是具体的逻辑。在声明式编程中，使用更高一层的抽象来表达意图。举例来说，命令式编程式编程通常是这样的“打开所有员工集合，遍历集合中的元素，看该员工是否是经理，如果是，则将工资增加20%，然后将增加后的工资更新回集合中去”，而声明式编程风格大体会是“将经理的工资增加20%”，可见，命令式编程注重的是如何完成某一任务，生命式编程则注重的是要完成什么任务。

1.2 什么是Rx

Rx提供了一种新的组织和协调异步事件的方式，例如协调多个从云端返回的多个异步的数据流。Rx能够是的我们用一个简单的方式来处理这些数据流，极大的简化了代码的编写。例如，.NET中传统的Begin/End异步编程模式在处理单个异步操作时可以应付，但是如果同时多个异步调用时，线程控制就会使得代码变得比较复杂。使用Rx，Begin/End模式就变成了一条简单的方法，这使得代码更加清晰和容易理解。

Rx最显著的特性是使用可观察集合(Observable Collection)来达到集成异步(composing asynchronous)和基于事件(event-based)的编程的效果。Rx有一些几个特性。

- 组合(Composing): Reactive Extension的首要目标之一就是将多种异步操作组合起来是的代码更加简单。要做到这一点，数据流必须定义清楚，这样代码就很清晰集中，使得异步操作代码异步处理代码不会充斥整个应用程序。
- 异步(Asynchronous): 虽然Rx不仅仅能处理异步操作，但是使用Rx，大大简化了异步操作的实现，并且代码容易理解进而容易维护。
- 基于事件(Event-based): Rx简化了传统的异步编程方式，在后面的文章中我们会看到拖放(drag and drop)模式的实现
- 可观察集合(Observable collections): Obervable Collection是Rx的核心，它是一种集合，集合的元素在第一次访问的时候肯能还没有填充。它对与Rx的重要性类始于enumerable集合对LINQ的重要性。

2. 获取和安装Rx

如果使用.NET 3或者之后的版本，那么LINQ已经集成到framework中了，同样，如果安装了Windows Phone开发工具，Rx的WindowsPhone版本也会自动安装。要获得最新的Rx可以访问其官网(http://msdn.microsoft.com/en-us/data/gg577609)。

3. Rx与LINQ的异同

在处理静态集合数据方面，LINQ使用类似SQL的语法来操作和使用不同来源的数据。相反，Rx被设计出来用来处理将来才会填充好的集合，也就是说，集合类型定义好了，但是集合中的元素可能在未来的某一时刻才会被填充。

在使用LINQ之前要求集合中的所有元素都可以使用，也就是说在使用LINQ集合进行操作时，需要一个初始的静态的集合。但是，问题来了，如果集合中有些数据目前还在处理之中，或者说是调用远处的webservice结果还没有返回，或者是一些实时产生的数据会不断的往集合中添加，那么对于这类集合LINQ不知道该怎么处理，因为他们不是静态的，集合中的元素会在未来的某一个时刻继续添加，并且数据到达的事件不确定。Rx就是针对这种数据集合设置的，它是的操作这类数据集合就像LINQ操作静态数据集合那样简单。

就像大家开始接触LINQ那样会感觉不容易理解，初次接触Rx也会有类似的问题。一旦理解和熟悉了Rx的语法，就会感觉和LINQ一样亲切，一些复杂的异步操作使用Rx可以非常简单的实现。

LINQ和Rx在技术上有很多相似的地方。在LINQ对集合进行一系列操作如添加，移除，修改，提取后，会得到一个新的集合，新集合只是原始集合的一个修改版本。Rx也是一样，咋一看，集合和数据流看起来非常不同，但是他们在很多关键的地方有联系，这就是我们将数据流称之为未来的集合的原因。集合和数据流都是多数据按某种顺序进行排列。LINQ和Rx可以这些序列进行一系列操作然后得到一个新的序列。

使用LINQ和Rx能够带给我们很多好处，他们有一下好处：

- 他们是.NET框架中的一级成员(first-class member)。Visual Studio中完全支持对LINQ和Rx的智能感知和语法高亮。
- 他们能够以一种统一的方式来处理各种数据源产生的数据，这些数据源包括数据库或者XML文件。
- 他们具有很强的扩展性。你可以编写自己的类库来扩展这些框架。
- 组合性。LINQ和Rx都可以将一系列复杂的操作和变化组合到一小片代码段中。使得代码更加简洁。
- 声明式风格。LINQ和Rx为编程带来了一些函数式编程的风格。
- 简化实现。很多时候一条或者几条LINQ，Rx语句就能轻易实现很多需要复杂和晦涩的语句才能完成的功能。

