



# VISUAL STUDIO 2010 И .NET 4.0

*Калита Роман  
TaskManagementSoft*



# ЧТО НОВОГО В VISUAL STUDIO 2010

# Большое количество НОВОВВЕДЕНИЙ

- Breakpoint Labeling
- Breakpoint Searching
- Breakpoint Import/Export
- Dynamic Data Tooling
- WPF Tree Visualizer
- Call Hierarchy
- Improved WPF Tooling
- Historical Debugging
- Mini-Dump Debugging
- Quick Search
- Better Multi-Monitor Support
- Highlight References
- Parallel Stacks Window
- Parallel Tasks Window
- Document Map Margin
- Generate From Usage
- Concurrency Profiler
- Inline Call Tree
- Extensible Test Runner
- MVC Tooling
- Web Deploy
- JQuery Intellisense
- SharePoint Tooling
- HTML Snippets
- Web.config Transformation

# Большое количество нововведений

- Общие улучшения
- Отладка
- Параллелизм
- Веб
- Расширяемость





# ОБЩИЕ УЛУЧШЕНИЯ

# Подсветка ссылок

- CTRL+SHIFT+  
DOWN ARROW (вперед)  
  
CTRL+SHIFT+  
UP ARROW  
(назад)
- Автоматическая  
подсветка символов
- Используется с любым  
определенным символом  
– имена функций,  
переменных, классов,  
свойств и т.д.

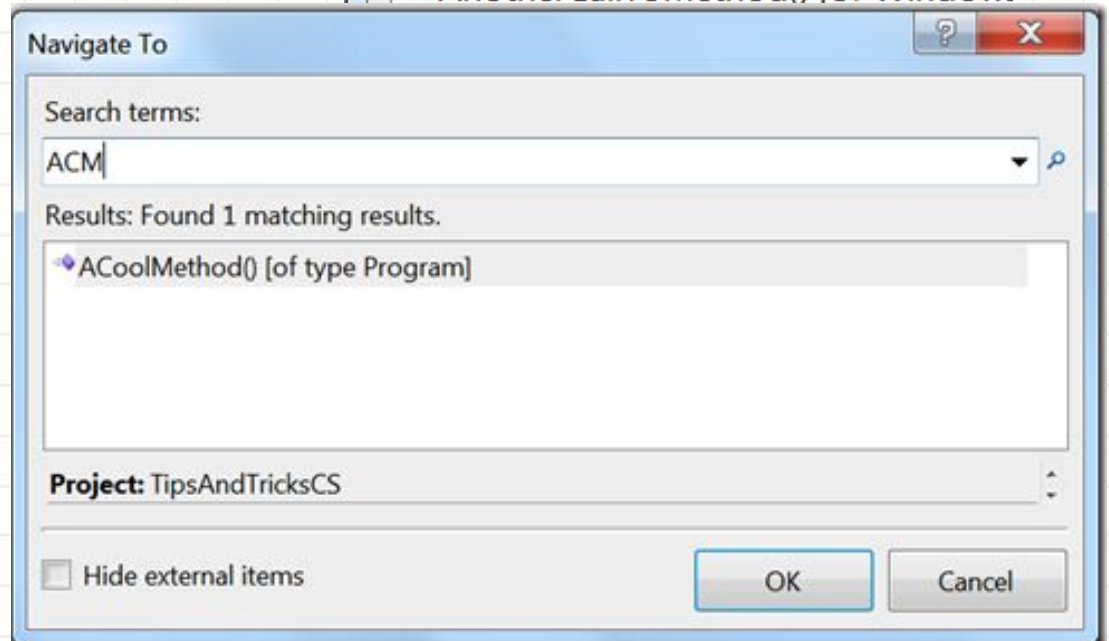
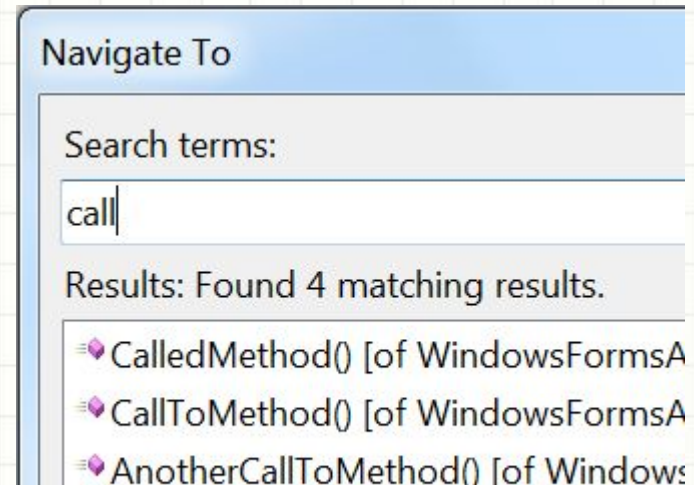
```
private void SomeMethod()  
{  
    MessageBox.Show("Wheeee");  
    CalledMethod();  
}
```

```
private void CallToMethod()  
{  
    SomeMethod();  
}
```

```
private void AnotherCallToMethod()  
{  
    SomeMethod();  
}
```

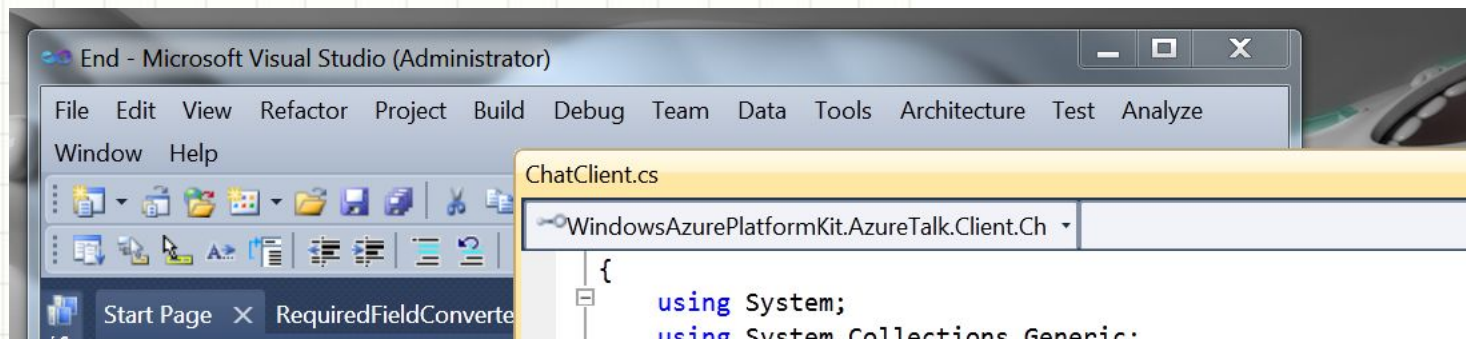
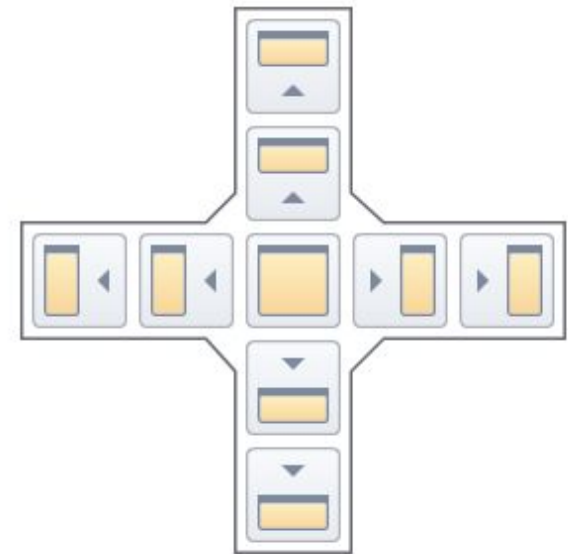
# Navigate To

