Reactive Extensions

Сергей Тепляков
STeplyakov@luxoft.com

О Вашем инструкторе

LUXOFT
Expertise in Software Services

- Сергей Тепляков
- Visual C# MVP, RSDN Team member
- Sergey.Teplyakov@gmail.com
- SergeyTeplyakov.blogspot.com

Цели курса…

LUXOFT
Expertise in Software Services

Слушатели изучат:

- Философию библиотеки Reactive Extensions
- Основы библиотеки TPL
- Использование библиотеки TPL
- Новые возможности C# 5

Необходимая подготовка

LUXOFT
Expertise in Software Services

Слушатели должны:

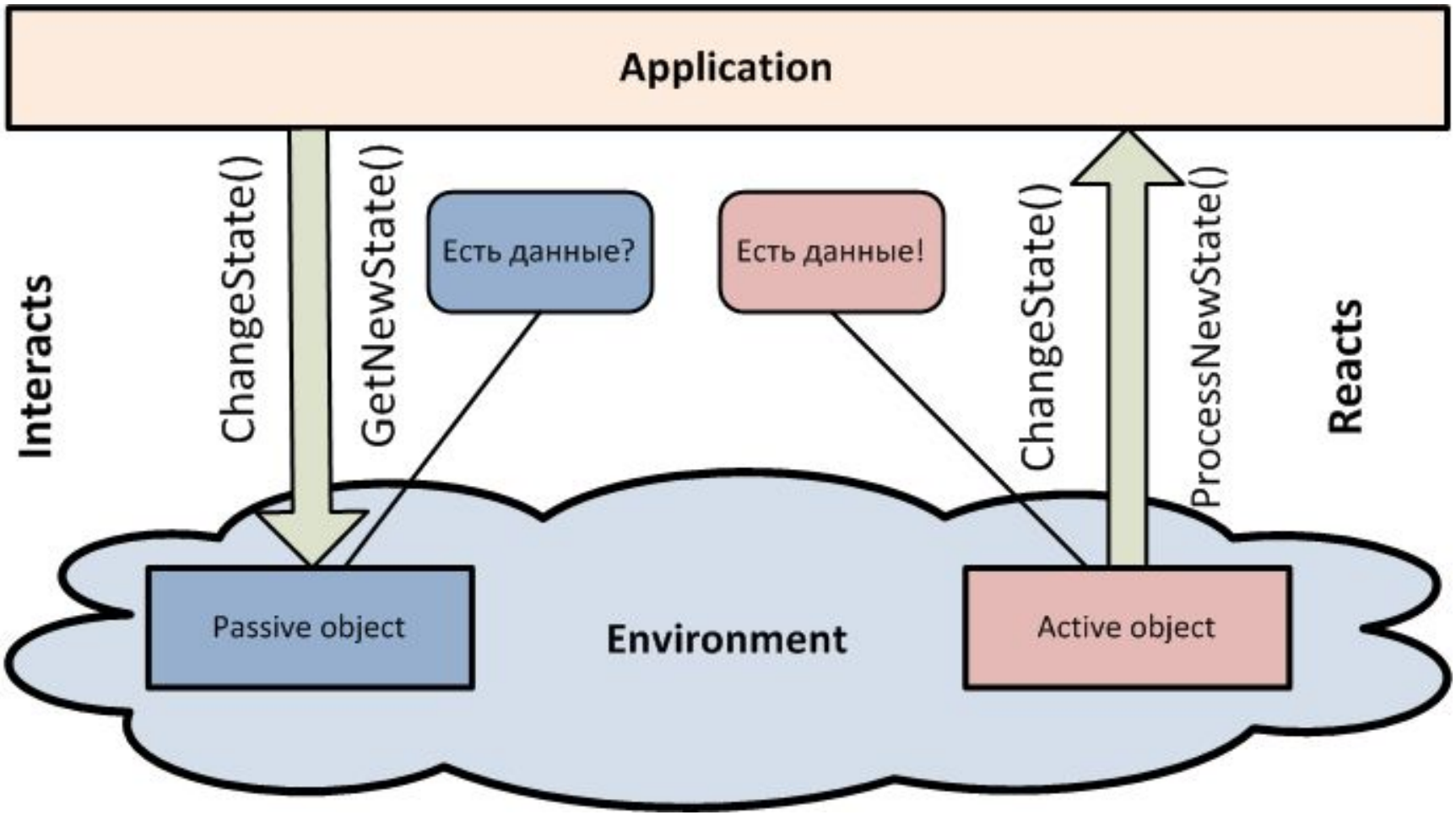
- Быть знакомы с основами языка C# и платформы .Net
- Обладать базовыми знаниями многопоточности
- Быть знакомы с LINQ (Language Integrated Query)

