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)

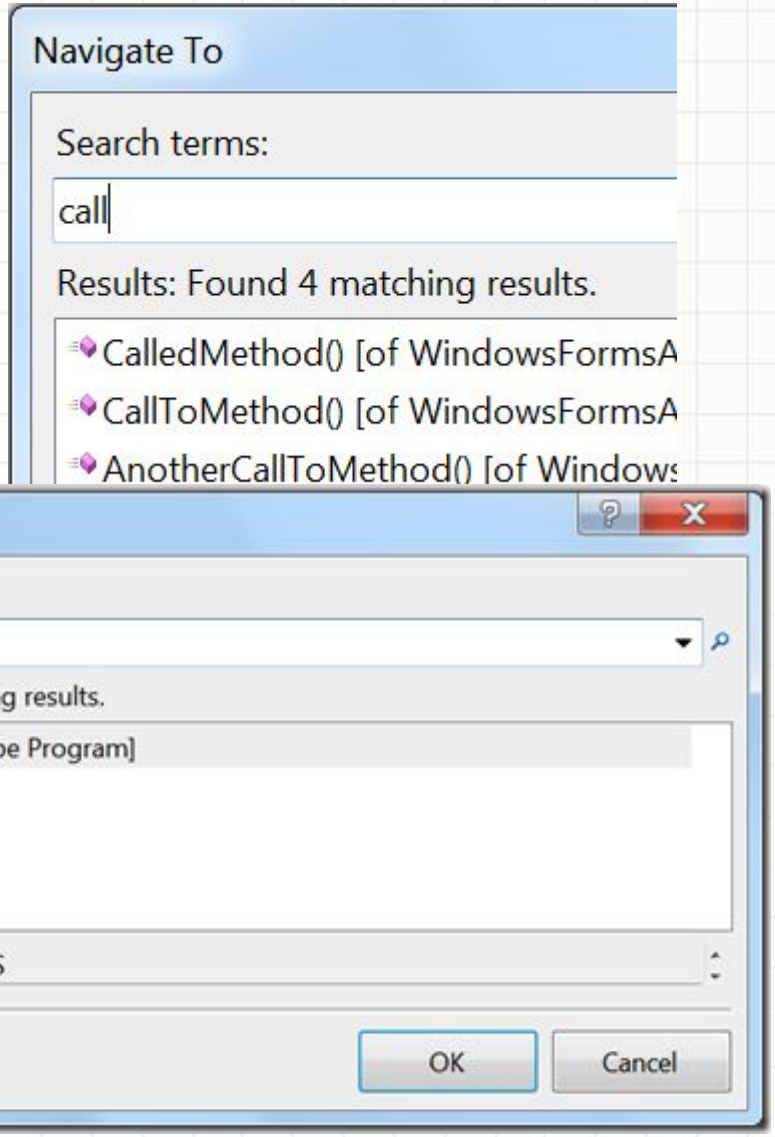
- Автоматическая подсветка символов

- 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();
}
```

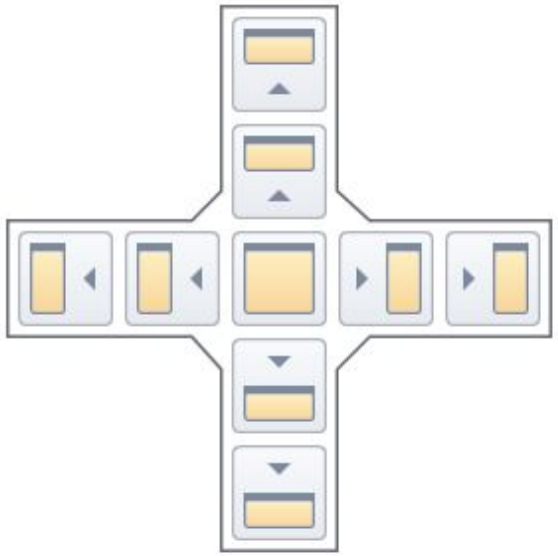
Navigate To

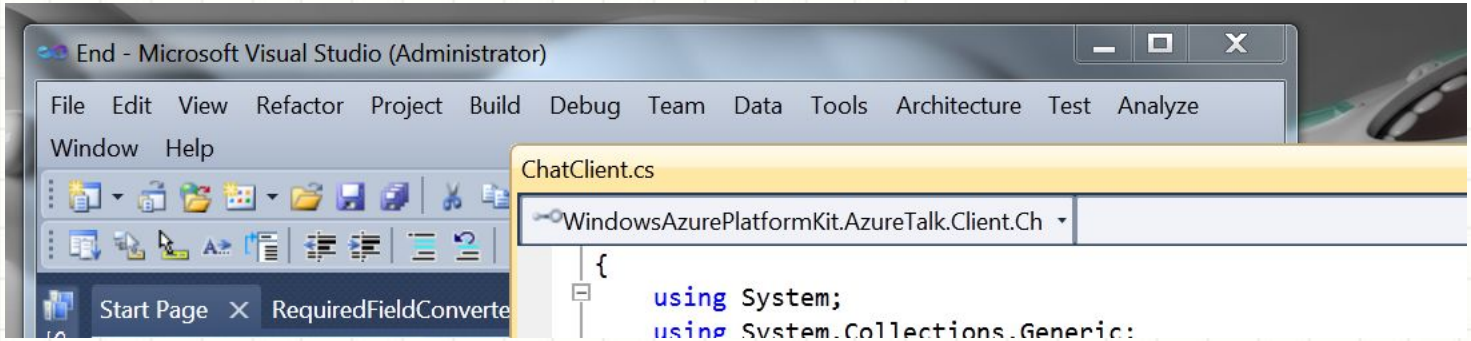
- CTRL + ,
- Поиск символов по имени



Docking

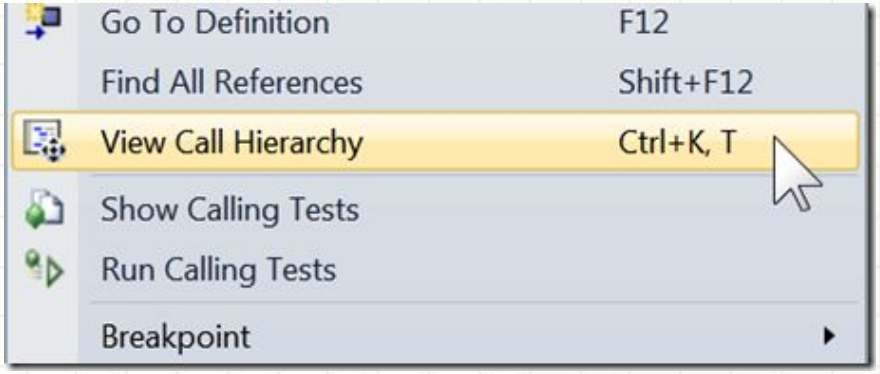
- Новый docking diamond
- Окна докируются где угодно
- Можно вынести окно кода за IDE

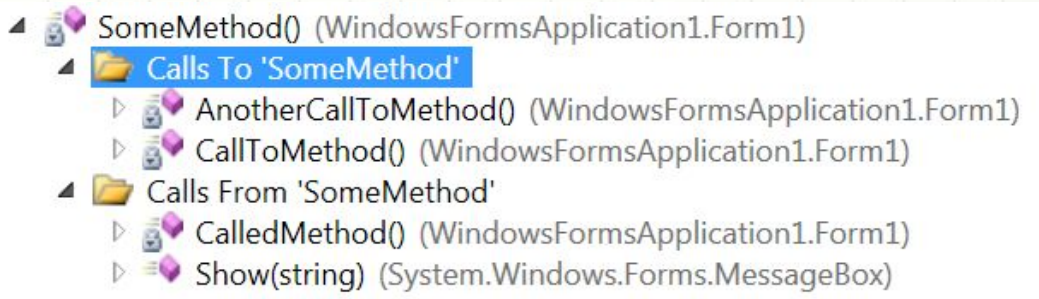