- CTRL + ,
- ПОИСК СИМВОЛОВ  
ПО ИМЕНИ



# Docking

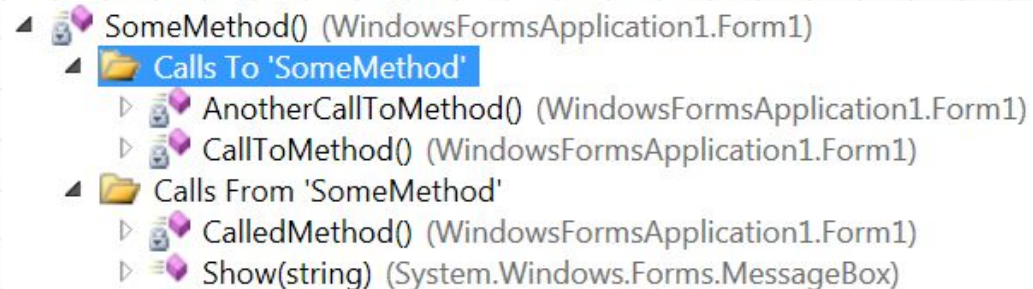
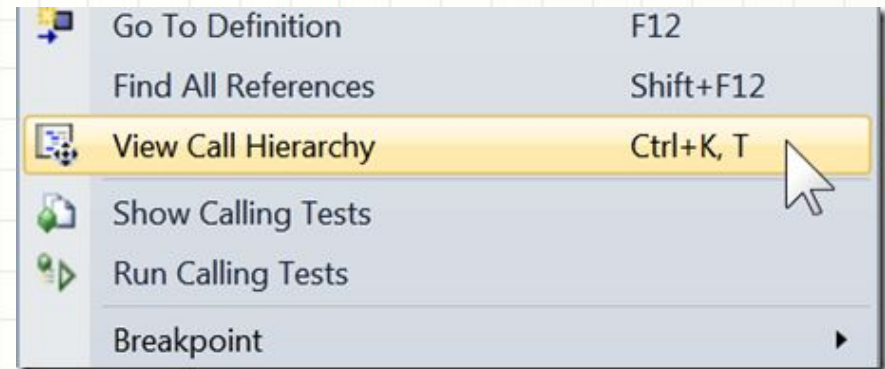
- НОВЫЙ docking diamond
- Окна докируются где угодно
- Можно вынести окно кода за IDE





# Call Hierarchy (C# only)

- CTRL + K, T
- Просмотр  
ВЫЗВОВОВ к/из  
метода
- Удобный способ  
ОТСЛЕЖИВАТЬ  
ВЫЗОВЫ в design  
time



# Zoom

- CTRL + колесико  
МЫШИ
- Увеличение/уменьшение  
размера  
кода
- Удобно например  
при парном  
программировании  
(или подобных  
сценариях)

```
private void
```

```
{
```

```
private void  
{
```

# Generate from Usage

- Используется для автоматической генерации кода, например заглушек
- Использование классов и их членов до их полного определения
- TDD стиль написания кода

```
Person bob;
```



Generate class for 'Person'

Generate other...

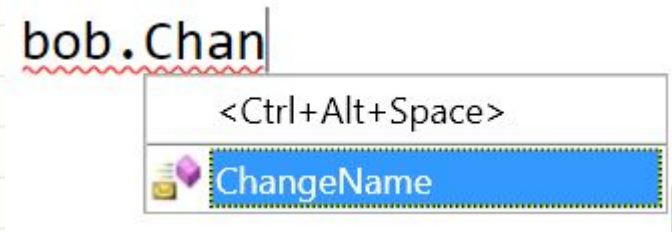
```
bob.ChangeName("Bubba");
```



Generate method stub for 'ChangeName' in 'WindowsFormsApplication1.Person'