Roadmap

LUXOFT
Expertise in Software Services

- Введение в реактивные расширения
- Дуализм интерфейсов
- Основы Rx
- Observable sequences
- Events и Observables
- Observables и асинхронные операции
- Concurrency
- Новости из Редмонта

Интерактивная и реактивная модель



Реактивное программирование

LUXOFT
Expertise in Software Services

Парадигма программирования, ориентированная на потоки данных и распространение изменений. Это означает, что должна существовать возможность легко выражать статические и динамические потоки данных, а также то, что выполняемая модель должна автоматически распространять изменения сквозь поток данных.

http://ru.wikipedia.org/
/wiki/Реактивное_программирование

WIKIPEDIA
The Free Encyclopedia

Реактивная модель

"Принцип Голливуда" – "Не звоните нам, мы сами вам позвоним"

(Hollywood Principle - Don't call us, we'll call you)

Rx - это

1. набор классов, представляющих собой асинхронный поток данных
2. набор операторов, предназначенных для манипуляции этими данными
3. набор классов для управления многопоточностью

Rx = Observables + LINQ + Schedulers

Зачем все это нужно?

LUXOFT
Expertise in Software Services

Stock tickers

RSS feeds

GPS

Social media

UI events

Server management

Интерфейс IEnumerable

Pull-based последовательности представлены интерфейсом IEnumerable. Это -

- коллекции в памяти (списки, вектора и т. д.)
- бесконечные последовательности (генераторы)
- xml-данные
- результаты запроса к БД

Интерфейс IEnumerable


```
public interface IEnumerable<out T>
{
    IEnumerator<T> GetEnumerator();
}

public interface IEnumerator<out T> : IDisposable
{
    bool MoveNext();
    T Current { get; }
    void Reset();
}
```

Ковариантность из .NET 4.0

Блокирующая операция

Упрощение интерфейса IEnumerator

LUXOFT
Expertise in Software Services

```
public interface IEnumerator<out T> : IDisposable
{
    (T | void | Exception) GetNext();
}
```

T – следующий элемент
void – окончание последовательности
Exception – ошибка

Интерфейс IObservable


```
public interface IObservable<out T>
{
    IDisposable Subscribe(IObserver<T> observer);
}

public interface IObserver<in T>
{
    void OnNext(T value);
    void OnError(Exception error);
    void OnCompleted();
}
```

Pull vs Push

LUXOFT
Expertise in Software Services

Pull модель

IEnumerable<T>
- IEnumerator<T> GetEnumerator()

