ReactiveX

Reactive Extensions

1. History
2. Definition of Rx
3. Data Pull Model vs Data Push Model
4. Observer Pattern
5. Cold/Hot Observables
6. Samples

History

blogs.msdn.com/b/rxteam/default.aspx?PostSortBy=MostRecent&PageIndex=2

How to cont... | 64 bit - Brightness C... | 11.10 - Screen Brigh... | Ruby | Dev interview | Dev stuff | Rhodes - Motorola ... | Mobile App Develo... | 8 Things To D

Announcing Reactive Extensions (Rx) for .NET & Silverlight

2

over 6 years ago by Rx team

Finally, the long wait is over. Rx is available on DevLabs!

What is Rx?
Rx is a .NET Library that allows programmers to write succinct declarative code to orchestrate and coordinate asynchronous and event-based programs based on familiar .NET idioms and patterns.

At the heart of the library are two new interfaces IObservable<T> and IObserver<T>. These interfaces are part of .NET 4 and Silverlight 4 by default. For .NET 3.5 and Silverlight 3.5, they ship with the Rx library.
The Rx library provides API create instances of these interfaces from scratch, convert existing asynchronous patterns such as events to these interfaces, query, transform and merge observables. All together creating a new powerful way of writing asynchronous applications.

In the recent past there have been several videos on Channel9 talking about Rx in Depth. We can strongly recommend watching them:

- Kim Hamilton and Wes Dyer: Inside .NET Rx and IObservable/IObserver in the BCL (VS 2010)
- Expert to Expert: Brian Beckman and Erik Meijer - Inside the .NET Reactive Framework (Rx)
- E2E: Erik Meijer and Wes Dyer - Reactive Framework (Rx) Under the Hood 1 of 2
- E2E: Erik Meijer and Wes Dyer - Reactive Framework (Rx) Under the Hood 2 of 2

How do I get started?

- First, visit the DevLabs Rx Project Page, watch the videos, read the documentation and

History

reactivex.io

Apps | 12.04 - How to cont... | 64 bit - Brightness C... | 11.10 - Screen Brigh... | Ruby | Dev interview | Dev stuff | Rhodes - Motorola ... | Mobile App Develo... | 8 Things To Do Afte... | Other bookmarks

ReactiveX | Introduction | Docs | Languages | Resources | Community

We use ReactiveX

Applied Duality | Microsoft | NETFLIX | GitHub

SOUNDCLOUD | Trello | treehouse | SeatGeek

odVoo | Couchbase | Futurice | welbe

DOCUMENTATION

- Observable
- Operators
- Subject
- Scheduler

LANGUAGES

- RxJava
- RxJS
- Rx.NET
- RxScala
- RxClojure
- RxSwift
- Others

RESOURCES

- Tutorials

COMMUNITY

- GitHub
- Twitter
- Others

An example how to polling the srv

```
/// <summary>
/// Example6. Polling async service.
/// </summary>
public void Example6()
{
    Console.WriteLine("Example6:");
    Console.WriteLine("Waiting for data3...");

    var getData3A
        = Observable
            .FromAsyncPattern<int>(
                asyncClient.BeginGetData3,
                asyncClient.EndGetData3)();
    Observable
        .Interval(TimeSpan.FromSeconds(1))
        .Subscribe(value => getData3A
            .Subscribe(data3 => Console.WriteLine("data3 has arrived: {0}", data3)));
```

Please do not DDOS the server with 1 sec polling time in real life. :)

