Оглавление

[UI 3](#_Toc115949)

[Меню 3](#_Toc115950)

[Меню Edit 3](#_Toc115951)

[Языки программирования 3](#_Toc115952)

[Редактор кода 3](#_Toc115953)

[Окно результатов 3](#_Toc115954)

[Автоматическое суммирование числовых данных 3](#_Toc115955)

[Горячие клавиши 4](#_Toc115956)

[Unsafe код 4](#_Toc115957)

[Поддержка NuGet 4](#_Toc115958)

[Тонкости 5](#_Toc115959)

[Хостинг кода скрипта 5](#_Toc115960)

[Extension методы 5](#_Toc115961)

[Disassemble() 5](#_Toc115962)

[Dump() 5](#_Toc115963)

[DumpTrace() 5](#_Toc115964)

[OnDemand() 5](#_Toc115965)

[ToImage() 5](#_Toc115966)

[Собственные extension методы 5](#_Toc115967)

[Классы 6](#_Toc115968)

[Класс DumpContainer 6](#_Toc115969)

[Класс Hyperlinq 6](#_Toc115970)

[Статический класс Util 6](#_Toc115971)

[Util.Cmd() 6](#_Toc115972)

[Util.ClearResults() 6](#_Toc115973)

[Util.CurrentQueryPath 6](#_Toc115974)

[Util.GetPassword() 6](#_Toc115975)

[Util.Highlight() 6](#_Toc115976)

[Util.HorizontalRun() 7](#_Toc115977)

[Util.Image() 7](#_Toc115978)

[Util.NewProcess 7](#_Toc115979)

[Util.Progress 7](#_Toc115980)

[Util.RawHtml() 7](#_Toc115981)

[Util.ReadLine() 7](#_Toc115982)

[Util.Run() 8](#_Toc115983)

[Util.WriteCsv() 8](#_Toc115984)

[Класс Util.ProgressBar 8](#_Toc115985)

# UI

При вставке из буфера обмена кода с using предлагает их добавить в Query Properties.

You can flag a connection as 'Production', via the connection properties dialog (or directly from the context menu). The editor then displays an orange warning banner on the left margin whenever you use that connection, to remind you that you're working with production data.

Клавиша F5 выполняет выделенный текст.

## Меню

В данном разделе будут описаны интересные пункты меню LINQPad

### Меню Edit

#### Paste as Escaped String

Вставляет текст из буфера обмена в виде C# строки с escape последовательностями.

## Языки программирования

* Expression
* Statement(s)
* Program
  + Если Main имеет возвращаемое значение, то оно автоматически печатается в окно вывода
  + В LINQ 5 для Main можно использовать модификатор async

## Редактор кода

В режиме «C# Program» можно использовать директивы #region … #endregion.

## Окно результатов

### Автоматическое суммирование числовых данных

При выводе коллекции типов с числовоми полями, их значения будут автоматически просуммированы. Выполним такой код:

DriveInfo.GetDrives()

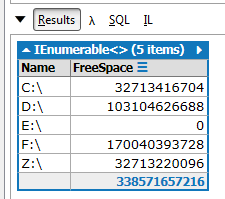
.Select(di => new {

di.Name,

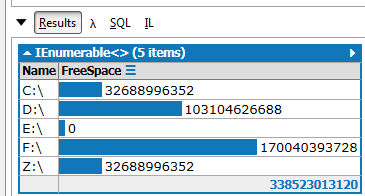
FreeSpace = di.IsReady ? di.AvailableFreeSpace : 0}

).Dump();

Результат выполнения:



Если нажать на иконку  рядом с колонкой FreeSpace, то LIQNPad покажет полосу для визуальной оценки числа.



## Горячие клавиши

По умолчанию LINQPad настроен на использование некоторых клавиатурных комбинаций Visual Studio. (Edit -> Preferences, Advanced, Use Visual Studio shortcut keys)

* Ctrl+K, C закоментировать выделенный блок
* Ctrl+K, U снять коментарий с выделенного блока

# Unsafe код

LINQPad позволяет использовать unsafe код.

# Поддержка NuGet

Бесплатная версия LINQPad имеет ограниченную поддержку NuGet пакетов и разрешает использовать только пакеты с LINQPad скриптами. В пакете должна быть папка linqpad-samples и он должен быть [помечен тегом linqpad-samples](http://www.nuget.org/packages?q=Tags%3A%22linqpad-samples%22).

Можно обойти это ограничение сохранив скрипт в файле с расширением linq и добавив элемент NuGetReference внутри элемента Query.

<NuGetReference>morelinq</NuGetReference>

<NuGetReference Version="1.4.0">morelinq</NuGetReference>

# Тонкости

## Хостинг кода скрипта

Для каждой закладки создается отдельный дочерний процесс LINQPad в который и загружается сборка. Это также означает что скрипты выполнятся в разных доменах приложения. LINQPad можно попросить создавать новый процесс при выполнении скрипта используя свойство Util.NewProcess.

# Extension методы

## Disassemble()

С помощью метода Disassemble() можно декомпилировать метод в IL код.

typeof (Uri).GetMethod ("GetHashCode").Disassemble().Dump();

## Dump()

С помощью Dump() мы выводим

имяПеременной.Dump(int) уровень вложенности

имяПеременной.Dump(string) название секции

Dump() возвращает значение которое не вывел позволяя строить цепочки вызовов:

string s = "xyz-zyx";

s = s.Replace("x", "1").Dump()

.Replace("y", "2").Dump()

.Replace("z", "3").Dump("Результат");

LINQPad также позволяет использовать метод Console.WriteLine() для вывода данных в окно результатов.

## DumpTrace()

Метод DumpTrace() работает аналогично методу Dump(), но выводит данные только если определен символ TRACE.