IEnumerator<T>
- bool MoveNext();
- T Current {get;}
- ~~void Reset();~~

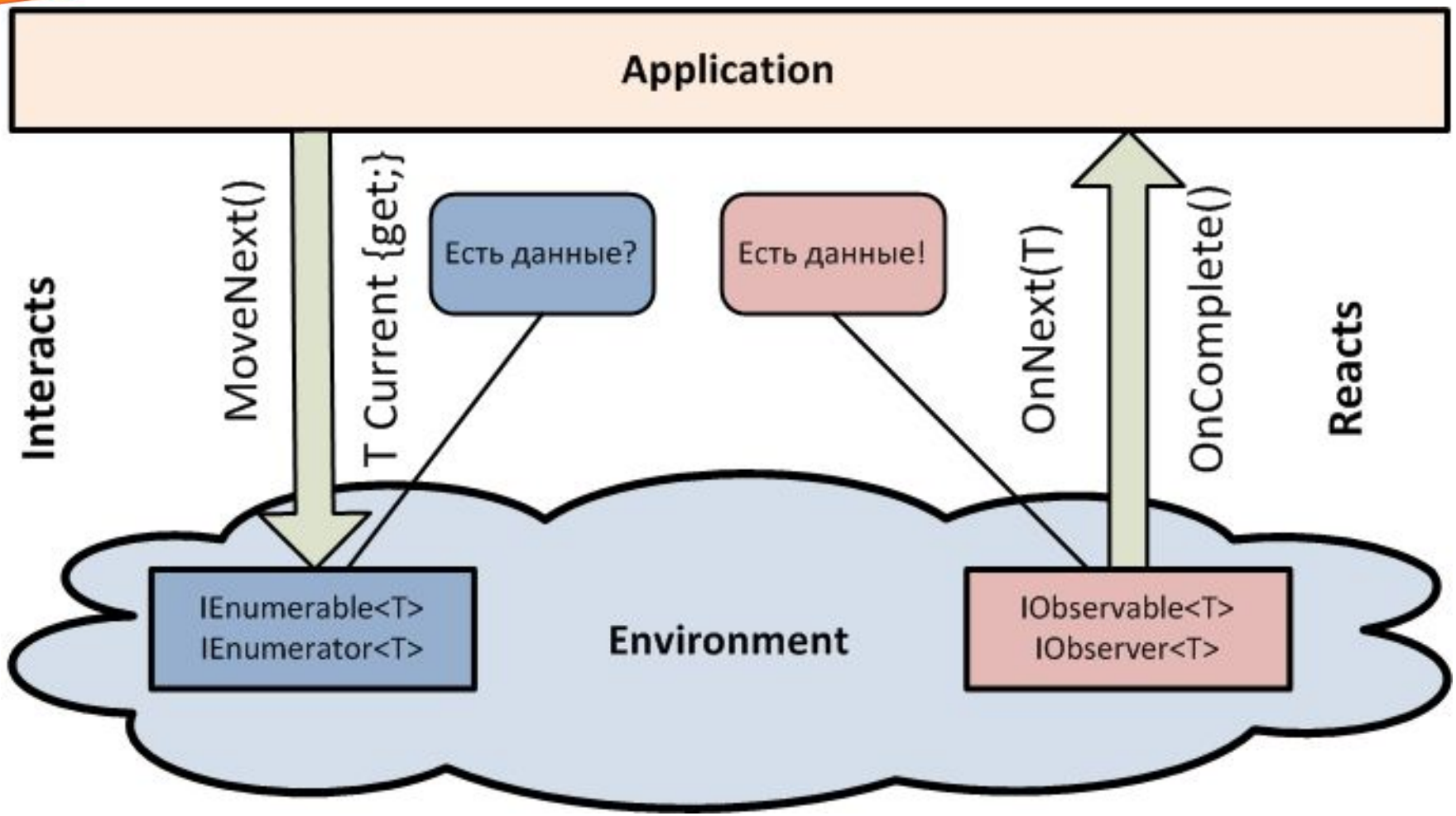
Или
- (T|void|Exception) GetNext();

Push модель

IObservable<T>
- IDisposable Subscribe(IObserver<T>)

IObserver<T>
- void OnNext(T value);
- void OnError(Exception error);
- void OnCompleted();