Call Hierarchy (C# only)

- CTRL + K, T
- Просмотр вызовов к/из метода
- Удобный способ отслеживать вызовы в design time



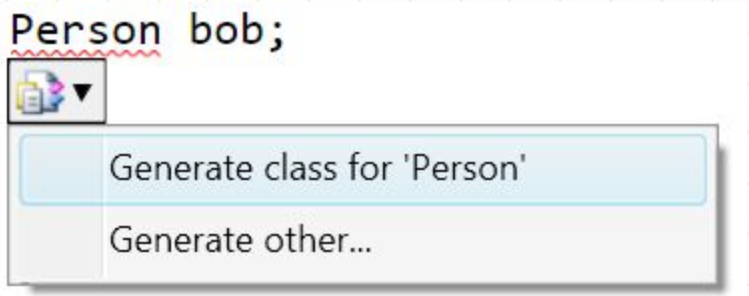


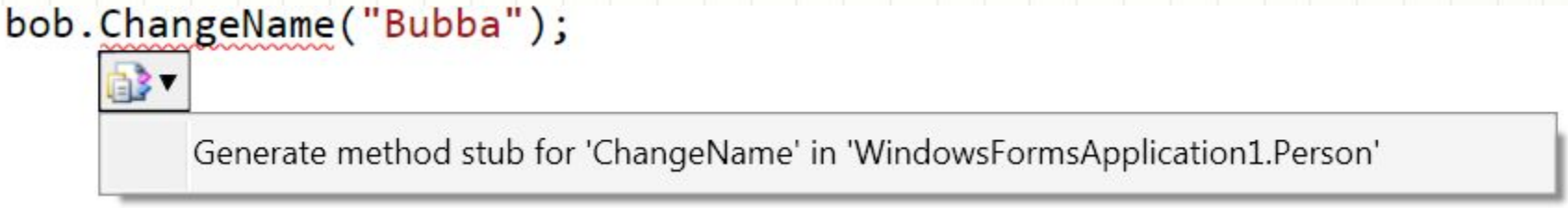
Zoom

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

Generate from Usage

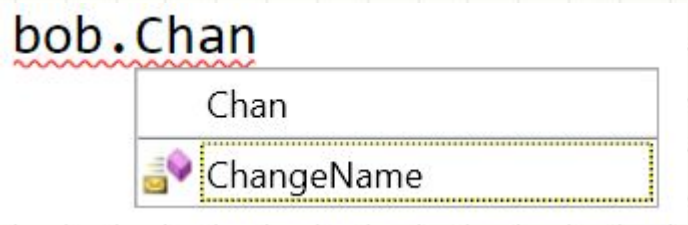
- Используется для автоматической генерации кода, например заглушек
- Использование классов и их членов до их полного определения
- TDD стиль написания кода





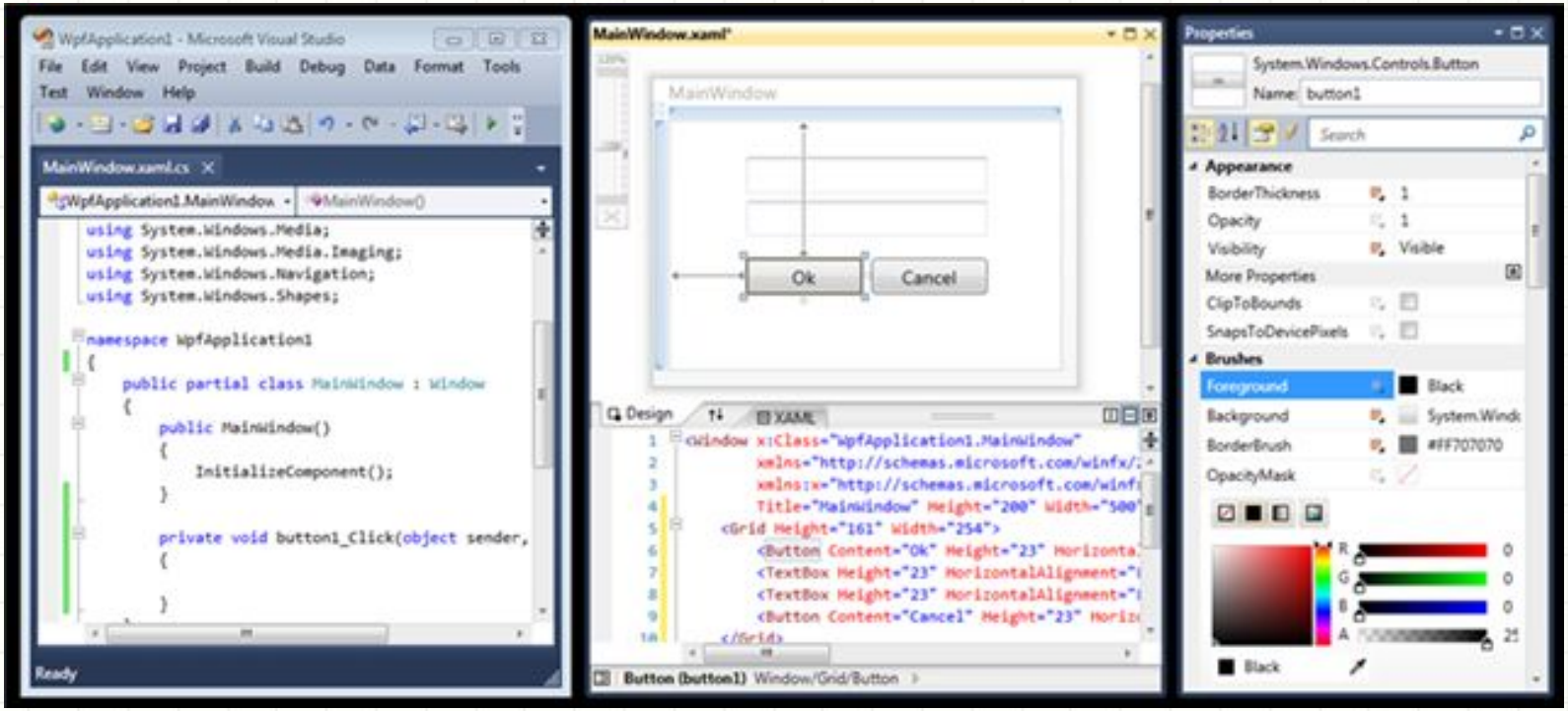