# Intellisense Suggestion Mode

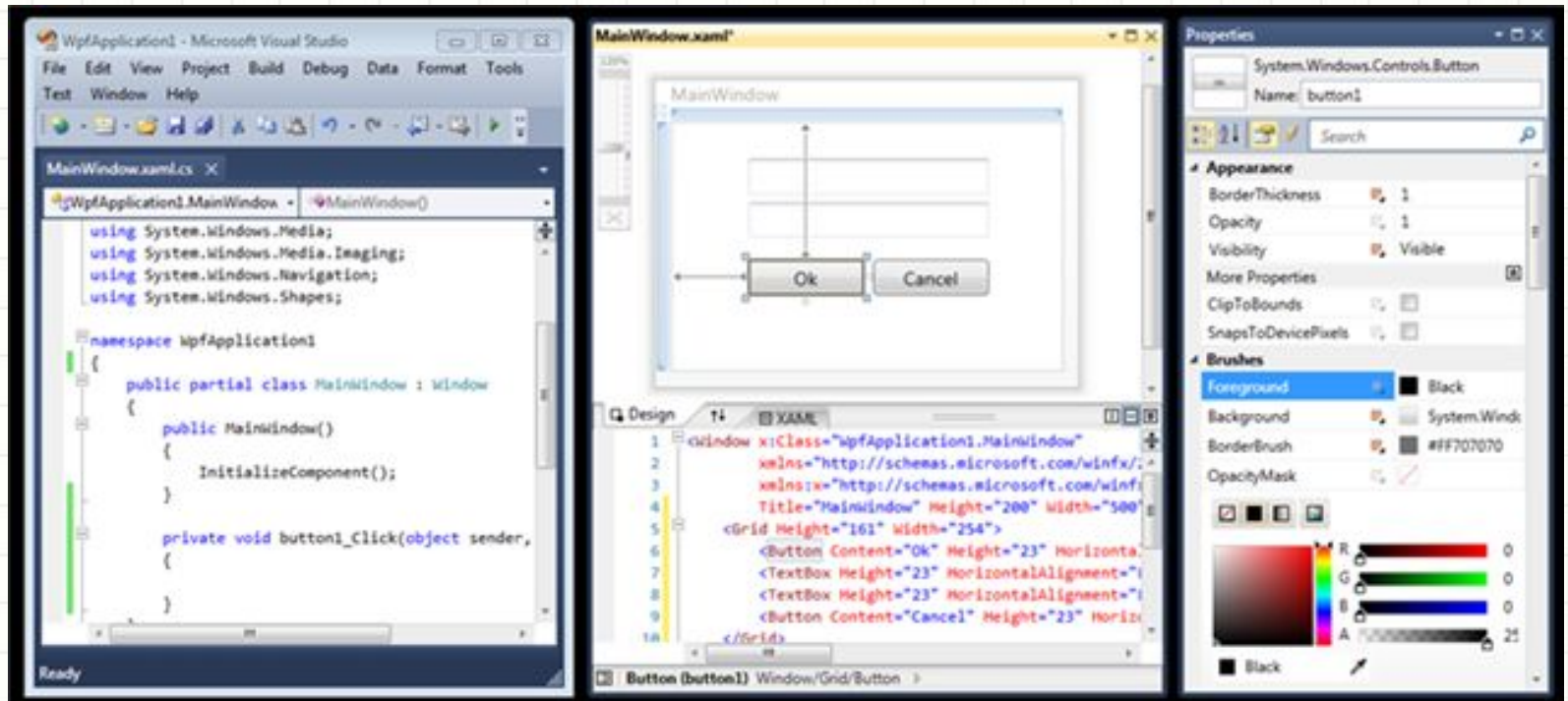
- CTRL + ALT + SPACE
- Когда классы и их члены используются до их полного определения
- Избегает ситуаций kHelps to reduce situations where IntelliSense inserts unintended text into the editor
- TDD стиль написания кода





# Поддержка нескольких МОНИТОРОВ

- Все окна теперь можно сделать плавающими
- Даже Code Editor и Design View








# УЛУЧШЕНИЯ ОТЛАДКИ





# Breakpoints

- Можно добавлять метки к точкам останова

Name	Labels
<input checked="" type="checkbox"/>  Form1.cs, line 29 character 13	Call ChangeName
<input checked="" type="checkbox"/>  Form1.cs, line 39 character 13	Show Message
<input checked="" type="checkbox"/>  Form1.cs, line 45 character 13	Call SomeMethod

Search:

- Можно искать по точкам останова

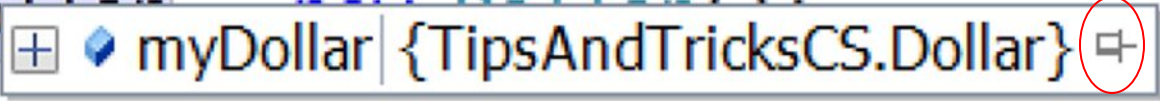
    Columns Search:

Export all breakpoints matching the current search criteria

- Импорт/ Экспорт точек

# Плавающие подсказки (Data Tips)

```
myDollar = new Dollar();
```



```
{  
    Dollar myDollar = new Dollar();
```







# Call Hierarchy (C# only)

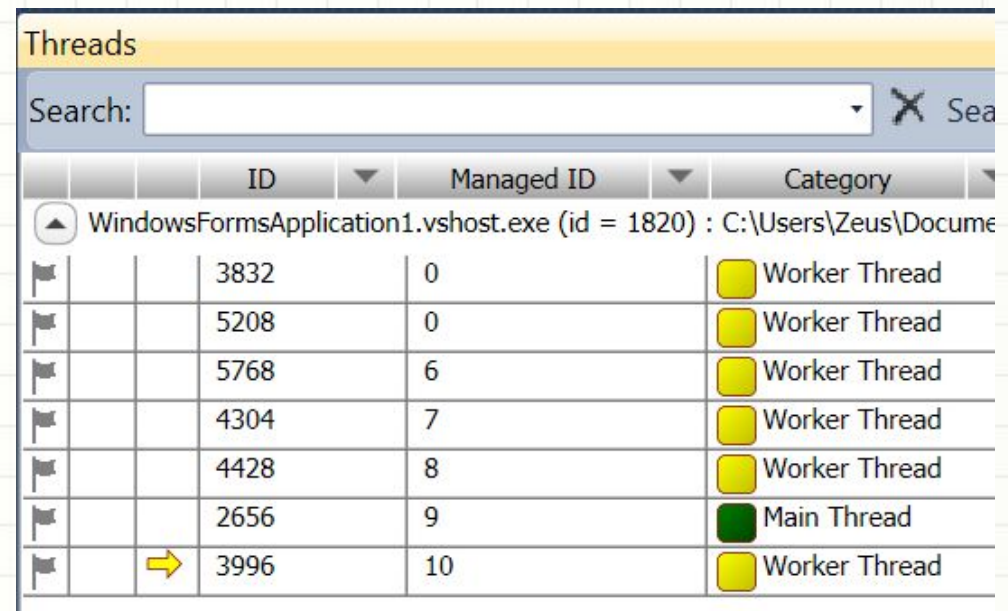
# Окно потоков

- Полностью redesigned

- Фильтрация, поиск по call-stack, сворачивание, группировки

- Новые колонки:

- Affinity masks
- Process names
- Managed IDs

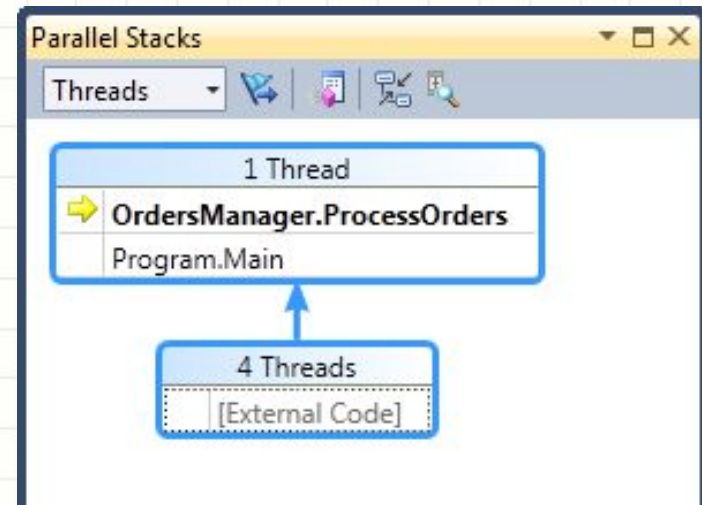


The screenshot shows the 'Threads' window in Windows Task Manager. It features a search bar at the top, a table with columns for 'ID', 'Managed ID', and 'Category', and a list of threads for the process 'WindowsFormsApplication1.vshost.exe (id = 1820)'. The threads are listed with their IDs, Managed IDs, and categories (Worker Thread or Main Thread). A yellow arrow points to the thread with ID 3996.

