

C# Professional

Async & Await

C# Professional

После урока обязательно



Повторите этот урок в видео формате на
[ITVDN.com](http://itvdn.com)



Проверьте как Вы усвоили данный материал на
TestProvider.com

Async & Await

C# Professional

C# 5.0 – What's new?

C# 2 = Generics, Lambda

C# 3 = var, LINQ

C# 4 = dynamic, TPL

C# 5 = Async

C# Professional

Синхронность: проблемы

```
private void GetButtonClick(object sender, RoutedEventArgs e)
{
    var req= (HttpWebRequest) WebRequest.Create("http://microsoft.com/");

    .....

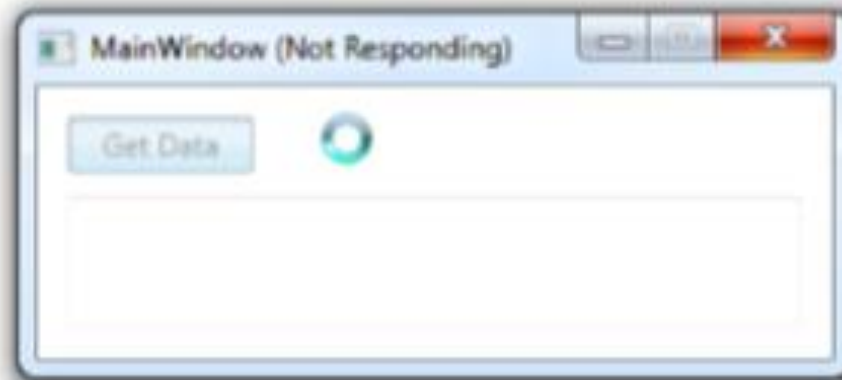
    ➡ var resp= (HttpWebResponse) req.GetResponse();

    dataTextBox.Text+= resp.Headers.ToString();
}
```

C# Professional

Синхронность: проблемы

Вызывающий поток блокируется, до завершения длительной операции



C# Professional

Асинхронная модель до C# 5

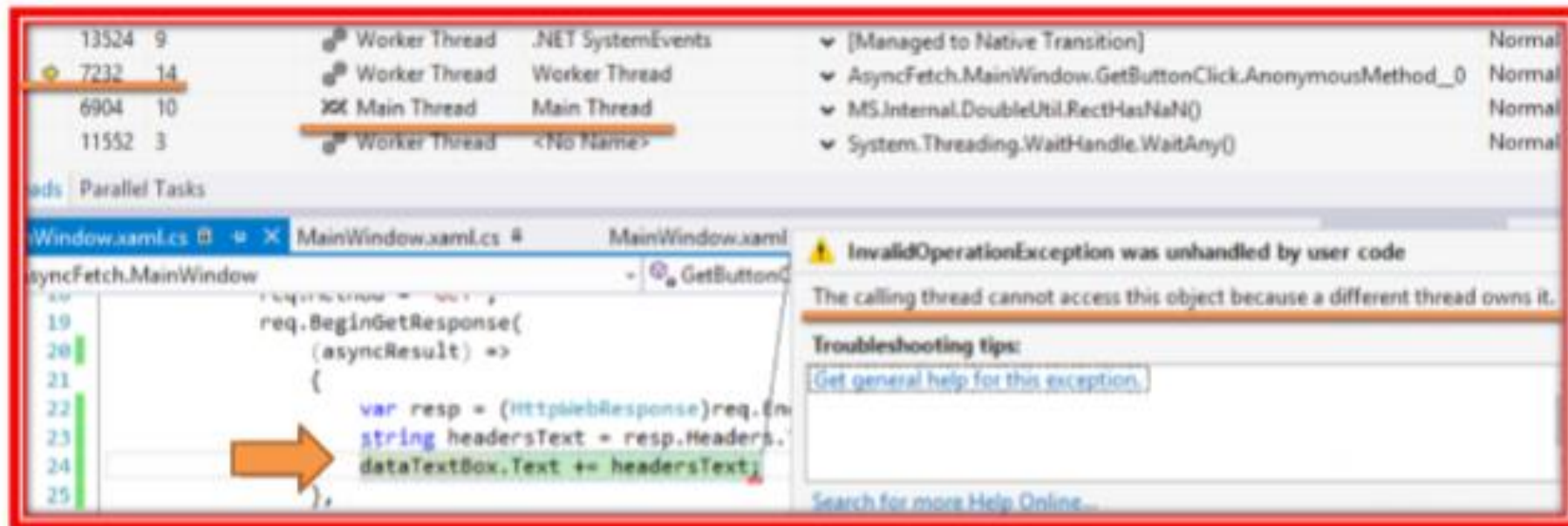
```
var req = (HttpWebRequest)WebRequest.Create("http://www.google.com");  
  
req.Method = "GET";  
  
req.BeginGetResponse((asyncResult) =>  
    {  
        var resp = (HttpWebResponse)req.EndGetResponse(asyncResult);  
  
        string headersText = resp.Headers.ToString();  
  
        dataTextBox.Text += headersText;  
    },  
    null);
```