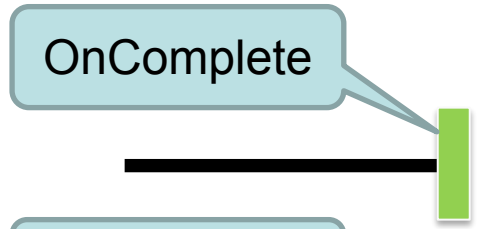
Pull vs Push


Application
Interacts
MoveNext()
T Current {get;}
Есть данные?
Есть данные!
OnNext(T)
OnComplete()
Reacts
IEnumerable<T>
IEnumerator<T>
Environment
IObservable<T>
IObserver<T>

Простые последовательности

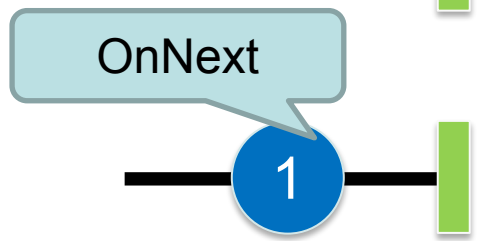
LUXOFT
Expertise in Software Services

```
Observable.Empty<int>();
```



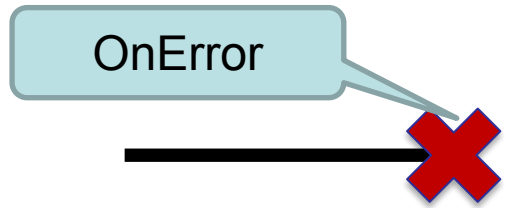

```
new int[] {};
```

```
Observable.Return(42);
```




```
new int[] {42};
```

```
Observable.Throw<int>(ex);
```



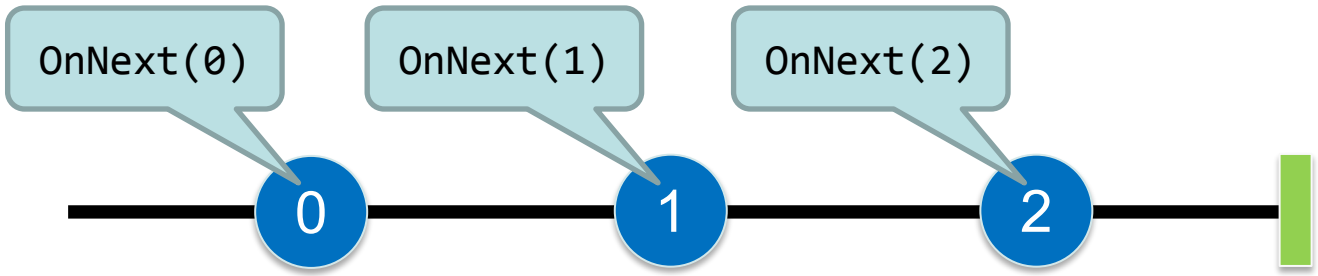
Итератор, генерирующий исключение

```
Observable.Never<int>();
```

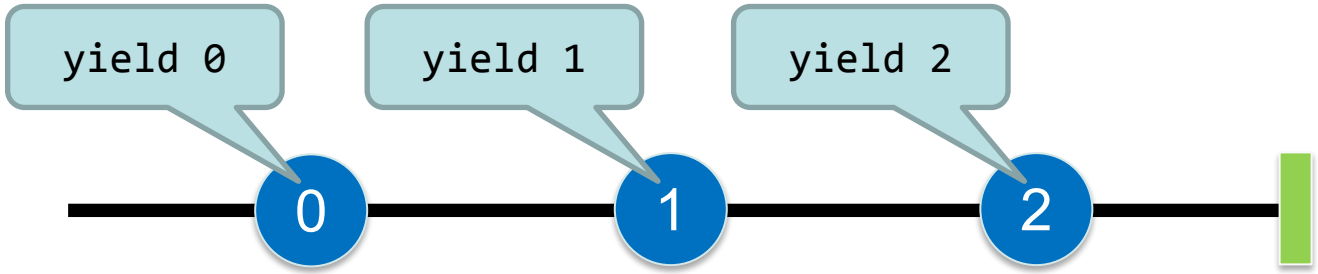
Итератор, который не возвращает управление