## OnDemand()

Вывод данных только после щелчка по ссылке в окне результатов

Util.OnDemand ("Generate keypair", () => RSA.Create().ToXmlString(true)).Dump()

Customers.OnDemand()

## ToImage()

ToImage(this System.Data.Linq.Binary)

## Собственные extension методы

В файле “My Extensions”, доступном в окне My Queries, вы можете размещать собственные extension методы и они автоматически станут доступны во всех запросах.

# Классы

## Класс DumpContainer

Доступен начиная с версий 2.47 и 4.47.

Данный класс позволяет вывести в окно результатов блок с заменяемым содержимым. Не забудьте вывести экземпляр DumpContainer с помощью метода Dump()!

var dc = new DumpContainer();  
dc.Content = "Первое значение";  
dc.Dump();  
"Строка после DumpContainer".Dump();  
Thread.Sleep(1500); // Имитация длительной операции  
dc.Content = "Второе значение";  
"Вторая строка после DumpContainer".Dump();

## Класс Hyperlinq

The Hyperlinq class now accepts file paths, as well as URIs and query snippets.

new Hyperlinq (() => MessageBox.Show ("Hello, world!"), "Click me")

Additionally, the Hyperlinq class now supports an Action delegate that will be called when you click the link, allowing you to react to it in code and not just link to external webpages.

## Статический класс Util

### Util.Cmd()

Метод Cmd() выполняет команду командного процессора (cmd) и выводит результат на экран.

Util.Cmd(@"dir C:\");

### Util.ClearResults()

Метод ClearResults() очищает окно результатов.

### Util.CurrentQueryPath

Данное свойство возвращает полный путь к файлу текущего запроса. Если запрос не был сохранен, то свойство вернет null.

Util.ClearResults();

### Util.GetPassword()

Метод GetPassword() возвращает пароль с указанным именем ранее созданным в меню File \ Password Manager. Пароли хранятся в зашифрованном виде в папке %locaappdata%.

string pwd = Util.GetPassword("DlinkRouter");

### Util.Highlight()

Метод Highlight() позволяет подсвечивать текст выводимый в окно результатов.

Util.Highlight("Внимание!").Dump();

### Util.HorizontalRun()

Метод HorizontalRun() выводит переданные ему аргументы на одной строке. Первый аргумент имеет тип bool и указывает нужно ли добавлять пробелы между значениями.

Пример №1:

Util.HorizontalRun(  
 true,  
 "Переменная окружения LOCALAPPDATA равна",  
 Environment.GetEnvironmentVariable("LOCALAPPDATA")  
).Dump();

### Util.Image()

Метод Image() отображает картинку с заданным адресом в окне результатов.

var url = "http://chart.apis.google.com/chart?cht=p&chs=350x140"  
 + "&chd=t:15.183,15.365,4.160,0.802"  
 + "&chl=Таможенный союз|ЕС|Украина|Бразилия";  
Util.Image(url).Dump("Экспорт из РБ в 2012 году, млн. долларов");

### Util.NewProcess

Если присвоить этому свойству true, то следующее выполнение скрипта будет идти в новом процессе и новом домене приложения.

### Util.Progress

Свойство Progress позволяет выводить прогресс операции в progress bar под окном результатов. Присваивайте ему значение от 0 до 100.

### Util.RawHtml()

Метод RawHtml() разбирает переданную html строку для отображения.

string html ="<h1>Заголовок</h1><p style='color:red'>Текст</p>";  
Util.RawHtml(html).Dump();

### Util.ReadLine()

Метод ReadLine() позволяет пользователю ввести данные с клавиатуры. Варианты использования:

* string input = Util.ReadLine();  
  Ввод строки
* string input = Util.ReadLine("Введите строку");  
  Ввод строки. Поверх поля ввода выводится указанное пояснение.
* string input = Util.ReadLine("Введите строку", "abc");  
  Ввод строки. Поверх поля ввода выводится указанное пояснение. Вторым аргументом передается значение по умолчанию.

Метод ReadLine позволяет вводить типизированные данные:

* int num = Util.ReadLine<int>();
* int num = Util.ReadLine<int>("Введите целое число");
* int num = Util.ReadLine<int>("Введите целое число", 12);

В данных примерах от пользователя ожидается целое число. Если он введет значение которое невозможно преобразовать в int, то LINQPad попросить повторить ввод.

Для ввода строковых значений можно также использовать метод Console.ReadLine().

### Util.Run()

Метод Run() исполняет указанный LINQPad файл.

QueryResultFormat.Text

QueryResultFormat.Html

### Util.WriteCsv()

Доступен с версии v4.45.05 (beta)

Util.WriteCsv (Customers, @"c:\temp\customers.csv");

**Советы**:

* Если таблица содержит много данных, то поставьте ObjectTrackingEnabled = false; перед запросом чтобы избежать кэширования данных в памяти.

## Класс Util.ProgressBar

Класс Util.ProgressBar выводит progress bar в окне результатов с настраиваемым пояснением.

void Main()

{

var pb1= new ProgressBar();

Thread.Sleep(50);

pb1.Update(50, "Doing something"); Thread.Sleep(550);

pb1.Update(100); Thread.Sleep(50);

}

// Define other methods and classes here

public class ProgressBar

{

Util.ProgressBar prog;

public ProgressBar()

{

Init("Processing");

}

private void Init(string msg)

{

prog = new Util.ProgressBar(msg).Dump();

prog.Percent=0;

}

public void Update(int percent)

{

Update(percent, null);

}

public void Update(int percent, string msg)

{

prog.Percent=percent;

if (String.IsNullOrEmpty(msg))

{

if (percent>99) prog.Caption="Done.";

}

else

{

prog.Caption=msg;

}

}

}