

Observable vs. Task

This article explains when to use observables and when to use tasks.



Introduction

In [Implementing GeoCoordinateWatcher as a Reactive Service](#) we saw how to convert an event-based asynchronous API into an API based on *observables*, while in [Bringing async/await to the Contacts service](#) we saw how to convert to an API based on *tasks*.

This article answers the question "in what circumstances should each of the techniques be used".

When to use observables

Reactive Programming allows you to compose asynchronous and event-based programs using observable sequences and LINQ-style query operators. It is very suited to handling, combining and filtering *streams* of events.

The need to combine several event streams or filtering is a good clue for the use of observables.

When to use tasks

As the name suggests, a task is something that has a beginning and an end. So, all asynchronous APIs based on callbacks or events are candidates to be converted into a task-based asynchronous API.

Applicability matrix

Another way to look at this is:

- Tasks are for a single future value
- Observables are for multiple future values

The following table shows the correspondence between asynchronous APIs and their synchronous counterparts:

	Synchronous	Asynchronous
Single Value	- <pre>// synchronous call var x = F(42); // synchronous call var y = G(x);</pre>	Task <pre>// asynchronous call var x = FAsync(42); // asynchronous call var y = GAsync(x);</pre>
Multiple Values	Enumerable <pre>// Shopping database var res = from o in orders from p in p.Products where p.Price > 29.95m select p.Name; // Synchronous MoveNext foreach (var p in res) { Console.WriteLine(p);</pre>	Observable <pre>// Stock trade events var res = from t in trades from q in t.Symbols where q.Quote > 29.95m select q.Symbol; // Asynchronous MoveNext res.Subscribe(p => Console.WriteLine(p));</pre>

```
}
```

References

- [Reactive Extensions \(Rx\)](#) 
- [Asynchronous Programming with Async and Await \(C# and Visual Basic\)](#) 