Методы Range

```
Observable.Range(0, 3);
```




```
Enumerable.Range(0, 3);
```



Циклы for


```
var xs = Observable.Generate(
            0,
            x => x < 10,
            x => x + 1,
            x => x);


xs.Subscribe(x => {
        Console.WriteLine(x);

    });
```

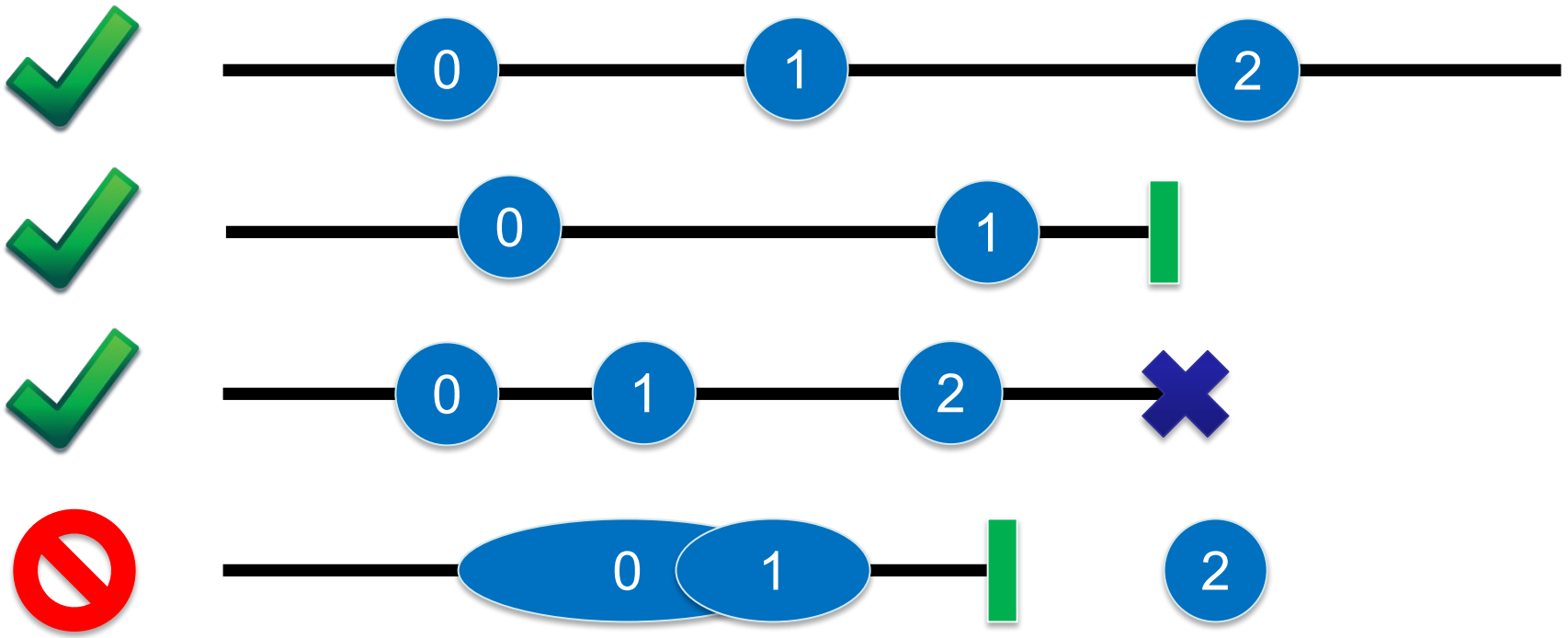
```
var xs = new IEnumerable<int>  {
    for(int i = 0;
        i < 10;
        i++)
      yield return i;
};

foreach(var x in xs) {
    Console.WriteLine(x);
}
```