	ID	Managed ID	Category
WindowsFormsApplication1.vshost.exe (id = 1820) : C:\Users\Zeus\Docume			
3832	0	Worker Thread	
5208	0	Worker Thread	
5768	6	Worker Thread	
4304	7	Worker Thread	
4428	8	Worker Thread	
2656	9	Main Thread	
3996	10	Worker Thread	

# Окна Parallel Stacks, Parallel Tasks

- Новые окна для визуализации и отладки параллельного кода на C++, C#, или Visual Basic
- Parallel Stacks – несколько call stack одновременно
- Parallel Tasks просмотр параллельных задач и их статусу



The screenshot shows the 'Parallel Tasks' window, which displays a table of task information. The table has three columns: 'Id.', 'Status', and an icon column. The tasks are listed as follows:

	Id.	Status
▽	6	ⓘ Scheduled
▽	7	ⓘ Scheduled
▽	8	ⓘ Scheduled
▽	9	ⓘ Scheduled
▽	10	ⓘ Scheduled
▽	1	ⓘ Waiting
▽ →	2	▶ Running
▽	3	ⓘ Waiting-Deadlocked

# Дампы

- Можно сохранять дампы в файл и дебажить его позже на другой машине например, где есть исходники и символы отладки
- Умеет читать дампы файлов содержащие информацию про управляемый, неуправляемый код и смешанный код

Save Dump As...

## ▲ Dump Summary

Dump File  
Last Write Time  
Process Name  
Process Architecture  
Exception Code  
Exception Information  
Heap Information

## ▲ System Information

OS Version  
CLR Version(s)

## ▲ Modules

*Search*

Module Name
WindowsFormsApplication1.vshost.exe
ntdll.dll





# ВЕБ ПРИЛОЖЕНИЯ

# Сниппеты

- Сниппеты для веб приложения
- Сниппеты для:
  - JavaScript
  - HTML
  - ASP.NET

```
<script language="javascript">  
    function blah() {  
    }  
</script>
```

Insert Snippet:

- ASP.NET AJAX
- JScript
- XML Comments

```
        title="MSDN ASP.NET Docs">documentat  
</p>  
</asp:Content>
```

Insert Snippet:

- ASP.NET
- ASP.NET MVC 2
- HTML

# Сниппеты

```
<body>
  <form id="form1" runat="server">

  <asp:TextBox ID="Textbox1" runat="server" />
  req|
  objectdatasource
  ol
  P
  panel
  placeholder
  pre
  q
  radiobutton
  radiobuttonlist
  rangevalidator
  regularexpressionvalidator
  repeater
  requiredfieldvalidator
```

- Из менеджера СНИППЕТОВ

- На уровне IntelliSense в коде

```
<body>
  <form id="form1" runat="server">

  <asp:TextBox ID="Textbox1" runat="server" />
  <asp:RequiredFieldValidator ID="Requiredfieldvalidator1" ErrorMessage="errormessage"
  ControlToValidate="Textbox1" runat="server" />

  </form>
</body>
</html>
```

# Intellisense для JavaScript

jQuery расширен  
новой функцией  
detonate

```
jQuery.fn.extend({  
  detonate: function(confirm) {  
    /// <summary>Are you sure? This cannot be undone!</summary>  
    /// <param name="confirm" type="boolean">Are you really sure!?!</param>  
    /// <return type="boolean">true if confirmed, false otherwise</return>  
    $(this)  
      .effect("confirm", {confirm: confirm})  
      .hide();  
  }  
});  
  
// jQuery Document  
$(function() {  
  $("#target")  
    .click(function() {  
      $(this).detonate();  
    });  
});
```

Intellisense доступен  
для функции  
detonate автоматически

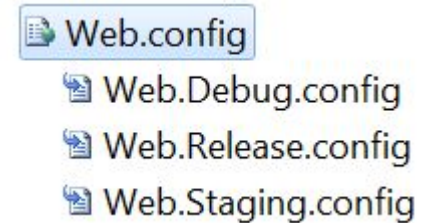
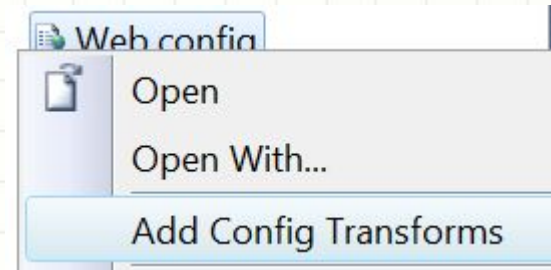
Комментарии  
тоже попадают в  
Intellisense

Are you sure? This cannot be undone!



# Web.config

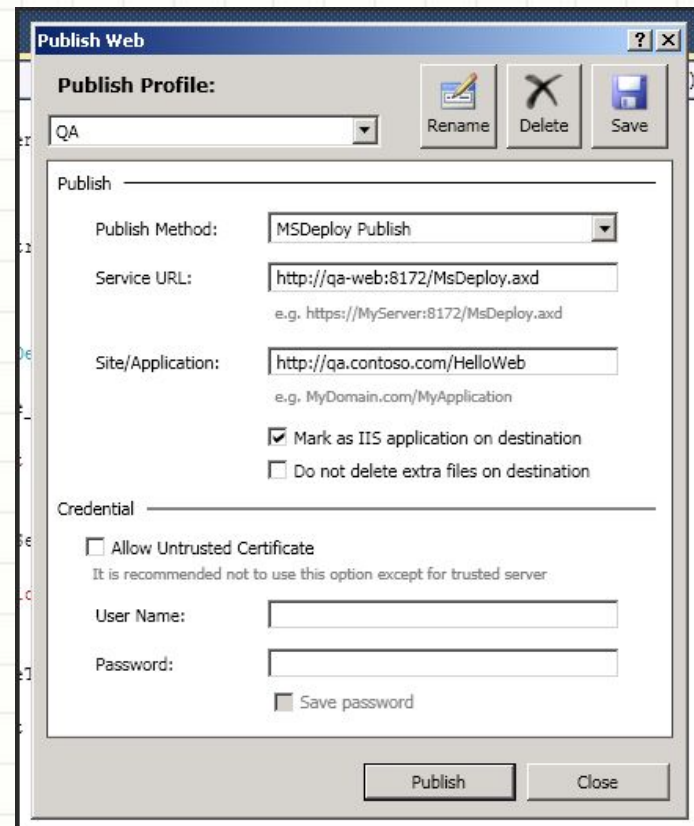
- Более чистый web.config файл приложения.
- Web.config transforms - web.config проекта меняется в зависимости от разных сред развертывания



```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application,
  see the following WebResourceUrl link.
  http://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.0" />
  </system.web>
</configuration>
```

