



Clipcode Pattern Library

Overview

Every serious software architect interested in asynchronous messaging has a copy of the wonderful book “Enterprise Integration Patterns” by Gregor Hohpe and Bobby Woolf (ISBN: 0-321-20068-3). Its subtitle is “Designing, Building and Deploying Messaging Solutions” and it explains in detail 65 asynchronous messaging patterns.

The goal of the Clipcode Pattern Library for Rx is to examine how Rx can be used to implement a selection of those patterns. It provides a more real-world selection of problems to see how well Rx can be used.

Sample Pattern

Here is a sample pattern.

```
namespace EIP_Patterns_In_Rx
{
    // CorrelationIdentifier - See Hohpe&Woolf page 163
    // How does a requestor that has received a reply
    // know which request this is the reply for?

    class Envelope<T>
    {
        public UInt64 CorrelationIdentifier { get; private set; }
        public T Message { get; private set; }

        public Envelope(UInt64 identifier, T message)
        {
            if (identifier == 0)
                throw new ArgumentException("identifier");
            if (message == null)
                throw new ArgumentNullException("message");
            CorrelationIdentifier = identifier;
            Message = message;
        }
    }

    public static class CoorelationIdentiferPattern
    {
        public static void Run()
        {
            Subject<Envelope<UInt64>> subject = new Subject<Envelope<ulong>>();

            subject.Subscribe(
                (respEnv) => Console.WriteLine("The response for request with Id "
                    + respEnv.CorrelationIdentifier.ToString() + " is "
                    + respEnv.Message.ToString()),
                (ex) => Console.WriteLine("Exception - " + ex.ToString()),
                () => Console.WriteLine("OnCompleted detected"));

            subject.OnNext(new Envelope<ulong>(1, 124));
            subject.OnNext(new Envelope<ulong>(2, 322));
            subject.OnNext(new Envelope<ulong>(3, 432));
        }
    }
}
```

```
    Console.WriteLine("Press ENTER to continue");
    Console.ReadLine();
}

static Envelope<UInt64> DoubleEngine(Envelope<UInt64> reqEnv)
{
    Envelope<UInt64> respEnv = new Envelope<UInt64>(
        reqEnv.CorrelationIdentifier, reqEnv.Message * 2);
    return respEnv;
}
}
```