Definition

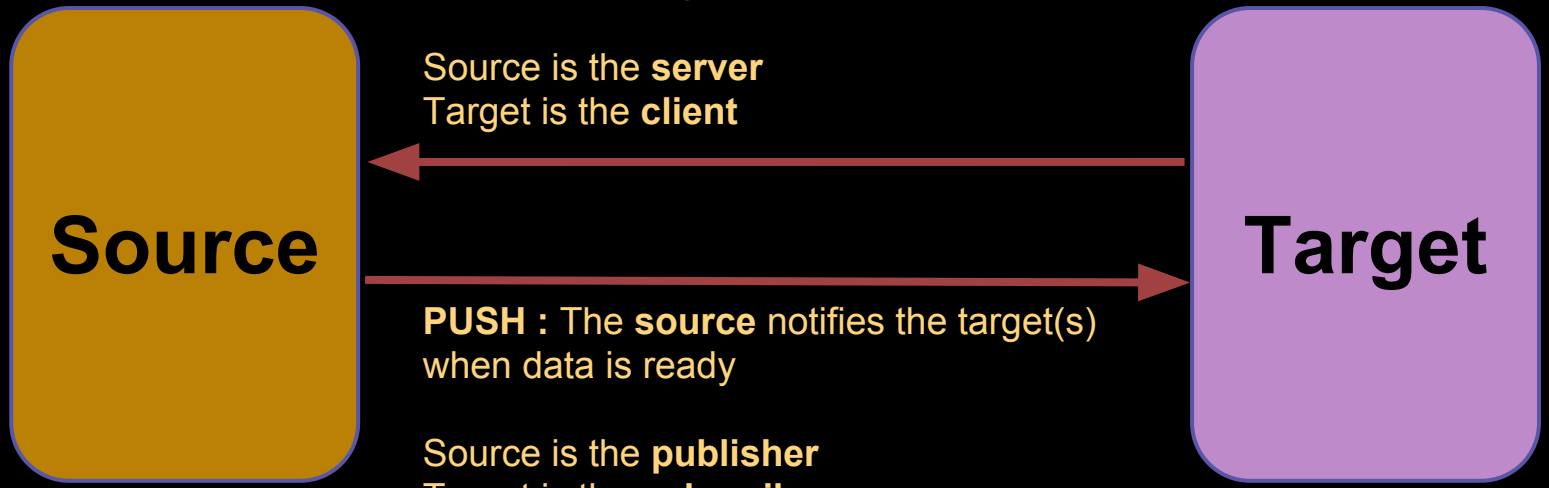
“The Reactive Extensions (Rx) is a library for composing ***asynchronous*** and ***event-based*** programs using ***observable sequences*** and ***LINQ-style query operators***.”

“RxJava is a Java VM implementation of Reactive Extensions: a library for composing ***asynchronous*** and ***event-based*** programs by using ***observable sequences***.”

“The Reactive Extensions for JavaScript (RxJS) is a set of libraries for composing ***asynchronous*** and ***event-based*** programs using ***observable sequences*** and ***fluent query operators***...”