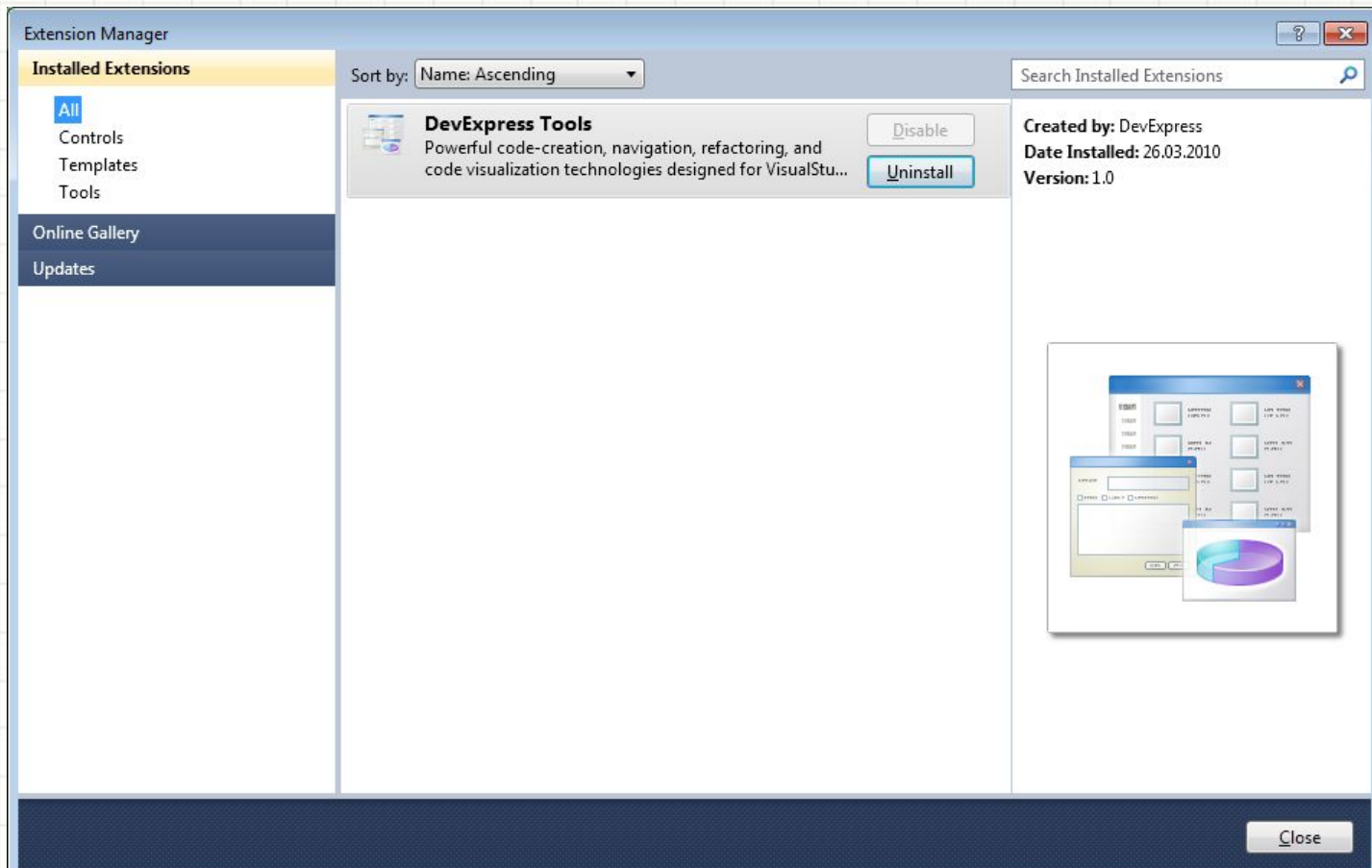
# One-Click Web Deployment

- MSDeploy интегрирован в Visual Studio 2010
- После конфигурации профайла, развертывание в ОДИН КЛИК



