Современная платформа .NET C#, или как упростить Ваш проект?

Alex Rudnitsky alex@payproglobal.com

Знакомство

- с 2001 Delphi
- с 2004 PHP
- с 2005 Ruby, Python
- с 2006 Java
- с 2007 C#

Эволюция C#

- C# 1.0 January 2002
- C# 1.2 April 2003
- C# 2.0 November 2005
- C# 3.0 November 2007 (3.5 появляется LINQ)
- C# 4.0 April 2010
- C# 5.0 August 2012

C# 2.0 November 2005

- Generics
- Partial types
- Anonymous methods
- Iterators
- Nullable types
- Private setters (properties)
- Method group conversions (delegates)
- Covariance and Contra-variance
- Static classes

C# 3.0 November 2007

- Implicitly typed local variables
- Object and collection initializers
- Auto-Implemented properties
- Anonymous types
- Extension methods
- Query expressions
- Lambda expressions
- Expression trees
- Partial Methods

C# 4.0 April 2010

- Dynamic binding
- Named and optional arguments
- Generic co- and contravariance
- Embedded interop types ("NoPIA")

C# 5.0 August 2012

- Asynchronous methods
- Caller info attributes

C# 6.0 Future

- Import type members into namespace
- Succinct syntax for primary constructors
- Readonly properties
- Property expressions (property lambdas)
- Method expressions
- Parameter arrays for IEnumerable interfaces
- Succinct null checking
- Multiple return values
- Constructor type inference

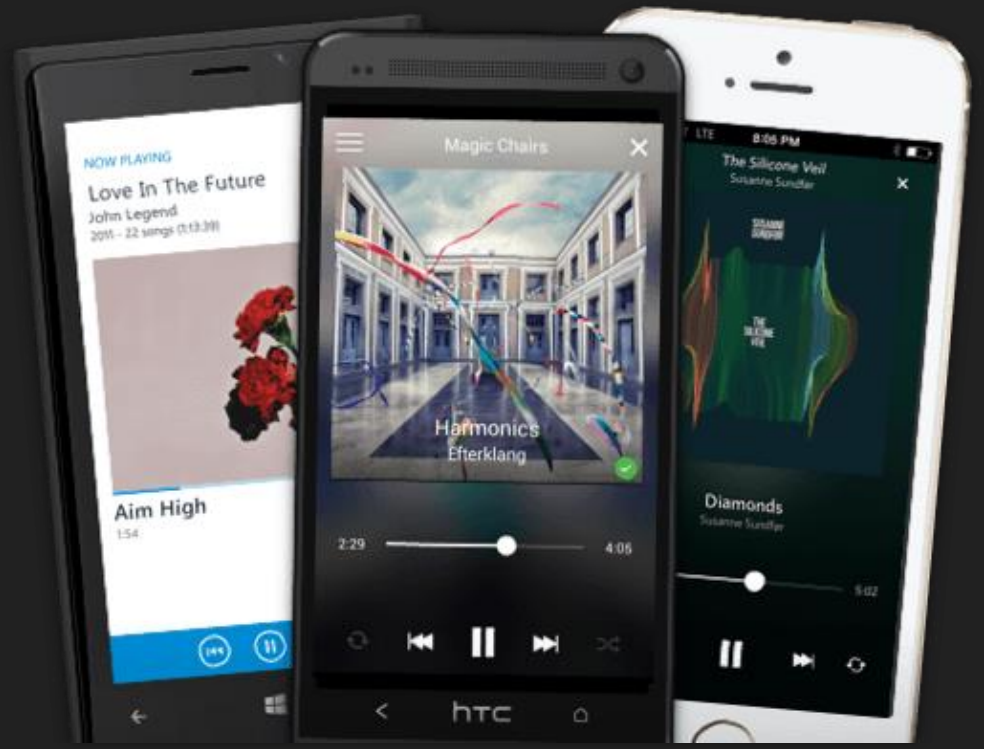
Поддерживаемые платформы

- Windows - .NET (Windows Forms, XAML, HTML5)
- Windows Phone - .NET Mobile (XAML, HTML5)
- Linux – Mono (GTK#, QtSharp, wxNet, XWT)
- Mac OS – Mono (MonoMac, XWT)
- iOS – Mono Xamarin
- Android - Xamarin

Unity 3D / 2D

- Игровой движок
- Основной язык написания сценариев – C#
- Мультиплатформенный инструмент для разработки двух- и трёхмерных приложений и игр
- Поддерживаемые платформы: iOS, Android, Windows, BlackBerry 10, OS X, Linux, web browsers, Flash, PlayStation 3, PlayStation Vita, Xbox 360, Windows Phone 8, и Wii U

Мобильные приложения на C#



http://xamarin.com

LINQ – упрощение кода проекта

- LINQ - Language-Integrated Query
- Проект Microsoft по добавлению синтаксиса языка запросов, напоминающего SQL, в языки программирования платформы .NET Framework.