Intellisense Suggestion Mode

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



Поддержка нескольких мониторов

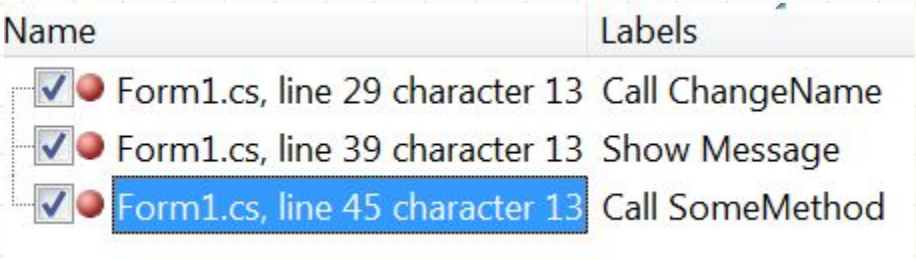
- Все окна теперь можно сделать плавающими
- Даже Code Editor и Design View



УЛУЧШЕНИЯ ОТЛАДКИ

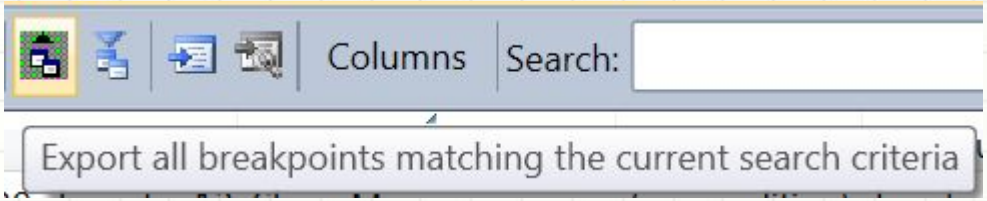
Breakpoints

- Можно добавлять метки к точкам останова

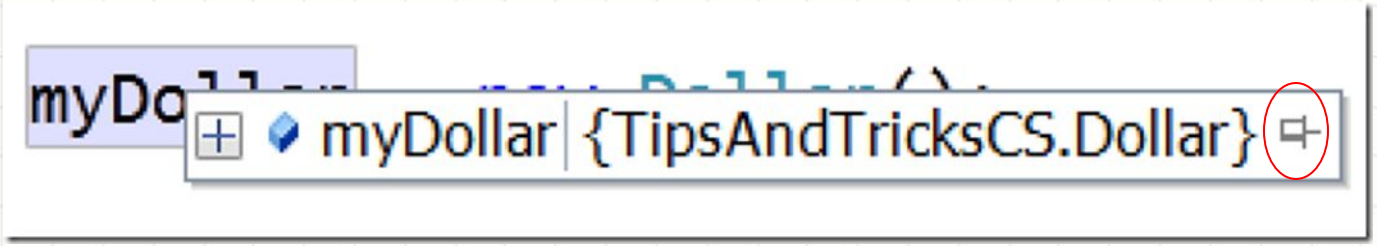


- Можно искать по точкам останова

- Импорт/ Экспорт точек



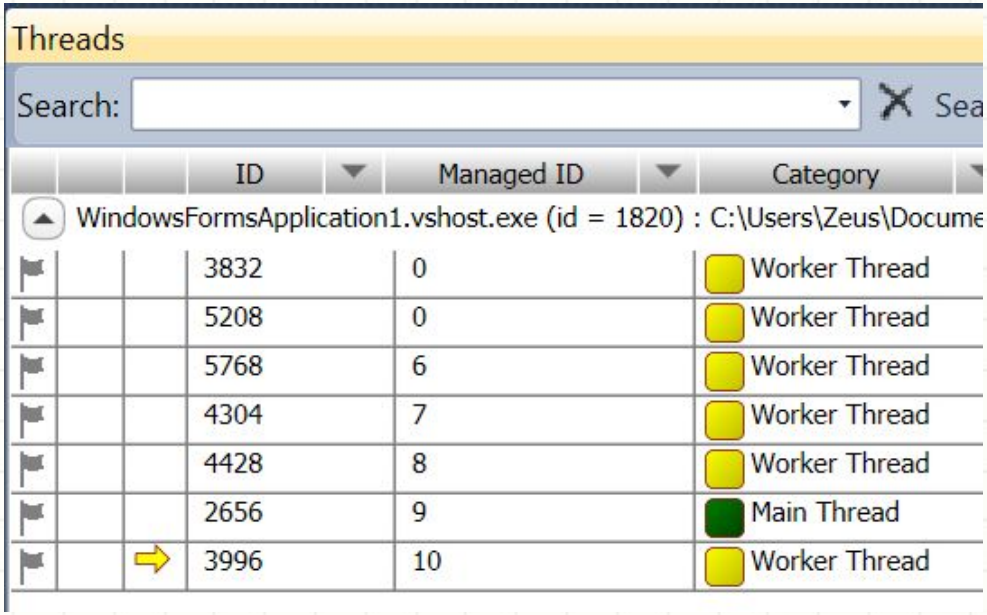
Плавающие подсказки (Data Tips)



