Reactive Extensions

We'll take a look at one of the projects from MS DevLabs: Reactive Extensions, Rx is a library to compose asynchronous and event-based programs using observable collections and LINQ-style query operators. We'll walk through the technology and how it affects our way of looking at this kind of problem and discuss how it can be used.

Pär Nordström, Valtech

Reactive Extensions

Data Development Center

We'll take a look at one of the projects from ~~MS DevLabs~~: Reactive Extensions, Rx is a library to compose asynchronous and event-based programs using observable collections and LINQ-style query operators. We'll walk through the technology and how it affects our way of looking at this kind of problem and discuss how it can be used.

Pär Nordström, Valtech

```
// Example 1.a
var list = new List<int>() { 1, 2, 3, 4, 5, 6 };

foreach (var item in list)
{
    Console.WriteLine(item);
}
Console.WriteLine("---");
```

```
//Example 1.b
var list = new List<int>() { 1, 2, 3, 4, 5, 6 };
var enumerator = list.GetEnumerator();
while (enumerator.MoveNext())
{
    Console.WriteLine(enumerator.Current);
}
Console.WriteLine("---");
```

```
// Example 2
var sequence = new FunkyEnumerable(i => i*10);
foreach (var item in sequence)
{
    Console.WriteLine(item);
    Console.WriteLine("Waiting...");
}
```

```
// Example 3
var sequence =
    new FunkyEnumerable(i => i * i)
    .ToObservable(Scheduler.TaskPool)
    .Subscribe(str => Console.WriteLine(str));

Console.WriteLine("Working with other stuff..");
while (true)
{
    Console.Write(".");
    Thread.Sleep(10);
}
```

```
// Example 4
var evenNumbers = from t in new FunkyEnumerable(i => 100)
                      .ToObservable(Scheduler.TaskPool)
                  where int.Parse(t)%2 == 0
                  select t;

var oddNumbers = from t in new FunkyEnumerable(i => 200)
                     .ToObservable(Scheduler.TaskPool)
                 where int.Parse(t) % 2 != 0
                 select t;

using (evenNumbers.Subscribe(str => Console.WriteLine(str + " - even")))
using (oddNumbers.Subscribe(str => Console.WriteLine(str + " - odd")))
{
    Console.WriteLine("Press Enter to stop...");
    Console.ReadLine();
}
```

```
var moves = from evt in Observable.FromEvent<MouseEventArgs>(frm, "MouseMove")
            where txt.Text.ToUpper() == "ON"
            select evt.EventArgs.Location.ToString();
var input = from evt in Observable.FromEvent<EventArgs>(txt, "TextChanged")
            select txt.Text;
var quit = from evt in Observable.FromEvent<EventArgs>(txt, "TextChanged")
           where txt.Text.ToUpper() == "QUIT"
           select txt.Text;
```

```
var moves = from evt in Observable.FromEvent<MouseEventArgs>(frm, "MouseMove")
                .Sample(TimeSpan.FromSeconds(1))
                .DistinctUntilChanged(e => e.EventArgs.Location)
            where txt.Text.ToUpper() == "ON"
            select evt.EventArgs.Location.ToString();
var input = from evt in Observable.FromEvent<EventArgs>(txt, "TextChanged")
                .Delay(TimeSpan.FromSeconds(3))
                .DistinctUntilChanged(e => txt.Text)
            select txt.Text;
var quit = from evt in Observable.FromEvent<EventArgs>(txt, "TextChanged")
               .Throttle(TimeSpan.FromSeconds(2))
           where txt.Text.ToUpper() == "QUIT"
           select txt.Text;
```

```
// Exempel 7
var loremSvc = new LoremService.LoremClient();

var svc1 = from text in Observable.FromAsyncPattern<string>(loremSvc.BeginGetLorem, loremSvc.EndGetLorem)()
           select "Svc1: " + text;
var svc2 = from text in Observable.FromAsyncPattern<string>(loremSvc.BeginGetLorem, loremSvc.EndGetLorem)()
           select "Svc2: " + text;
var svc3 = from text in Observable.FromAsyncPattern<string>(loremSvc.BeginGetLorem, loremSvc.EndGetLorem)()
           select "Svc3: " + text;
```

Rx for Windows Phone 7, Xbox, and Zune

Windows Phone 7 ships with a version of the Reactive Extensions baked into the ROM of the device. For more information, see Reactive Extensions for .NET Overview for Windows Phone. This version of the Reactive Extensions can be found in Windows Phone API's Microsoft.Phone.Reactive Namespace.

After shipping the initial version of the Reactive Extensions with the Windows Phone 7 release, our team continues to improve and extend the library. This section describes how to get started with the latest version of Reactive Extensions for Windows Phone 7, which is defined in the System.Linq namespace. As a result, the new APIs don't clash with the library built in to the phone (nor do they replace the version in the ROM).

We also provide implementations of the Reactive Extensions for XNA-based platforms, which can be used to write reactive applications that run on Xbox 360 and Zune.

Rx for JavaScript

We're proud to announce the availability of Reactive Extensions for JavaScript (RxJS), bringing the power of reactive programming to JavaScript. It allows you to use the Rx combinators in JavaScript, and it does this in a download size of less than 7KB (using GZip compression). RxJS provides easy-to-use conversions from existing DOM, XmlHttpRequest (AJAX), and jQuery events to Rx push-collections, allowing users to seamlessly plug Rx into their existing JavaScript-based web sites.

Install Rx through NuGet

Rx assemblies can also be included to a Visual Studio 2010 project using the NuGet package management system. For more information on how to install it, visit the NuGet project page on CodePlex. To find the Rx assemblies in the NuGet "Add Library Package Reference..." dialog, use "Rx" as the search term.

@parnordstrom
parnordstrom.com
par.nordstrom@valtech.se

Links

http://msdn.microsoft.com/en-us/data/gg577609