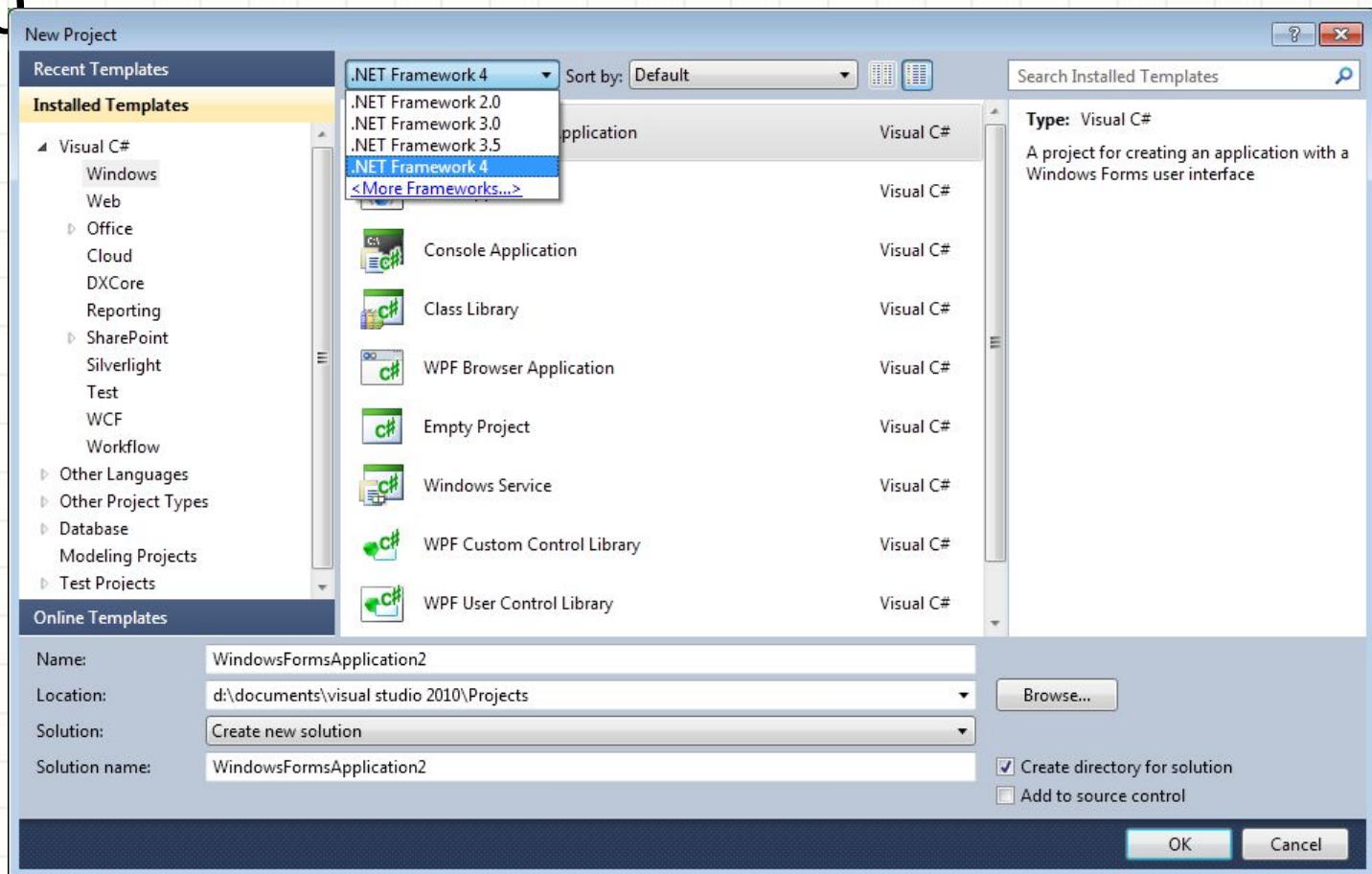
# Расширяемость

Включен новый Extension Manager для дополнений к Visual Studio 2010



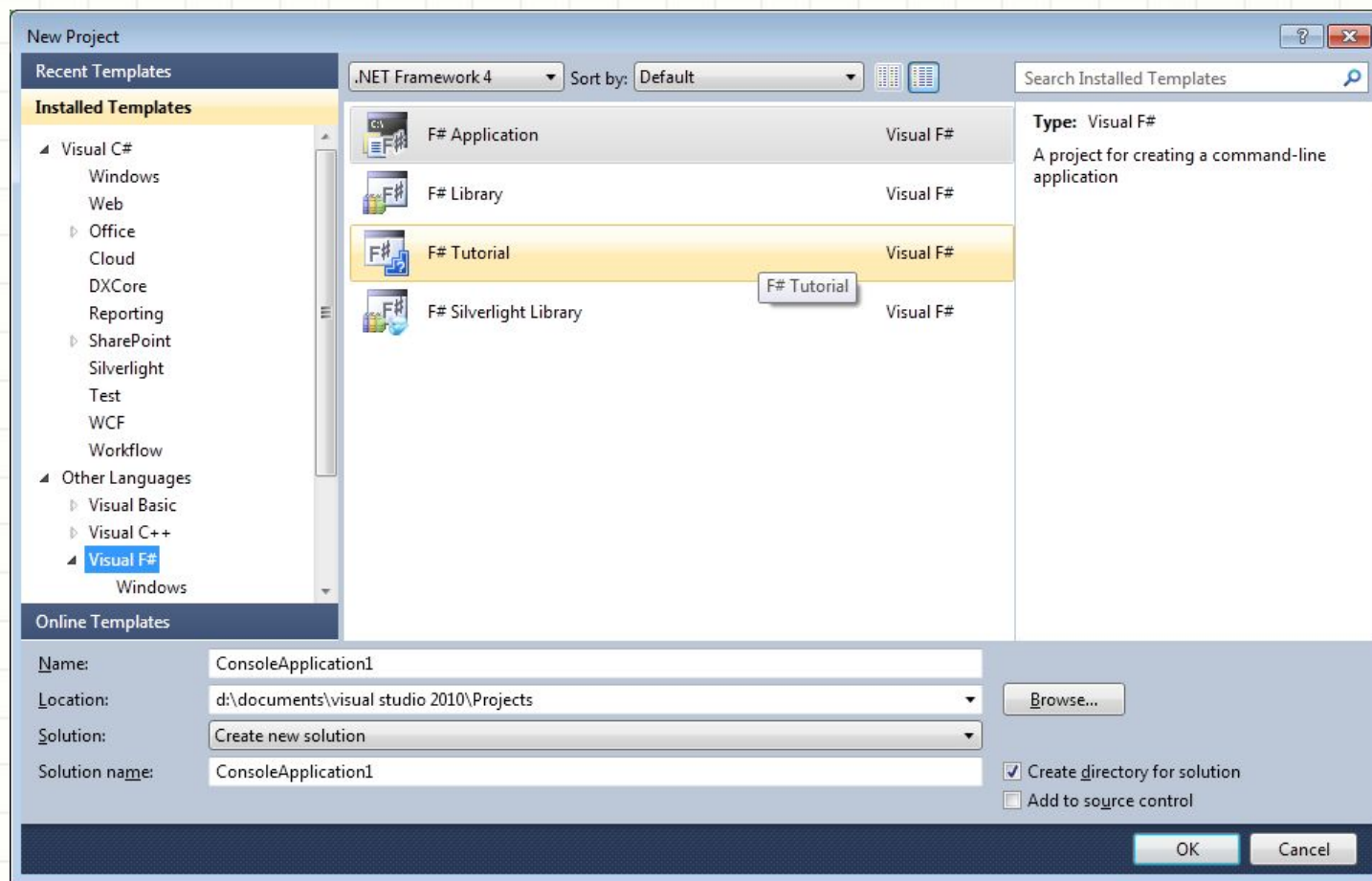
# Multitargeting

## Создание приложений от .net 2.0 до .net 4.0



# F#

## Новый язык программирования в Visual Studio 2010





# ССЫЛКИ

What's New in Visual Studio 2010

[http://msdn.microsoft.com/en-us/library/bb386063\(VS.100\).aspx](http://msdn.microsoft.com/en-us/library/bb386063(VS.100).aspx)