- asynchronous
- event-based
- observable

Data pull vs Data push



PULL : The **target** must ask for data

Source is the **server**
Target is the **client**

PUSH : The **source** notifies the target(s) when data is ready

Source is the **publisher**
Target is the **subscriber**

Pull vs Push pros and cons

Pull

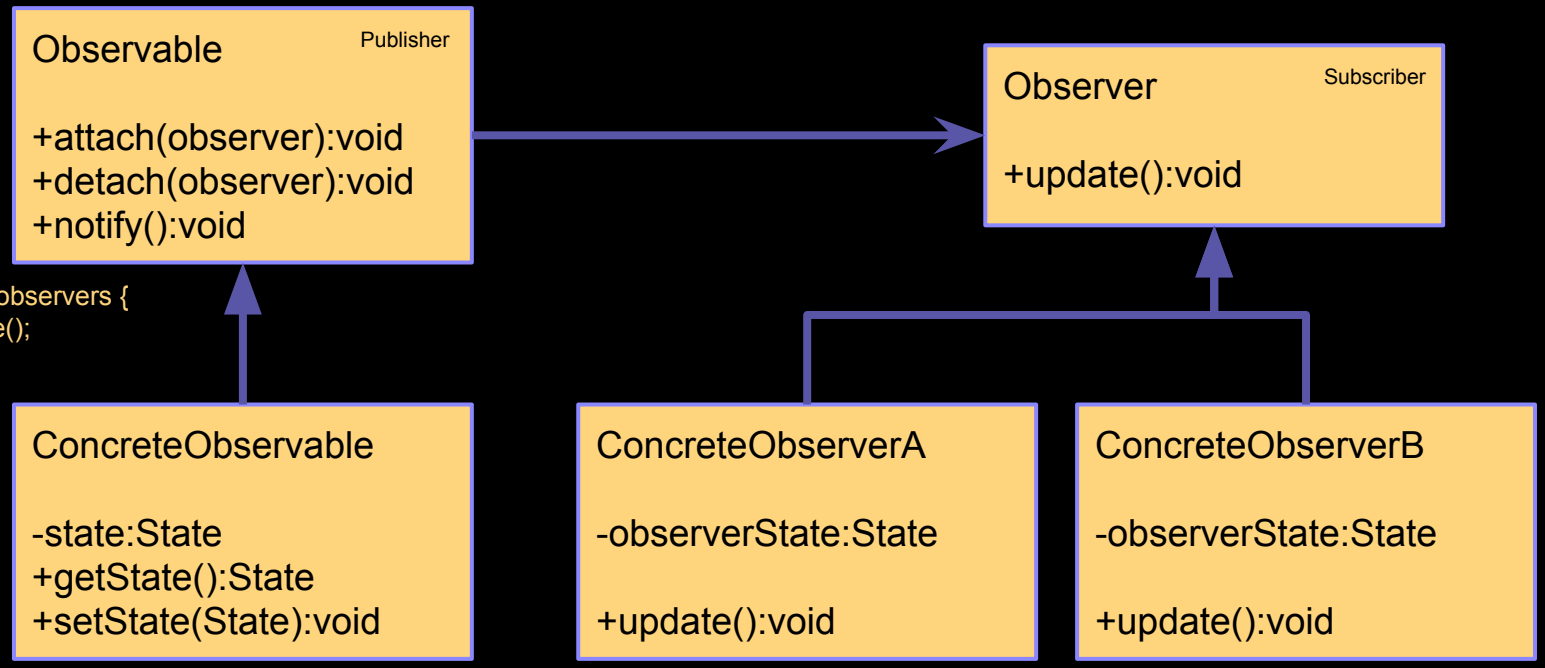
- Pros
  - Easy to get data at target side
  - Easy to understand the mechanism
- Cons
  - Custom polling methods needed

Push

- Pros
  - No need to poll the source for data
  - Easy to publish data to multiple targets
- Cons
  - Complicated to handle
  - Relatively higher threshold of understanding

Observer pattern

http://www.oodesign.com/observer-pattern.html

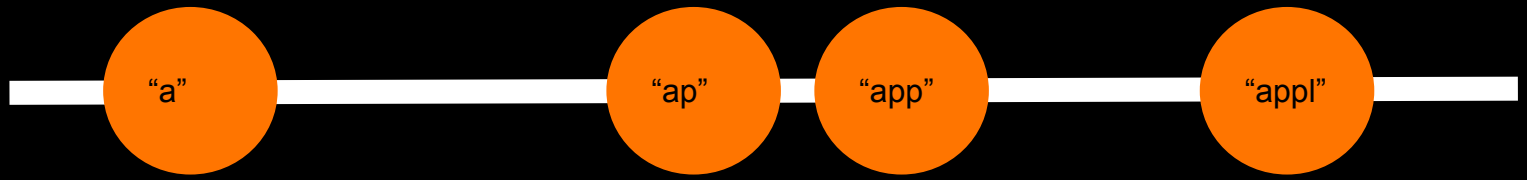



```
notify:

for all o in observers {
    o.update();
}
```