Предположим, у нас есть синтаксис анонимных итераторов

Контракт «реактивных» последовательностей

LUXOFT
Expertise in Software Services

- Grammar: OnNext* [OnCompleted | OnError]
- Методы наблюдателя вызываются последовательно



Упрощение работы с интерфейсом IObserver

LUXOFT
Expertise in Software Services

```
var xs = Observable.Range(0, 10);
```

1. Только OnNext:

```
xs.Subscribe(/*OnNext(int)*/x => Console.WriteLine(x));
```

2. OnNext и OnCompleted

```
xs.Subscribe(
    /*OnNext(int)*/x => Console.WriteLine(x),
    /*OnComplete*/() => Console.WriteLine("Completed"));
```

3. OnNext и OnError

```
xs.Subscribe(
    /*OnNext(int)*/x => Console.WriteLine(x),
    /*OnError(Exception)*/e => Console.WriteLine("Error: {0}", e));
```

4. OnNext, OnError и OnCompleted

```
xs.Subscribe(
    /*OnNext(int)*/x => Console.WriteLine(x),
    /*OnError(Exception)*/e => Console.WriteLine("Error: {0}", e),
    /*OnCompleted*/() => Console.WriteLine("Completed"));
```

Demo

LUXOFT
Expertise in Software Services

События в .NET

- Объявление

```
event Action<int> E;
```

- Публикация

```
E(42);
```

- Подписка

```
E += x => Console.WriteLine(x);
```

Rx

- Объявление

```
ISubject<int> S = new Subject<int>();
```

- Публикация

```
S.OnNext(42);
```

- Подписка

```
S.Subscribe(x => Console.WriteLine(x));
```

Events vs Observables


```
class Program {

  event Action<int> E;

  static void Main() {
    var p = new Program();

    p.E += x => Console.WriteLine(x);

    p.E(1);
    p.E(2);
    p.E(3);
  }
}
```

```
class Program {

  ISubject<int> S = new Subject<int>();

  static void Main() {
    var p = new Program();

    p.S.Subscribe(x => Console.WriteLine(x));

    p.S.OnNext(1);
    p.S.OnNext(2);
    p.S.OnNext(3);
  }
}
```

Отписка от событий

LUXOFT
Expertise in Software Services

- **Events**

```
static event Action<string> E;

//E += x => Console.WriteLine(x);
Action<string> action = x => Console.WriteLine(x);
E += action;
E -= action;
```

- **Observables**

```
static ISubject<int> S = new Subject<int>();

var token = S.Subscribe(x => Console.WriteLine(x));
token.Dispose();
```

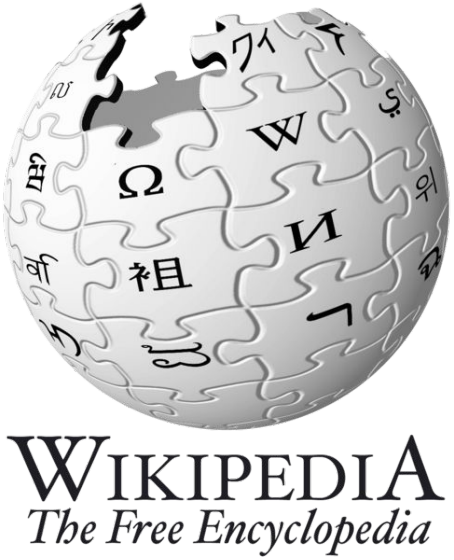
«События» первого класса

LUXOFT
Expertise in Software Services

Объект называют «объектом первого класса» когда он:

