Observable Extension API Specification

# Scenarios

## Create IObservable<T> from lambda expression (simple)

The following example creates an instance of IObservable<int> which notifies observers whenever Value property changes.

// obj is an instance of class which implements INotifyPropertyChanged

// with Value property of type int

IObservable<int> source = ObservableEx.Create(() => obj.Value);

## Create IObservable<T> from lambda expression (complex)

The following example creates an instance of IObservable<int> which notifies observers whenever Value property changes on any objects used in the expression (obj1, obj2).

// obj1 and obj2 are instances of class which implements

// INotifyPropertyChanged with Value property of type int

IObservable<int> source = ObservableEx.Create(() => obj1.Value + obj2.Value);

## Create IObservable<T> from lambda expression (property chain)

The following example creates an instance of IObservable<int> which notifies observers whenever Foo property changes on obj object or Bar property changes on obj.Foo object. If obj.Foo is changed to different instance it is checked to see if it implements INotifyPropertyChanged. In case of positive test, notifications which come from it will trigger expression reevaluation and will be propagated to observers.

If obj.Foo is null and Bar property cannot be evaluated all observers are informed about it through OnError method with NullReferenceException. Additionally this terminates the stream of events.

// obj1 is an instance of class which implements INotifyPropertyChanged

// with Foo property of another type which also implements this interface

// with property Bar of type int

IObservable<int> source = ObservableEx.Create(() => obj.Foo.Bar);

# API Design

public static class ObservableEx

{

public static IObservable<TResult>

Create<TResult>(Expression<Func<TResult>> expression);

}

# Functional Specification

|  |  |
| --- | --- |
| Member | public static IObservable<TResult>  Create<TResult>(Expression<Func<TResult>> expression); |
| Behavior | Creates IObservable<T> object from lambda expression supplied. Expression is reevaluated whenever any of the properties which take part in evaluation process changes. In order to detect these moments objects used in expression have to implement INotifyPropertyChanged interface. |
| Exceptions | ArgumentNullException – if expression is null |