Call Hierarchy (C# only)

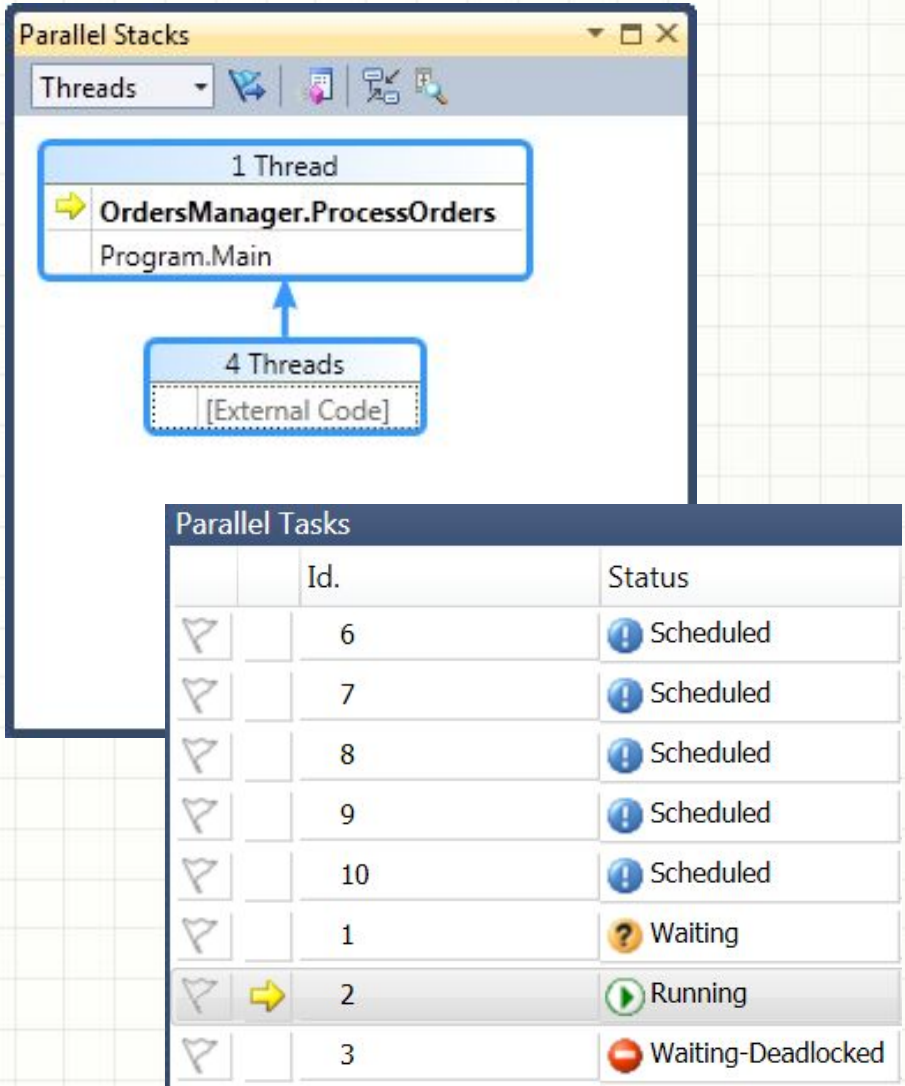
Окно потоков

- Полностью redesigned
- Фильтрация, поиск по call-stack, сворачивание, группировки
- Новые колонки:
  - Affinity masks
  - Process names
  - Managed IDs



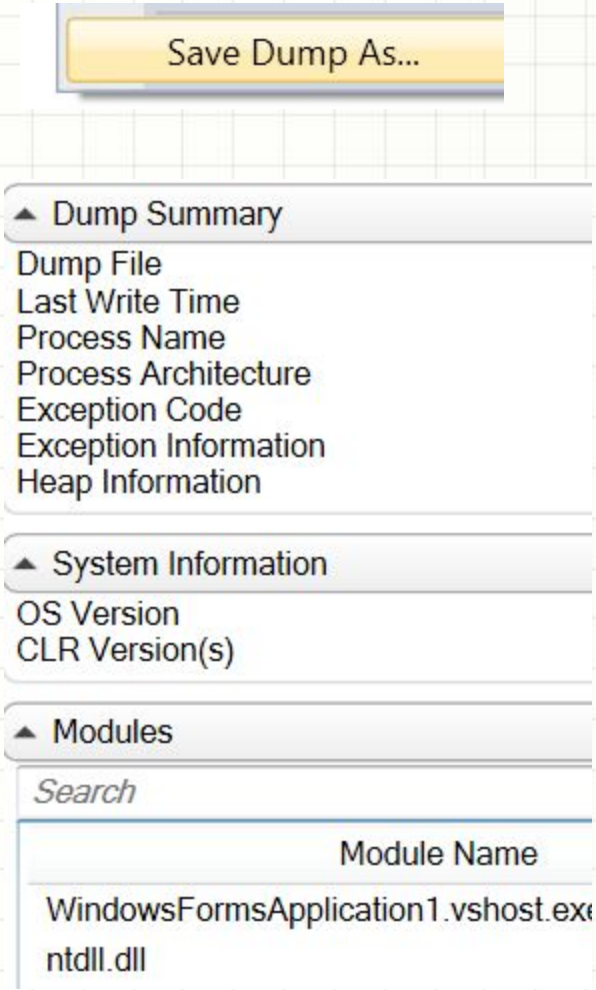
Окна Parallel Stacks, Parallel Tasks

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



Дампы

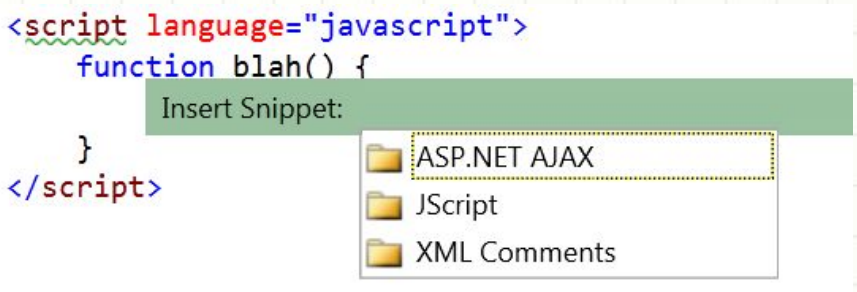
- Можно сохранять дамп файл и дебажить его позже на другой машине например, где есть исходники и симоволы отладки