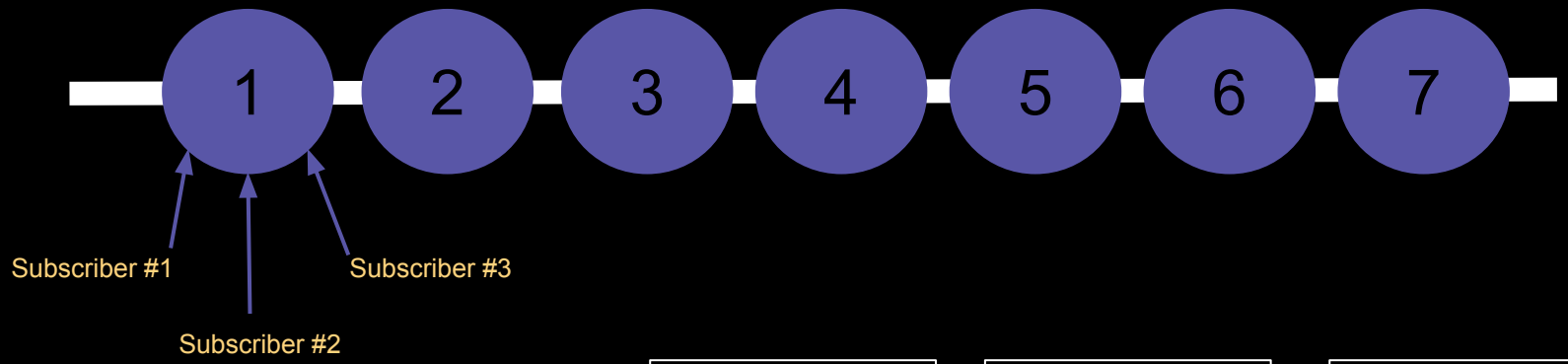
```
void update() {
    observerState = observable.getState();
}
```

Cold/Hot Observables



- Data arrive in various times
- Data type can be anything (either primitive or complex)

Cold Observables



- Any static list
- Distribution list
- Count down

Subscriber #1 result:
1
2
3
4
5
6
7

Subscriber #2 result:
1
2
3
4
5
6
7

Subscriber #3 result:
1
2
3
4
5
6
7

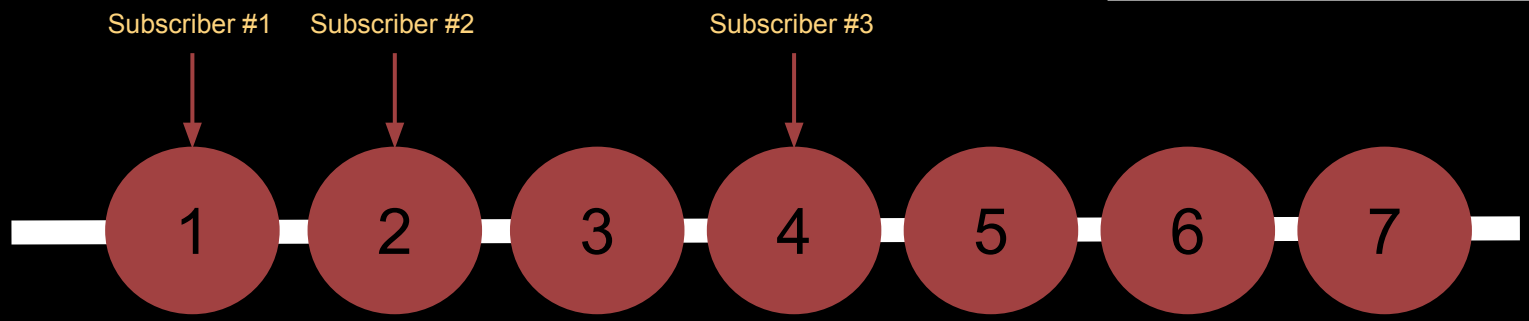
Hot Observables

Subscriber #1 result:
1
2
3
4
5
6
7

Subscriber #2 result:
2
3
4
5
6
7

Subscriber #3 result:
4
5
6
7

- MouseMove
- Data from sensors
  - temperature
  - speed
  - pressure



“We are developers, we write code” - Erik Meijer

Samples

Thank you

http://rxwiki.wikidot.com/101samples

https://github.com/silverforge/RxAsyncConsole

https://github.com/silverforge/TwitterClient