

C# Professional

Async & Await

C# Starter

Автор курса



Александр Шевчук
MCT



MCID: 9230440

C# Professional

После урока обязательно



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

Доступ можно получить через руководство вашего учебного центра



Проверьте как Вы усвоили данный материал на [TestProvider.com](http://testprovider.com)

C# Professional

Tema

Async & Await

Async & Await

C# 5.0 - What's new?

C# 2 = Generics, Lambda

C# 3 = var, LINQ

C# 4 = dynamic, TPL

C# 5 = Async

Async & Await

Синхронность: проблемы

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

    .....

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

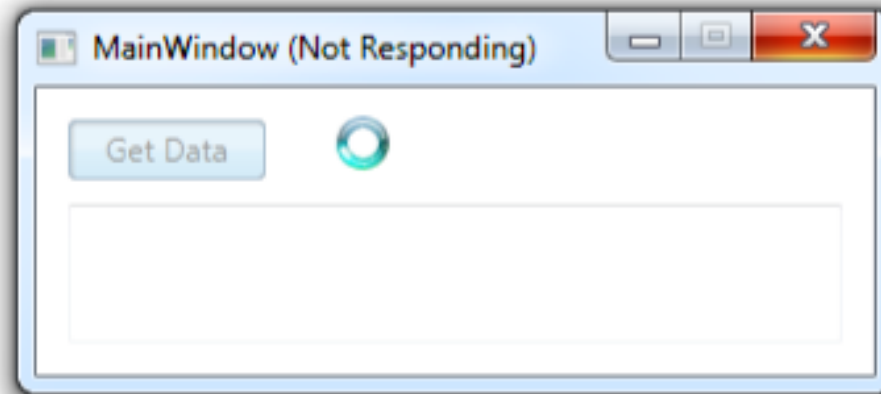
    dataTextBox.Text+= resp.Headers.ToString();
}
```



Async & Await

Синхронность: проблемы

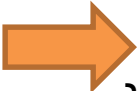
Вызывающий поток блокируется, до завершения длительной операции



Async & Await

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);
```



Async & Await

Асинхронная модель до C# 5

Попытка обновления UI из другого потока

The screenshot illustrates a cross-thread UI update attempt in a .NET application. The top thread dump shows the Main Thread (ID 6904) and several Worker Threads (IDs 13524, 7232, 11552). The code editor below shows the `GetButton_Click` method in `MainWindow.xaml.cs`, where an asynchronous request is made and the result is used to update `dataTextBox.Text`. An orange arrow points to this update line. On the right, the Exception Details pane shows an `InvalidOperationException` with the message: "The calling thread cannot access this object because a different thread owns it." This occurs because the UI control is accessed from a worker thread instead of the main thread.

Process ID	Thread ID	Thread Name	App Domain	Exception	Exception Type
13524	9	Worker Thread	.NET SystemEvents	[Managed to Native Transition]	Normal
7232	14	Worker Thread	Worker Thread	AsyncFetch.MainWindow.GetButton_Click.AnonymousMethod_0	Normal
6904	10	Main Thread	Main Thread	MS.Internal.DoubleUtil.RectHasNaN()	Normal
11552	3	Worker Thread	<No Name>	System.Threading.WaitHandle.WaitAny()	Normal

```
18 req.Method = GET;
19 req.BeginGetResponse(
20     (asyncResult) =>
21     {
22         var resp = (HttpWebResponse)req.EndGetResponse();
23         string headersText = resp.Headers.ToString();
24         dataTextBox.Text += headersText;
25     },
```

InvalidOperationException was unhandled by user code
The calling thread cannot access this object because a different thread owns it.

Troubleshooting tips:
[Get general help for this exception.](#)

[Search for more Help Online...](#)

Async & Await

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);
```

Async & Await

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).

Async & Await

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;
    }
}
```

Async & Await

Исключения

Удобная обработка исключений

The screenshot displays the Visual Studio IDE with the Async & Await window open. The window shows a list of tasks, including a Main Thread task for AsyncFetch.MainWindow.DoDownloadAsync. The code editor shows the implementation of DoDownloadAsync, which uses await to call DownloadStringTaskAsync. An exception, WebException, is shown in the Exception settings pane, indicating that the remote name could not be resolved. The exception settings pane also provides troubleshooting tips and a checkbox to break when the exception is user-unhandled.

Task ID	Thread	Task Name	Task State	Task Priority
13056	8	Worker Thread	vshost.RunParkingWindow	[Managed to Native Transition]
9940	9	Worker Thread	.NET SystemEvents	[Managed to Native Transition]
9648	10	Main Thread	Main Thread	AsyncFetch.MainWindow.DoDownloadAsync

```
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;
28     }
29 }
```

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 what the error was.
- Check the Response property of the exception to determine the response from the server.
- 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!

Async & Await

Асинхронность 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);
```

Смотрите наши уроки в видео формате

ITVDN.com



Посмотрите этот урок в видео формате на образовательном портале [ITVDN.com](http://itvdn.com) для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics



Проверка знаний

TestProvider.com

TestProvider | Мы помогаем людям оценить себя

Регистрация Войти

Главная Каталог Сертификация Microsoft Поддержка О нас

Тестирование

Языки программирования и информационные технологии

Microsoft

C# ASP.NET MVC JavaScript Patterns OF Design SQL Architecture Guide WCF HTML&CSS XML SEO WPF HTML5&CSS3 JQuery XNA SharePoint GUI for Android Windows Azure Platform Microsoft Patterns&Practices TFS SCRUM ReSharper TDD WWF LINQ Entity Framework Windows Forms Refactoring Microsoft Expression Blend 4 Windows Phone 8 Windows 8 AppStore Visual Studio Tips&Tricks MSF MEF SilverLight AJAX MEF Service Oriented Architecture

Пройти тест

Наши партнеры

Microsoft Partner CyberBionic ITVDN PROMETRIC TEST CENTER PEARSON VUE Authorized Test Center Windows Azure Cloud Partner EBA

Дополнительные ресурсы:

Очное обучение On-line обучение Видео обучение

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на [TestProvider.com](https://testprovider.com)

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.



Q&A

Информационный видеосервис для разработчиков программного обеспечения