4. Rx和LINQ的简单例子

讲了这么多，下面来看看一个简单的例子来说明LINQ和Rx的用法。大家可能对LINQ比较熟悉，那么先看看LINQ。

```
static void Main(string[] args)
{
    List<Int32> its = Enumerable.Range(1, 15).ToList();
    its.Where(i => i % 2 == 0).ToList().ForEach(i => Console.Write("{0} ", i));
    Console.ReadKey();
}
```

上述代码输出结果：2 4 6 8 10 12 14

上面例子很简单，首先使用Enumerable对象的Range方法初始化一个list集合，然后使用LINQ的方法型表达式过滤所有偶数，LINQ语句代码执行返回一个IEnumerable集合，然后使用IEnumerable集合的扩展方法Tolist将其转换为List集合，在调用ForEach方法传入一个匿名方法将所有集合中的所有元素打印出来，整个代码紧凑易读，逻辑清晰。

下面来如何使用Rx实现上述功能。要使用Rx必须添加System.Reactive.dll，默认安装路径下，我本机的地址为C:\Program Files\Microsoft Reactive Extensions SDK\v1.0.10621\Binaries\.NETFramework\v4.0\ System.Reactive.dll。

```
static void Main(string[] args)
{
    IObservable<Int32> input = Observable.Range(1, 15);
    input.Where(i => i % 2 == 0).Subscribe(x => Console.Write("{0} ", x));
    Console.ReadKey();
}
```

代码输出结果和LINQ版本的完全相同，但是程序的内在逻辑完全不同。上述代码中，首先使用Observable对象产生一个IObservable对象集合，然后使用Where子句过滤其中的元素，得到一个新的IObservable对象，然后对对象中的每一个元素注册一个方法，上述代码中使用的是匿名方法将每个元素打印出来。当集合中有元素时，因为元素注册了该方法，所以就会调用该方法把元素打印出来。

LINQ和Rx的能处理的集合类型不同导致其产生内在逻辑不同，下面先简单了解一下Enumerable集合和Observable集合。

5. Enumerable集合和Observable集合

LINQ和Rx都是用来对集合进行操作。LINQ操作的集合实现了IEnumerable借口，能够使用foreach语句遍历集合。而Rx操作的集合实现了IEnumerable，IQueryable集合，这样的集合称之为Observable集合。大家对Enumerable集合可能很熟悉，他是foreach语句的基础，我的另一篇文章对这个有详细介绍，这里就不多说了，下面主要来看看Observable集合。

Rx对Observable集合进行操作，这个集合的命名是从观察者设计模式得来的，观察者模式的基础是委托和事件，所以要了解这一模式需要理解委托和事件，在这里推荐张子阳的文章C#中的委托和事件。Enumebrable集合中所有的元素在集合中都已经填充好了，是静态的，用户可以使用“拉”的方式从集合中遍历元素进行处理。而Observable集合则不同，在创建该集合时，集合中的元素可能会在以后的某个时间才能添加进去。由于集合注册了事件，一旦集合中的元素到达，就会触发这一事件，将信息“推”到注册者哪里去。

现在在Windows Phone中演示如何使用这两个集合，在WindowPhone界面上添加两个ListBox控件，一个使用Enumerable集合填充一个使用Observable集合填充。具体步骤如下：

1. 新建一个Windows Phone项目，在界面上放两个ListBox，分别命名为lblEnumerable和lblObservable。

2. 添加System.Observable.dll和Microsoft.Phone.Reactive.dll，并在代码中添加对Microsoft.Phone.Reactive的引用。

3. 初始化一个数组变量，然后分别利用两种方式分别对这两个ListBox进行填充。

完整代码如下：

```
using Microsoft.Phone.Controls;
using Microsoft.Phone.Reactive;

namespace IntroductionRxPhoneApp
{
    public partial class MainPage : PhoneApplicationPage
    {
        readonly List<String> fruits = new List<string>{
        "Apple","Orange","Banana","Pears","melon"};
        // 构造函数
        public MainPage()
        {
            InitializeComponent();
            PopulateCollection();
        }

        private void PopulateCollection()
        {
            //使用Enumerable集合进行填充
            fruits.ForEach(fruit => lblEnumerable.Items.Add(fruit));

            //使用Observable集合进行填充
            IObservable<String> observable = fruits.ToObservable();
            observable.Subscribe(fruit => lblObservable.Items.Add(fruit));
        }
    }
}
```

运行程序可以看到两个下拉框中的内容是一样的。

6. 结语

本文简要介绍了什么是Rx以及Rx如何对LINQ进行的扩展。并以实际代码简单演示了Rx的用法，希望本文对您了解Rx有所帮助，因为Rx和LINQ有在技术方面有很多共同的地方，所以下一篇文章将会介绍LINQ的一些核心的用法，敬请期待！