Visual Studio on MSDN

<http://msdn.microsoft.com/vstudio>

Scott Guthrie blog

<http://weblogs.asp.net/scottgu/>



# ЧТО НОВОГО В .NET 4.0

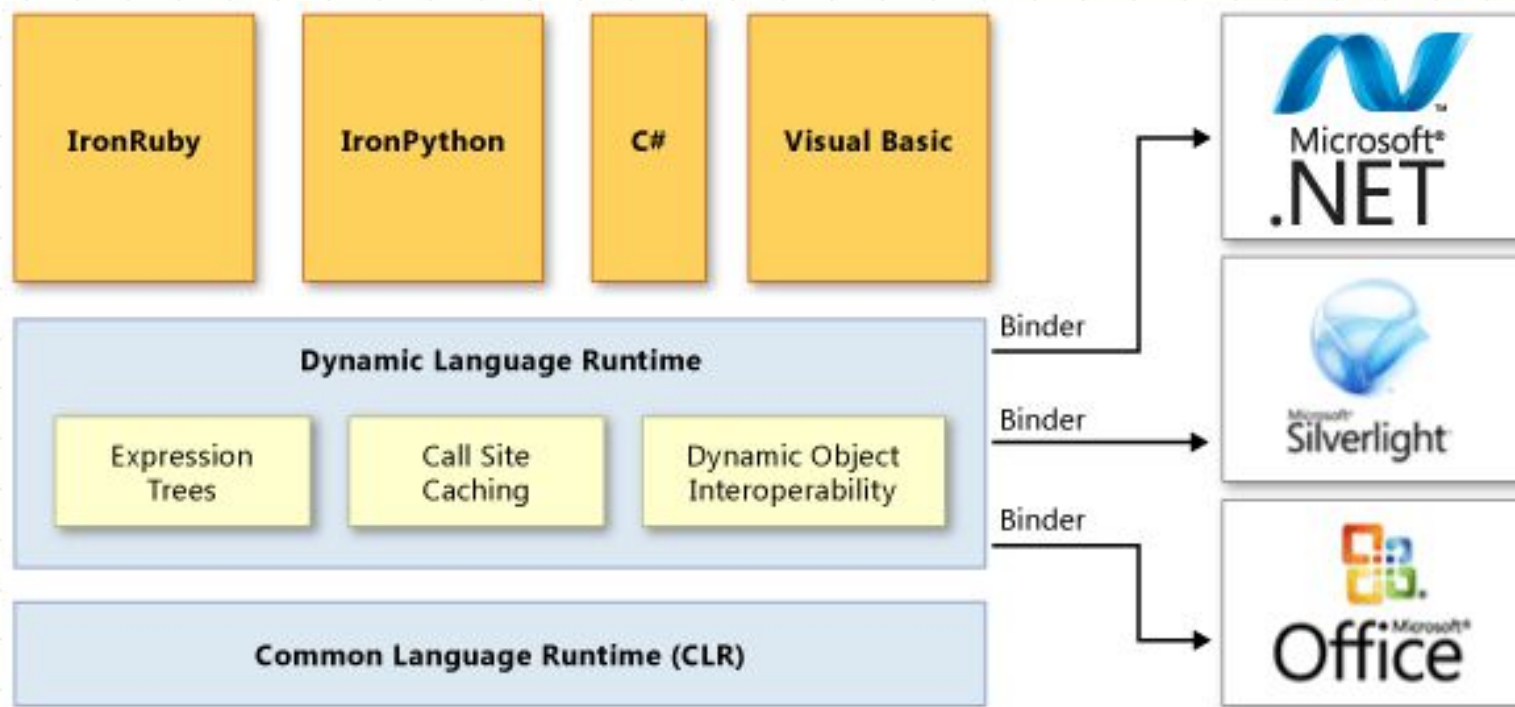
# Dynamic ТИП

Операции которые содержат выражения типа `dynamic` не проверяются компилятором. Компилятор упаковывает информацию об операции и эта информация используется для выполнения операций в `run time`.

```
dynamic int1 = 1;  
dynamic ex1 = new Exception("Oops!");  
dynamic result = int1 + ex1;
```

# DLR

- DLR
  - Новая среда выполнения для поддержки динамических типов и динамических



# DLR

- DLR набор сервисов для CLR для поддержки динамических типов:
  - *Expression Trees*. Используются для представления семантики языка
  - *Call site caching*. Кэширование операций, сбор информации необходимой для выполнения операций
  - *Dynamic object interoperability*.  
*IDynamicMetaObjectProvider*, *DynamicMetaObject*, *DynamicObject* и *ExpandoObject*



# DLR

- Основное применение – интероп и рефлексия.
- Пример – DynamicXML

```
<nodes>  
  <firstnode>  
    <samplenode value="sample value"></samplenode>  
  </firstnode>  
</nodes>
```

```
dynamic dynamicXml = new DynamicXml(document);  
string value = dynamicXml.nodes.firstnode.samplenode.value;
```

# Именованные и необязательные параметры в методах

```
static void DoTask(string taskName = "sample task",  
    int repeatCount = 1, object yetAnotherArg = null)  
{  
    Console.WriteLine("a1 = {0}, a2 = {1}, a3 = {2}",  
        taskName, repeatCount, yetAnotherArg);  
}
```

```
DoTask("name", 1, 15);  
DoTask(taskName: "new task", yetAnotherArg: 15);  
DoTask();
```

# Возможности для COM Interop

- Теперь можно COM объекты определять как динамические и не приводить постоянно получаемые объекты к определенным типам для вызова методов или свойств.

```
excel.Cells[1, 1].Value = "Hello";  
// ВМЕСТО  
((Excel.Range)excel.Cells[1, 1]).Value2 = "Hello";
```

# Ко- и контравариантность generic

- Приведение generic
  - Теперь generic типы можно приводить к базовому и к наследнику (контра- и ковариантность, структуры - инвариантны)

```
IEnumerable<Derived> d = new List<Derived>();  
IEnumerable<Base> b = d;
```

- In, Out в generic для обозначения контра- и ковариантности

```
Func<Object, ArgumentException> func1 = null;  
// явное приведение не нужно  
Func<String, Exception> func2 = func1;
```

# Типы BigInteger и Complex

- BigInteger
  - Неизменяемый тип который представляет большое целое число чье значение теоретически не имеет пределов
- Complex
  - Тип для представления комплексных чисел.
- SortedSet<T>
  - Представляет самобалансирующееся дерево которое поддерживает данные в сортированном виде после вставок, удалений и поиска элементов