- может быть **сохранен** в переменной
- может быть **передан** в функцию как параметр
- может быть **возвращен** из функции как результат
- может быть **создан** во время выполнения программы
- **внутренне самоидентифицируем** (независим от именования)

http://ru.wikipedia.org/wiki/Объект_первого_класса



«События» первого класса

LUXOFT
Expertise in Software Services

```
// Сохранение в переменной
IObservable<string> textChanged = ...;

// Передача в качестве параметра
void ProcessRequests(IObservable<string> input) {...}

// Возвращение в качестве результата
IObservable<int> QueryServer() {...}
```

Возможности наблюдаемых последовательностей

LUXOFT
Expertise in Software Services

```
IObservable<Point> mouseMoves =
    from e in Observable.FromEventPattern<MouseEventArgs>(frm, "MouseMove")
    select e.EventArgs.Location;
```

Преобразование события в поток объектов Point

```
var filtered = mouseMoves.Where(p => p.X == p.Y);
```

Фильтрация «событий»

```
var subscription =
    filtered.Subscribe(p => Console.WriteLine("X, Y = {0}", p.X));
```

Обработка «событий»

```
subscription.Dispose();
```

Отписка от «событий»

Demo

LUXOFT
Expertise in Software Services

Асинхронные операции
Classical Async Pattern

LUXOFT
Expertise in Software Services

```
FileStream fs = File.OpenRead("data.txt");

byte[] buffer = new byte[1024];


fs.BeginRead(buffer, 0, buffer.Length,
                 ar =>
                     {
                         int bytesRead = fs.EndRead(ar);
                         // Обработка данных в массиве buffer
                     },
                 null);
```

Невозможность использования привычных языковых конструкций (using, try/finally etc)

«Вывернутый» поток исполнения

Сложность обработки ошибок, а также чтения и сопровождения кода

FromAsyncPattern

LUXOFT
Expertise in Software Services

```
static int LongRunningFunc(string s)
{
    Thread.Sleep(TimeSpan.FromSeconds(5));
    return s.Length;
}

Func<string, int> longRunningFunc = LongRunningFunc;

Func<string, IObservable<int>> funcHelper =
    Observable.FromAsyncPattern<string, int>(
        longRunningFunc.BeginInvoke, longRunningFunc.EndInvoke);

IObservable<int> xs = funcHelper("Hello, String");
xs.Subscribe(x => Console.WriteLine("Length is " + x));
```

FromAsyncPattern преобразует возвращаемое значение метода в IObservervable<T>

Tasks vs FromAsyncPattern

LUXOFT
Expertise in Software Services

- Задачи (Tasks) – это унифицированный способ представления Single value asynchrony в .Net Framework
- Observables – Multi value asynchrony
- Существует простой способ преобразования Tasks -> Observables

Task -> ToObservable

LUXOFT
Expertise in Software Services

Метод расширения можно написать один раз, а использовать повсюду, а не только с Rx

```
static class FuncExtensions
{
    internal static Task<int> ToTask(this Func<string, int> func, string s)
    {
        return Task<int>.Factory.FromAsync(
            func.BeginInvoke, func.EndInvoke, s, null);
    }
}
Func<string, int> longRunningFunc = LongRunningFunc;


string s = "Hello, String";

IObservable<int> xs = longRunningFunc.ToTask(s).ToObservable();

xs.Subscribe(x => Console.WriteLine("Length is " + x),
    () => Console.WriteLine("Task is finished"));
        }

    }
```

Это более предпочтительный способ, поскольку метод FromAsyncPattern скоро будет устаревшим

Чем Rx не является

- **Rx не заменяет существующей «асинхронности»:**
  - .NET все еще незаменимы
  - Асинхронные методы все так же применимы в библиотеках
  - Задачи представляют собой single value asynchrony
  - Существуют и другие асинхронные источники, как SSIS, PowerShell, etc
- **Но rx…**
  - **Унифицирует** работу
  - Предоставляет возможность **композиции**
  - Предоставляет **обобщенные операторы**
- **Так что rx – это …**
  - Мост!