LINQ – упрощение кода проекта

- LINQ To Objects
- LINQ To SQL
- LINQ to DataSet
- LINQ to Xml

Операции LINQ:

- Не отложеные (выполняются в момент вызова операции)
- Отложеные (выполняются в момент первого обращения к результату операции)

Особенности LINQ

- LINQ - ориентирован на запросы, возвращающие набор подходящих объектов, единственный объект или подмножество полей из объекта либо набора объектов.
- В LINQ этот возвращенный набор называется последовательностью (sequence). Большинство последовательностей LINQ имеют тип IEnumerable<T>, где T — тип данных объектов, находящихся в последовательности
- LINQ - это не только нечто связаное только с запросами, предпочтительнее воспринимать LINQ как механизм итерации данных (data iteration engine)

LINQ на практике

- Приходилось ли вам когда-нибудь вызывать метод, возвращающий данные в структуре, которую затем приходилось преобразовывать в еще одну структуру данных, прежде чем передать другому методу?

LINQ на практике

- Предположим, например, что вызывается метод A, и этот метод возвращает массив типа string, содержащий числовые значения в виде строк.
- Затем нужно вызвать метод B, но метод B требует массива целых чисел.
- Обычно приходится организовывать цикл для прохода по массиву строк и наполнения вновь сконструированного массива целых чисел.

Краткий пример мощи Microsoft LINQ

- Предположим, что имеется массив строк, которые приняты от метода A, как показано ниже
- string[] numbers = { "40", "2012", "176", "5" };
- // Преобразуем массив строк в массив типа int используя LINQ
  int[] nums = numbers.Select(s => Int32.Parse(s)).ToArray();
- foreach (int n in nums) Console.Write(n + " ");
  // выведет: 40 2012 176 5
- Вот и все. Что может быть проще?

Краткий пример мощи Microsoft LINQ

- Возможно, вы подумали, что просто в строках отброшены ведущие пробелы. Но убедит ли вас, если отсортировать результат? Если бы это были по-прежнему строки, то 5 окажется в конце, а 176 — в начале.
- string[] numbers = { "40", "2012", "176", "5" };
- // Преобразуем массив строк в массив типа int и сортируем по возрастанию используя LINQ
  int[] nums = numbers.Select(s => Int32.Parse(s)).OrderBy(s => s).ToArray();
- // результат: 5 40 176 2012

Чуть более сложный пример

- Есть класс Employee и метод возвращающий всех сотрудников
- Есть класс Contact с определенным в нем методом, публикующим контакты.
- Задача: опубликовать всех сотрудников в виде контактов

Класс Employee

```
public class Employee
{
    public int id;
    public string firstName;
    public string lastName;

    public static ArrayList GetEmployees()
    {
        ArrayList al = new ArrayList();
        // В реальном коде коллекция заполнялась бы из базы данных
        al.Add(new Employee { id = 1, firstName = "Alexandr", lastName = "Erohin" });
        al.Add(new Employee { id = 2, firstName = "Alexey", lastName = "Volkov" });
        al.Add(new Employee { id = 3, firstName = "Dmitry", lastName = "Moiseenko" });
        return (al);
    }
}
```

Класс Contact

```
public class Contact
{
    public int id;
    public string Name;

    public static void PublishContacts(Contact[] contacts)
    {
        foreach (Contact c in contacts)
            Console.WriteLine("Contact Id: {0} Contact: {1}", c.id, c.Name);
    }
}
```

Решение и результат

```
static void Main()
{
    ArrayList al = Employee.GetEmployees();
    Contact[] contacts = al
        .Cast<Employee>()
        .Select(e => new Contact
        {
            id = e.id,
            Name = string.Format("{0} {1}", e.firstName, e.lastName)
        })
        .ToArray<Contact>();

    Contact.PublishContacts(contacts);
    Console.ReadLine();
}
```

Пример фильтрации данных

```
Northwind db = new Northwind(@"Data Source=.\SQLEXPRESS;Initial Catalog=Northwind");

var allCustomersOrders = db.Customers
    .Where(x => x.Country == "USA" && x.Region == "WA")
    .SelectMany(x=>x.Orders);

Console.WriteLine(allCustomersOrders.GetType());
```

LINQ предоставляет операции для

- Агрегации (Aggregate, Average, Min, Max, Count, Sum)
- Преобразования (AsEnumerable, Cast, ToArray, ToLookup, ToDictionary, ToList)
- Конкатенации (Concat)
- Доступа к элементам (DefaultEmpty, ElementAt, ElementAtOrDefault, First, Last, FirstOrDefault, LastOrDefault, Single, SingleOrDefault)
- Работы со множествами (Except, Distinct, Interest, Union)
- Генерации (Empty, Range, Repeat)
- Группирования (GroupBy)

LINQ предоставляет операции для

- Соединения (Join, GroupJoin)
- Упорядочивания (OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse)
- Проекции (Select, SelectMany)
- Разбиения (Skip, SkipWhile, Take, TakeWhile)
- Ограничения (Where)
- Квантификации (Any, All, Contains)
- Эквивалентности (SequanceEqual)

Упрощайте проект с C# !