# Новое в VB.NET

- Auto-Implemented Properties
  - Упрощенный синтаксис для объявления свойств

```
Public Property Owner As String = "DefaultName"
```

- Инициализация коллекций
  - Упрощенный синтаксис для объявления и наполнения коллекций значениями

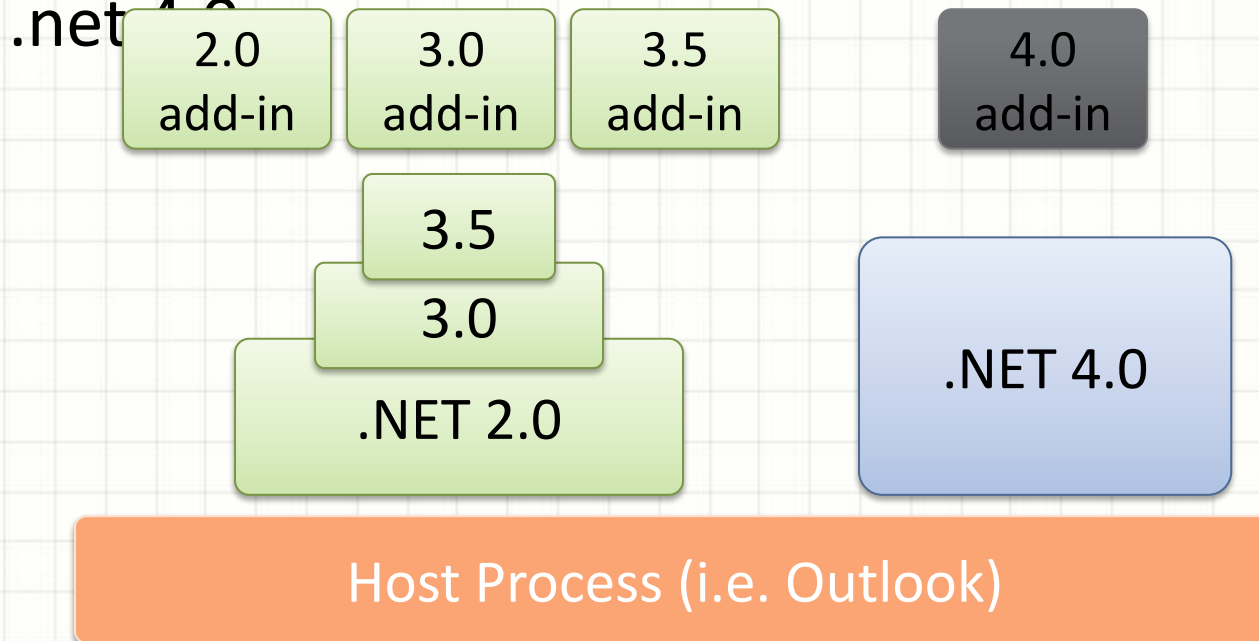
```
Public Property Items As New List(Of String) = {"M", "T", "W"}
```

- Implicit Line Continuation

Для продолжения конструкции на следующей строке нет необходимости использовать подчеркивание

# In-Process Side-by-Side Execution

- Позволяет загружать и стартовать несколько версий .net в одном процессе
  - Например приложение для которого написаны плагины как на .net 2.0 так и на .net 4.0



# Code contracts

- Новый способ задавать контракты к методу или типу
- Сценарии использования контрактов:
  - Статическое нахождение багов.  
Программирование в стиле *defensive programming*
  - Создание утверждений для автоматизированных утилит тестирования для улучшения покрытия кода тестами
  - Играет роль документации к коду

# Code contracts

- Старый вариант

```
if (item == null)
    throw new ArgumentNullException("item", "item is
null.");
```

- Новый вариант

```
Contract.Requires<NullReferenceException>(item != null);
Contract.Requires(item.Price >= 0);
```

# Managed Extensibility Framework

MEF – это механизм, который позволяет минимумом кода внедрить в проекты поддержку расширяемости (например, плагинов)



# Parallel Computing. Новые типы для синхронизации

IProducerConsumerCollection<T>  
ConcurrentQueue<T>  
ConcurrentStack<T>  
ConcurrentBag<T>  
ConcurrentDictionary<TKey,TValue>

## Phases and work exchange

Barrier  
BlockingCollection<T>  
CountdownEvent

## Partitioning

{Orderable}Partitioner<T>  
Partitioner.Create

## Exception handling

AggregateException

## Initialization

Lazy<T>  
LazyInitializer.EnsureInitialized<T>  
ThreadLocal<T>

## Locks

ManualResetEventSlim  
SemaphoreSlim  
SpinLock

# Parallel Computing. Parallelize For.

Управление поток – основная работа

```
for (int i = 0; i < n; i++)  
{  
    work(i);  
}
```

```
foreach(var item in data)  
{  
    work(item);  
}
```

```
StatementA();  
StatementB();  
StatementC();
```

Можно распаралелить если итерации независимы друг от друга

```
Parallel.For(0, n, i=>  
{  
    work(i);  
});
```

```
Parallel.ForEach(data, item=>  
{  
    work(item);  
});
```

```
Parallel.Invoke(  
    () => StatementA(),  
    () => StatementB(),  
    () => StatementC());
```

Synchronous

All work quiesces, regularly or exceptionally

Lots of knobs

Cancellation, breaking, task-local state, custom partitioning, scheduling, degree of parallelism

# PLINQ

- Реализует полный набор стандартных LINQ операций
- Реализовано на экстеншн методы к IEnumerable
- Дополнительные операторы для параллельных операций

```
from n in names.AsParallel().WithDegreeOfParallelism(ProcessorsToUse.Value)
    where n.Name.Equals(queryInfo.Name,
StringComparison.InvariantCultureIgnoreCase) &&
        n.State == queryInfo.State &&
        n.Year >= yearStart && n.Year <= yearEnd
    orderby n.Year ascending
select n;
```

# Task Parallel Library (TPL)

- Набор типов и APIs
- System.Threading
- System.Threading.Tasks

```
// Последовательно  
foreach (var item in sourceCollection)  
{  
    Process(item);  
}
```

```
// Параллельно  
Parallel.ForEach (sourceCollection, item => Process(item));
```


# I/O

- **Файлы спроектированные в память (Memory-Mapped File)**
  - Используются для редактирования очень больших файлов и создания разделяемой памяти для межпроцессного взаимодействия
- **Stream.CopyTo**
  - Позволяет копировать содержимое одного потока в другой



# Сборка мусора

- В .net 4.0 появилась фоновая сборка мусора (background garbage collection) на смену concurrent garbage collection
  - Обеспечивает улучшенную производительность



**Спасибо за  
внимание:)**