Demo

LUXOFT
Expertise in Software Services

Управление многопоточность

LUXOFT
Expertise in Software Services

- **Управление многопоточностью осуществляется с помощью интерфейса IScheduler**

```
var xs = Observable.Range(0, 10, Scheduler.ThreadPool);

xs.Subscribe(x => Console.WriteLine(x));
```

Будет исполняться в контексте планировщика

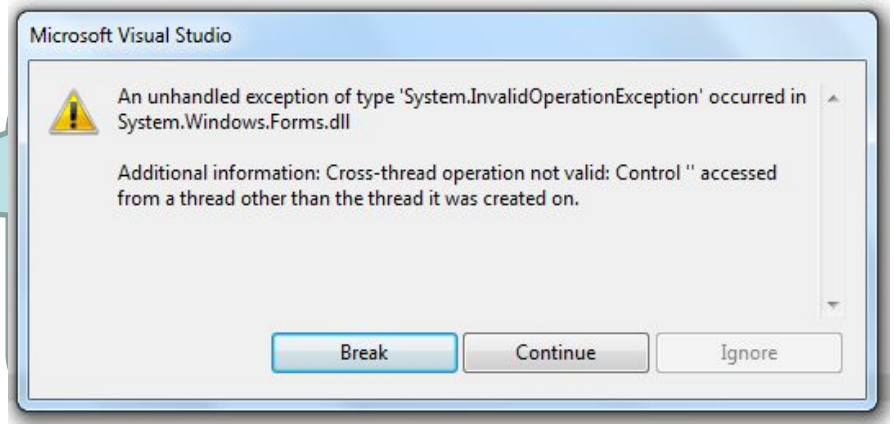
Параметризация с помощью IScheduler доступна в большинстве операций

Синхронизация

LUXOFT
Expertise in Software Services

- **Обновление пользовательского интерфейса**

```
Label lbl = new Label();
Form frm = new Form() {Controls = {lbl}};

var xs = Observable.Return(42, Scheduler.ThreadPool);

xs.Subscribe(x =>
{
    Thread.Sleep(1000);
    lbl.Text = "Result is " + x;
});
```



- **Использование ObserveOn**

```
xs.ObserveOn(frm).Subscribe(x =>
{
    Thread.Sleep(1000);
    lbl.Text = "Result is " + x;
});
```

IScheduler поддерживает разные контексты синхронизации

Demo

LUXOFT
Expertise in Software Services

Что мы изучили?

LUXOFT
Expertise in Software Services

- Введение в реактивные расширения
- Дуализм интерфейсов
- Основы Rx
- Observable sequences
- Events и Observables
- Observables и асинхронные операции
- Concurrency

Experimental vs Stable

- **Две версии библиотеки Reactive Extensions:**
  - Stable
  - Experimental
- **Interactive Extensions**

Где взять?

LUXOFT
Expertise in Software Services

- NuGet
- Web - http://msdn.microsoft.com/en-us/data/gg577610
- Can’t find? Use google☺

Дополнительные ссылки

LUXOFT
Expertise in Software Services

- The Reactive Extensions (Rx)... (Data Developer Center) - http://msdn.microsoft.com/en-us/data/gg577609
- Reactive Extensions (MSDN) - http://msdn.microsoft.com/en-us/library/hh242985(v=vs.103).aspx
- Rx Team Blog - http://blogs.msdn.com/b/rxteam/
- Реактивные расширения и асинхронные операции - http://sergeyteplyakov.blogspot.com/2010/11/blog-post.html

Roslyn CTP is awailable!

LUXOFT
Expertise in Software Services

http://msdn.microsoft.com/ru-ru/roslyn

Вопросы?

LUXOFT
Expertise in Software Services

Горячие и холодные последовательности

LUXOFT
Expertise in Software Services


LUXOFT
Expertise in Software Services