- Умеет читать дамп файлы содержащие информацтю про управляемый, неуправляемый код и смешаный код



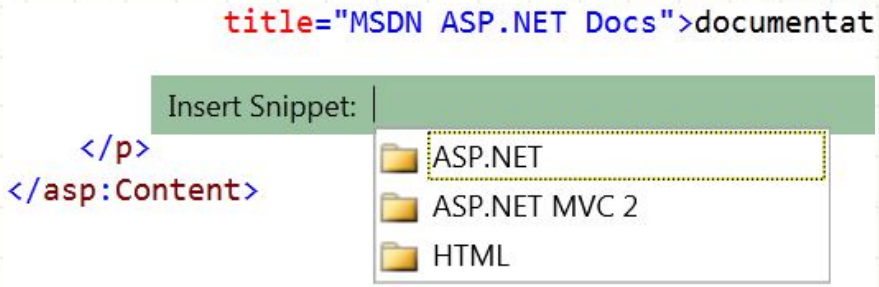
ВЕБ ПРИЛОЖЕНИЯ

Сниппеты

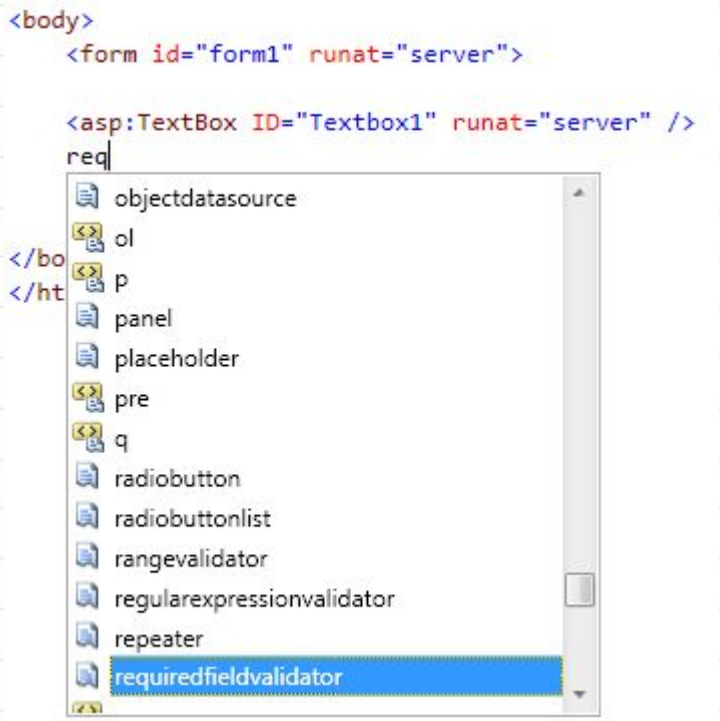
- Сниппеты для веб рприложения



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



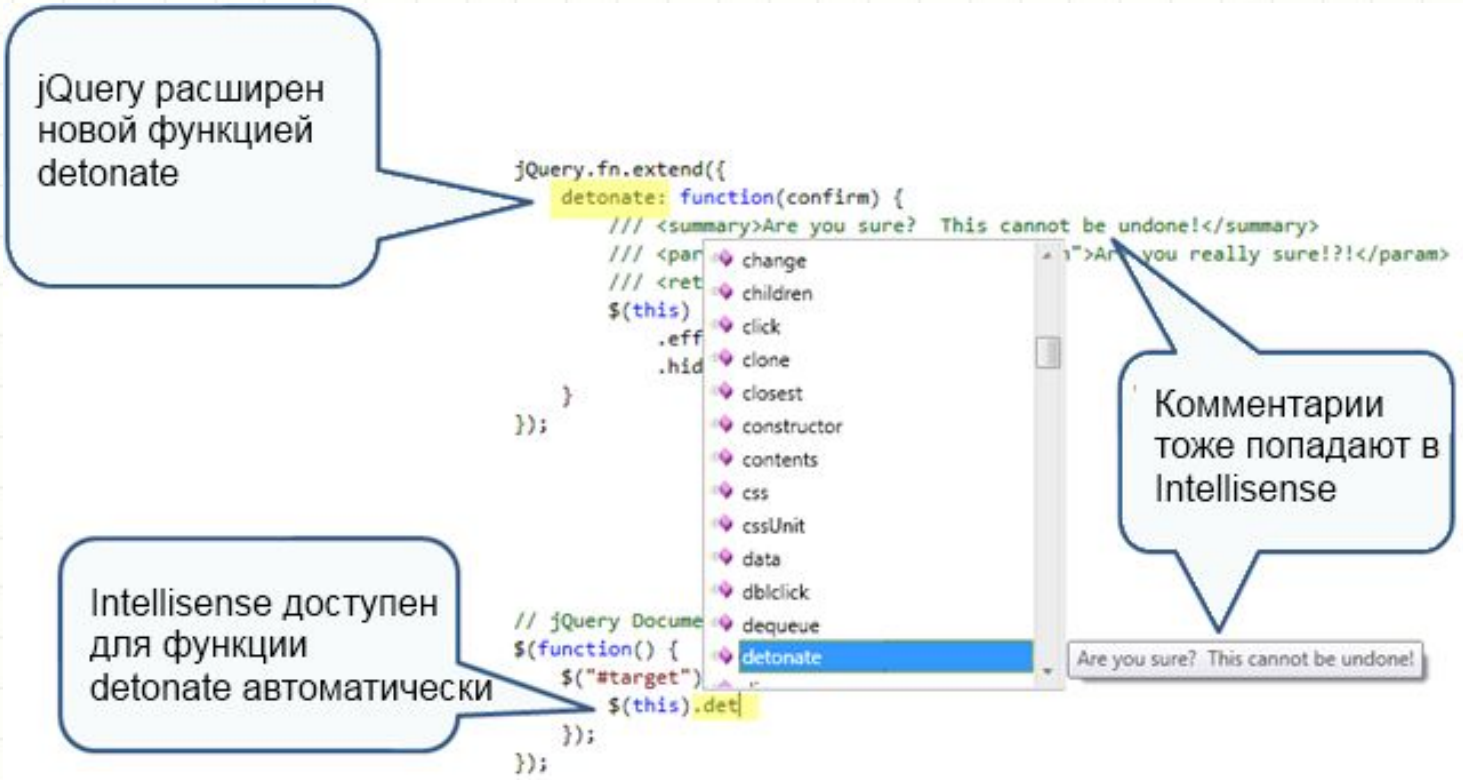
Сниппеты



- Из менеджера сниппетов
- На уровне IntelliSense в коде

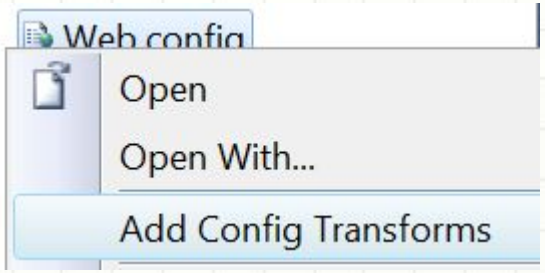
```
<body>
    <form id="form1" runat="server">