C# Professional

Асинхронная модель до C# 5

Попытка обновления UI из другого потока



C# Professional

Асинхронная модель: обновление UI

Обновление UI из другого потока через [SynchronizationContext](#)

```
var sync = SynchronizationContext.Current;

req.BeginGetResponse(asyncResult=>
{
    var resp = (HttpWebResponse)req.EndGetResponse(asyncResult);

    sync.Post(delegate
    {
        // ОБНОВЛЕНИЕ UI
    },
    null);
},
null);
```

C# Professional

Асинхронная простота

Ключевое слово `async` указывает компилятору, что метод является асинхронным.

```
async void getButton_Click(object sender, RoutedEventArgs e)
{
    var w = new WebClient();

    string txt = await w.DownloadStringTaskAsync("...");

    dataTextBox.Text = txt;
}
```

`await` указывает компилятору, что в этой точке необходимо дождаться окончания асинхронной операции (при этом управление возвращается вызвавшему методу).

C# Professional

Асинхронная простота

```
async void DoDownloadAsync()
{
    using (var w = new WebClient())
    {
        string txt = await w.DownloadStringTaskAsync("http://www.microsoft.com/");
        dataTextBox.Text = txt;
    }
}

void DoDownload()
{
    using (var w = new WebClient())
    {
        string txt = w.DownloadString("http://www.microsoft.com/");
        dataTextBox.Text = txt;
    }
}
```

C# Professional

Исключения

Удобная обработка исключений

The screenshot shows the Visual Studio IDE with a debugger window open. The 'Threads' window at the top lists three threads: 'Worker Thread' (ID 13056), 'Worker Thread' (ID 9940), and 'Main Thread' (ID 9648). The 'Main Thread' is selected. Below it, the 'Parallel Tasks' window shows a task 'AsyncFetch.MainWindow.DoDownloadAsync'. The 'Code' window displays the source code for 'MainWindow.xaml.cs'. The code includes a 'DoDownloadAsync()' method that calls 'Debug.WriteLine("DoDownload done");' and an 'async void DoDownloadAsync()' method that uses 'WebClient' to download a string from 'http://www.microsoft.com'. The exception 'WebException was unhandled by user code' is shown in the 'Exceptions' window. The message is 'The remote name could not be resolved: 'www.microsoft.com''. The 'Troubleshooting tips' section suggests checking the 'Status' and 'Response' properties of the exception. The 'Exception settings' section shows a checkbox 'Break when this exception type is user-unhandled' which is checked.

```
13056 8 Worker Thread vshost.RunParkingWindow [Managed to Native Transition] Normal
9940 9 Worker Thread .NET SystemEvents [Managed to Native Transition] Normal
9648 10 Main Thread Main Thread AsyncFetch.MainWindow.DoDownloadAsync Normal
```

Parallel Tasks

MainWindow.xaml.cs # MainWindow.xaml.cs # MainWindow.xaml.cs # Ma

AsyncFetch.MainWindow

```
18 DoDownloadAsync();
19 Debug.WriteLine("DoDownload done");
20 }
21
22 async void DoDownloadAsync()
23 {
24     using (var w = new WebClient())
25     {
26         string txt = await w.DownloadStringTaskAsync("http://www.microsoft.com");
27         dataTextBox.Text = txt;
```

WebException was unhandled by user code

The remote name could not be resolved: 'www.microsoft.com'

Troubleshooting tips:

- Check the Status property of the exception to determine w
- Check the Response property of the exception to determin
- Get general help for this exception.

Search for more Help Online...

Exception settings:

- ☒ Break when this exception type is user-unhandled

Исключения «выбрасываются» в месте вызова асинхронной операции, а не Callback-метода!

C# Professional

Асинхронная и Многопоточность

`Task.WhenAll()` , `Task.WhenAny()`

```
var wc1 = new WebClient();  
var wc2 = new WebClient();  
  
Task<string> task1 = wc1.DownloadStringTaskAsync(url1);  
Task<string> task2 = wc2.DownloadStringTaskAsync(url2);  
  
string[] results = await Task.WhenAll(task1, task2);
```

Q&A

Информационный видеосервис для разработчиков программного обеспечения