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

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

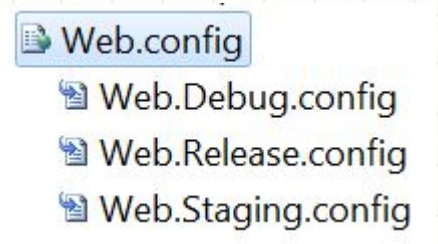
Intellisense для JavaScript



Web.config

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

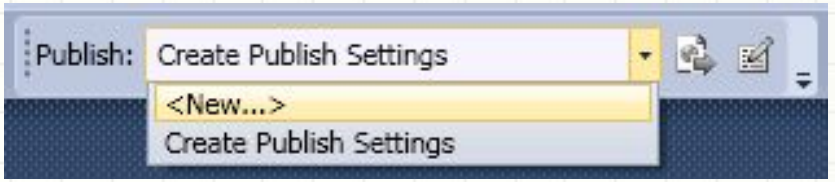


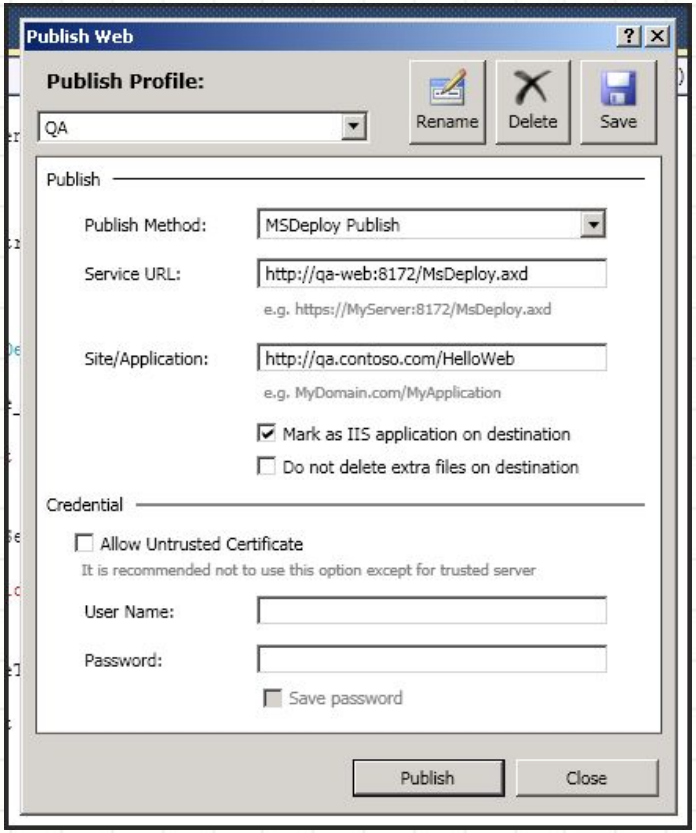



```
<?xml version="1.0"?>

<!--
  For more information on how to configure your ASP.NET appl
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->

<configuration>
    <system.web>
        <compilation debug="true" targetFramework="4.0" />
    </system.web>

</configuration>
```

One-Click Web Deployment

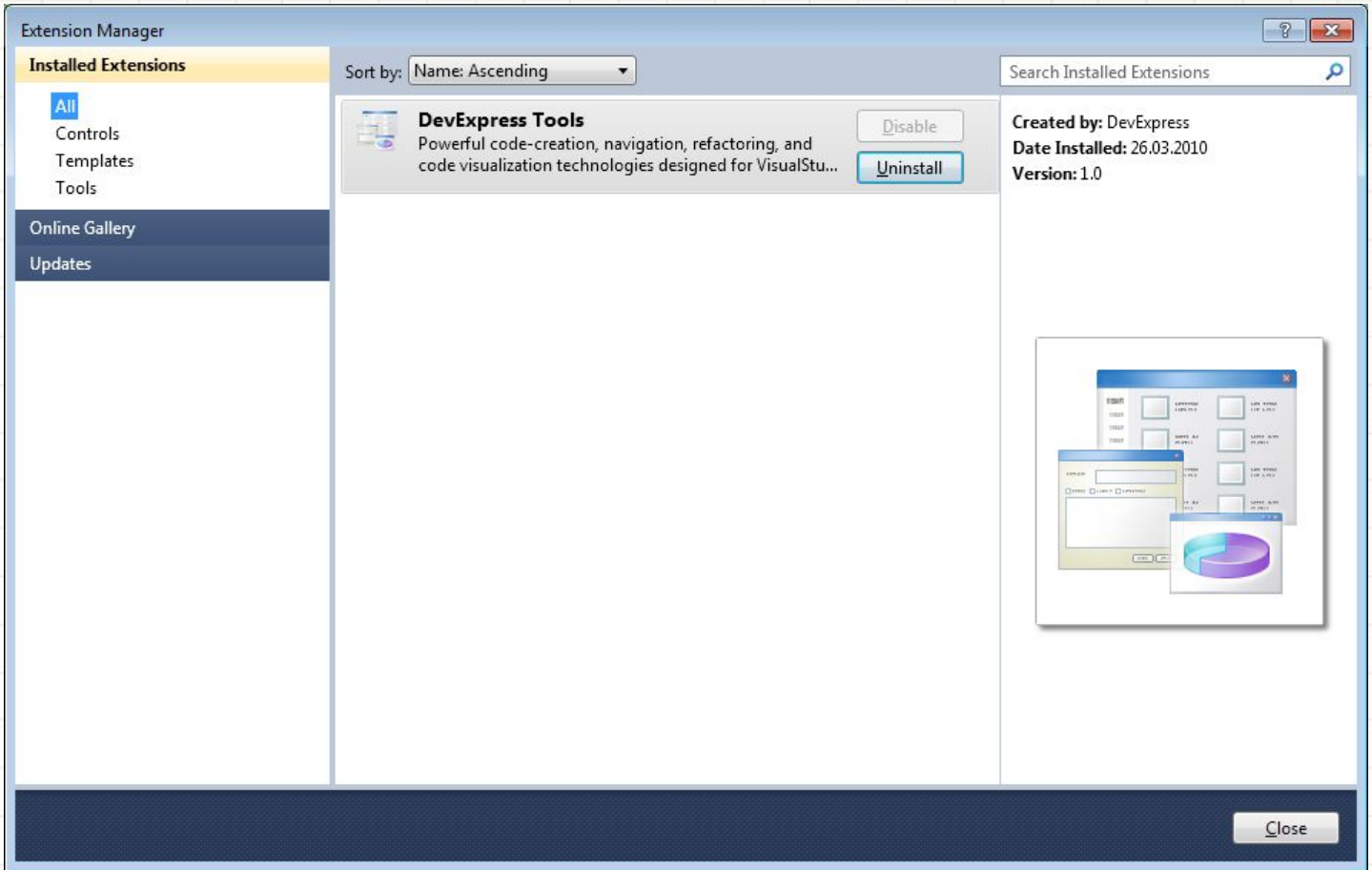
- MSDeploy интегрирован в Visual Studio 2010
- После конфигурации профайла, развертывание в один клик





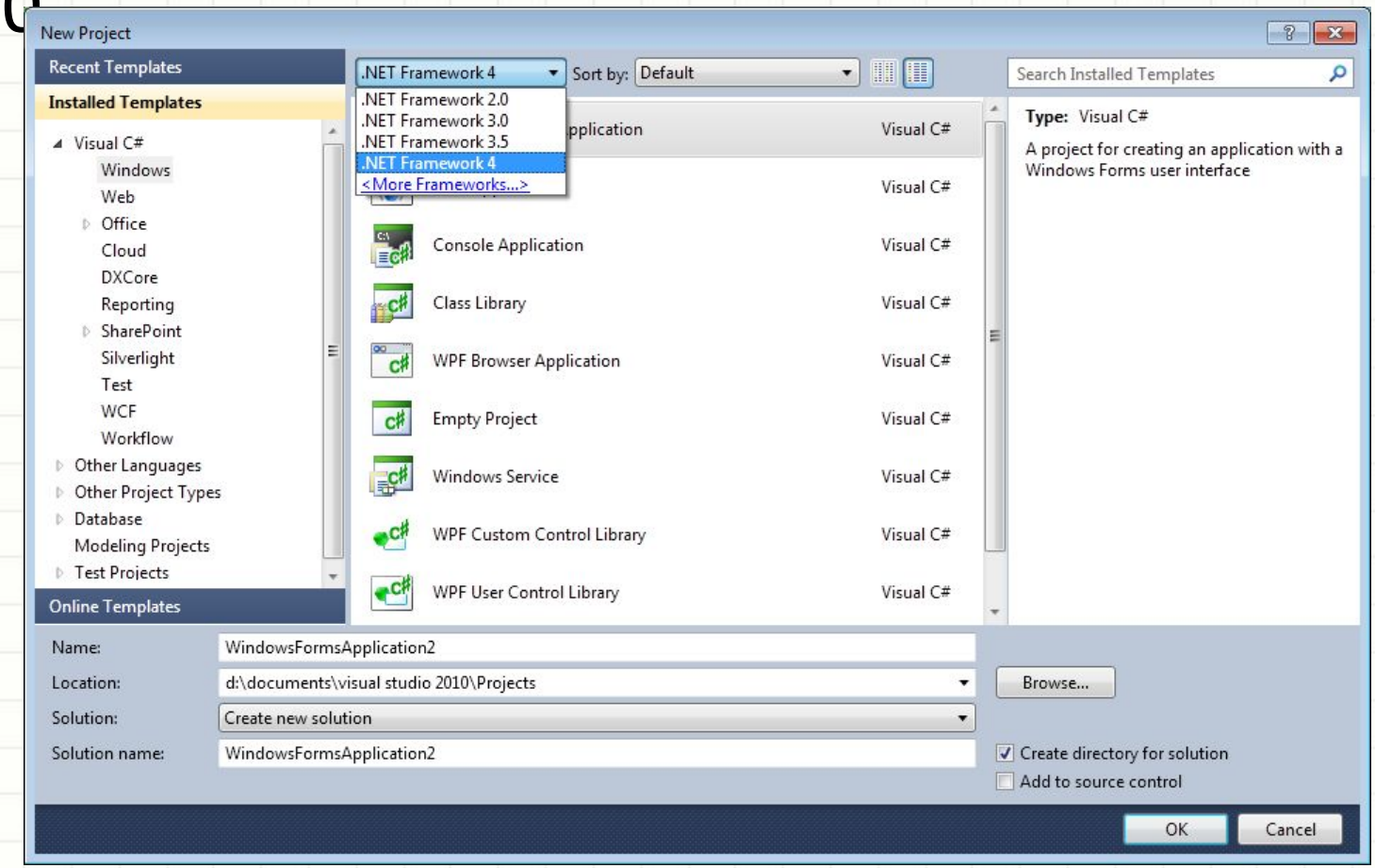
Расширяемость

Включен новый Extenion Manager для дополнений к Visual Studio 2010



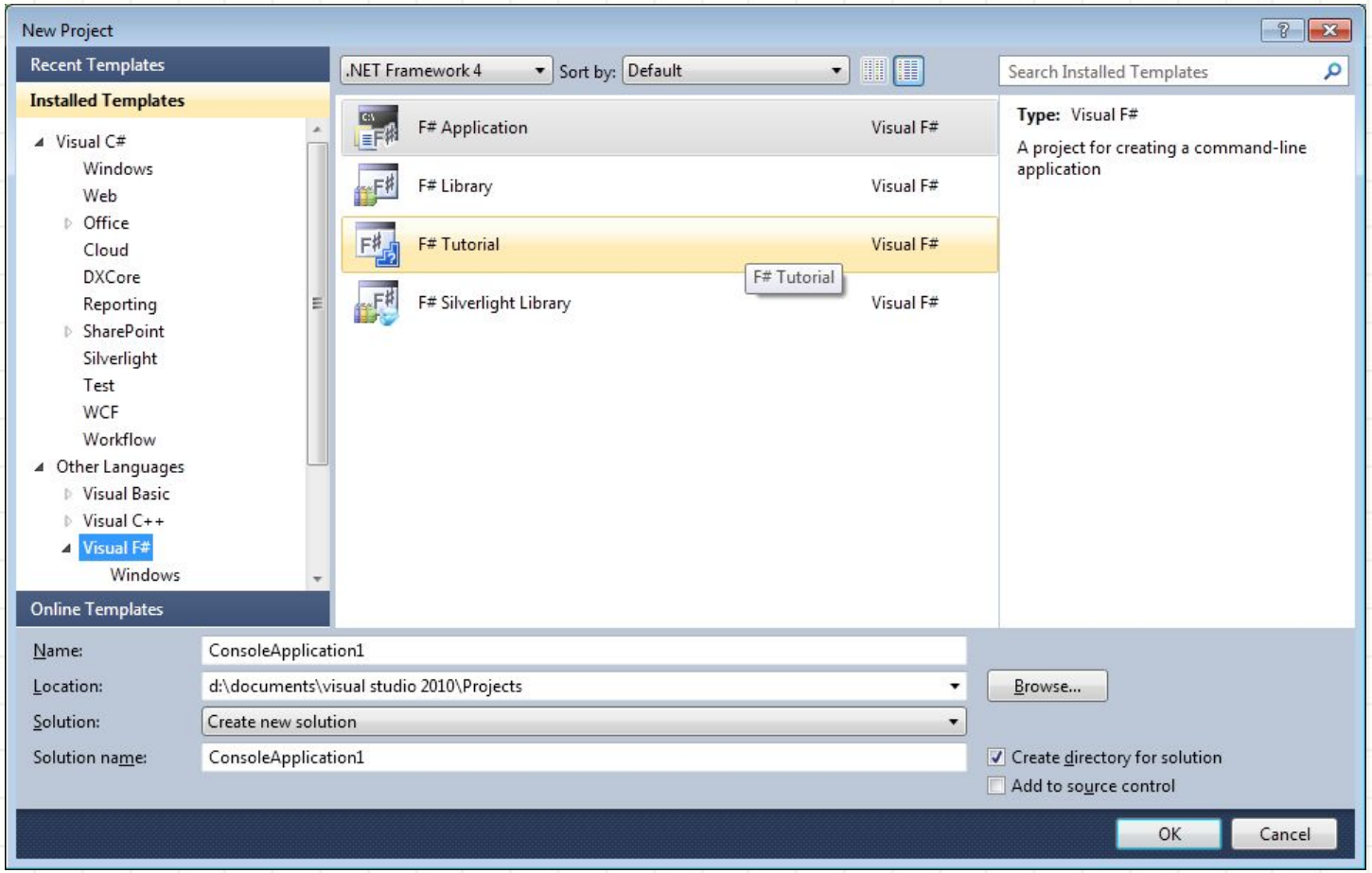
Multitargeting

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

Visual Studio on MSDN
http://msdn.microsoft.com/vstudio

Scott Guthrie blog
http://weblogs.asp.net/scottgu/

ЧТО НОВОГО В .NET 4.0

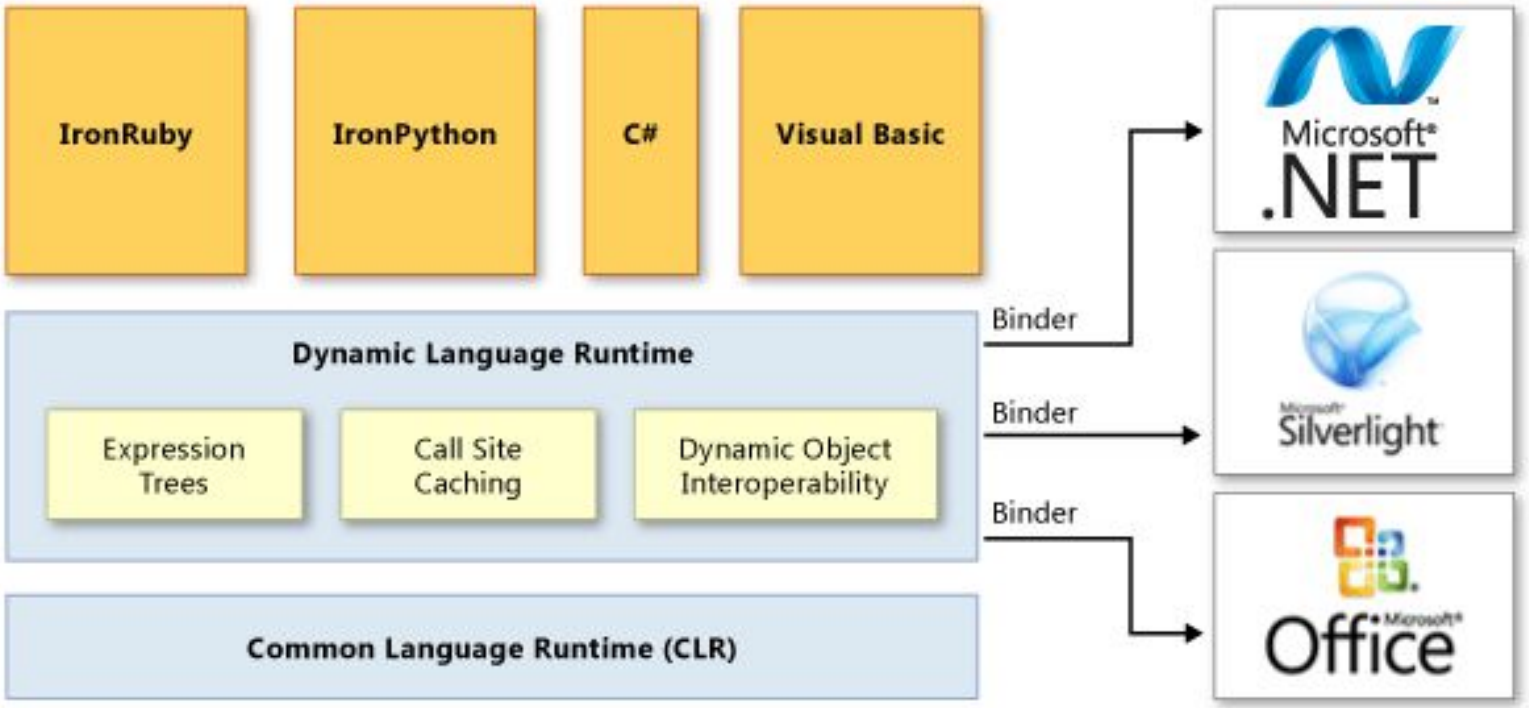
Dynamic тип

Операции которые содержат выражения типа dynamic are 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. Кеширование 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 котнра- и ковариантности

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

Типы BigInterger и Complex

- BigInteger
  - Неизменяемый тип которые 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 сортированом виде после вставок, удалений и поиска элементов

Новое в VB.NET

- Auto-Implemented Properties
  - Упрощенный синтаксис для обьявления свойств

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

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

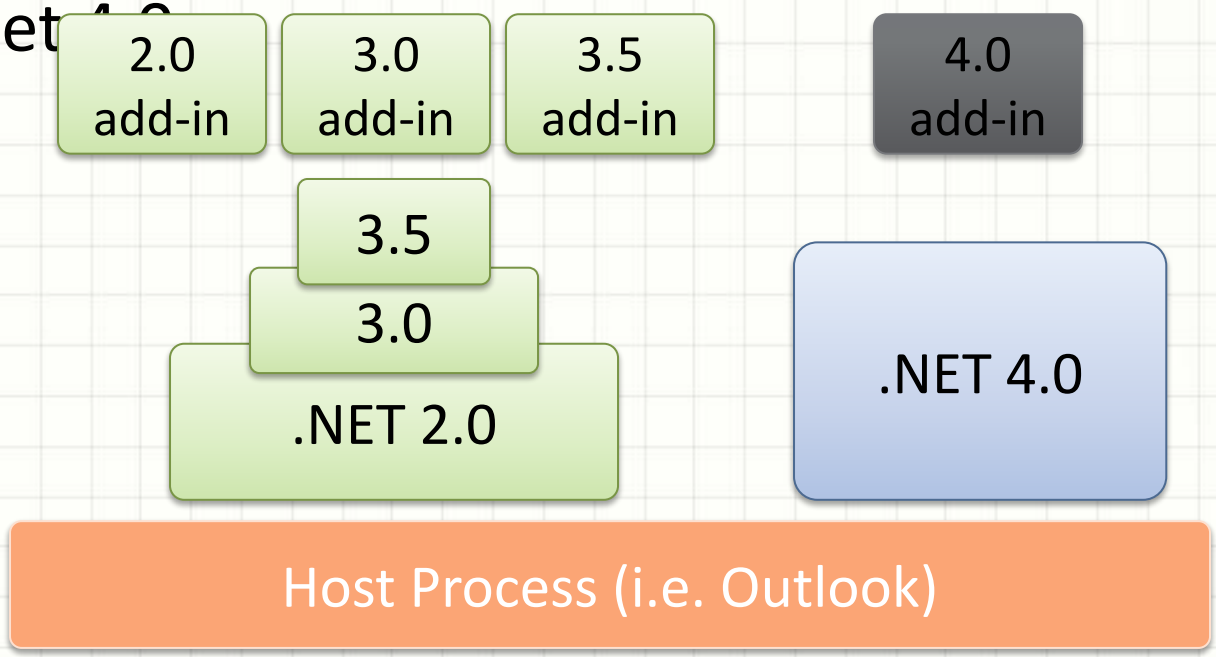
```
Public Property Items As New List(Of String) = {“M“, “T“, “W“}
```

- Implicit Line Continuation

  Для продолжения конструкции на следующей строчке 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 defenisve programming
  - Создание утверждений для автоматизированых 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 Extesibility 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

ManyCore: http://microoftpdc.com/Sessions/P09-09 (50:00)

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 независимі 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

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

ManyCore: http://microsoftpdc.com/Sessions/P09-09

PLINQ

- Реализует полный набор стандартных LINQ операций
- Реализовано на екстеншн методы к IParallelEnumerable
- Дополнительные операторы для паралельных 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);
}

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

I/O

- Файлы спроектированые в память

(Memory-Mapped File)

  - Используются для редактирования очень большых 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:)