



ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование



Меня хорошо видно && слышно?

Ставьте смайлик в чат, если все хорошо
Напишите, если есть проблемы

Правила вебинара



Активно участвуем и включаем камеры



Задаем вопросы голосом или в чат

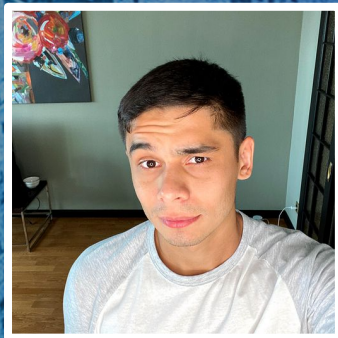


~~Off-topic обсуждаем в Slack~~



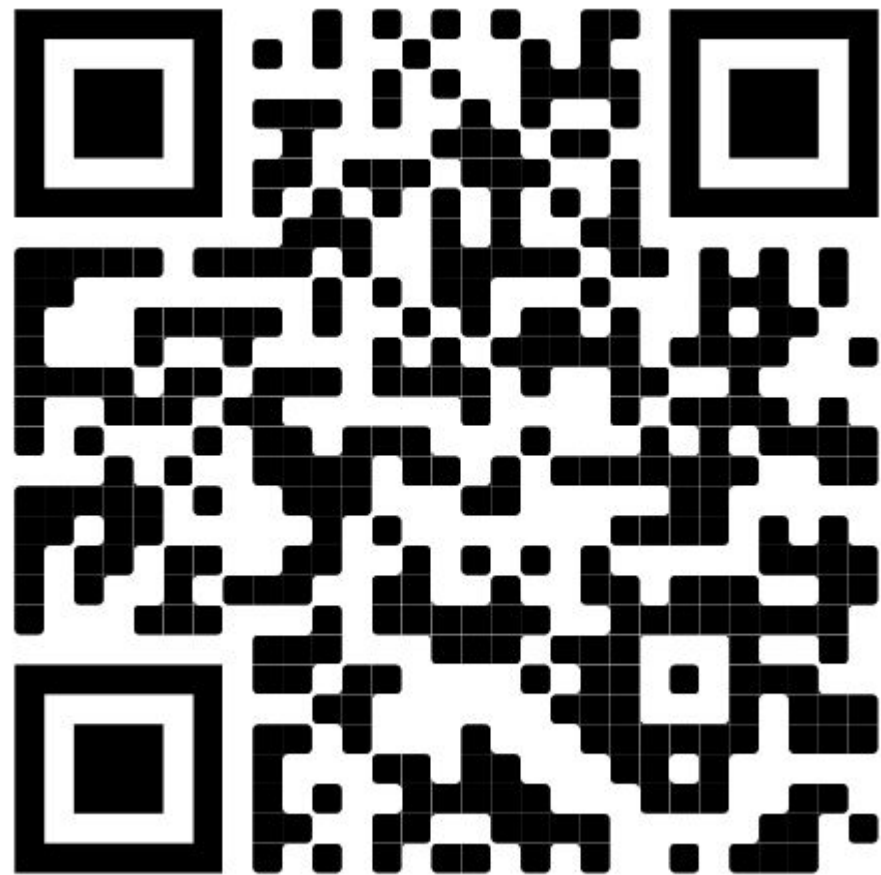
Вопросы в чате вижу, могу ответить не сразу

Асинхронные операции



Максим Горбатьюк
Lead Software Developer
TG: @maximgorbatyuk

Ссылка на чат



Цели вебинара | После занятия вы сможете

1

Рассказать, что такое асинхронность
и
где стоит ее применять

2

Писать асинхронный код

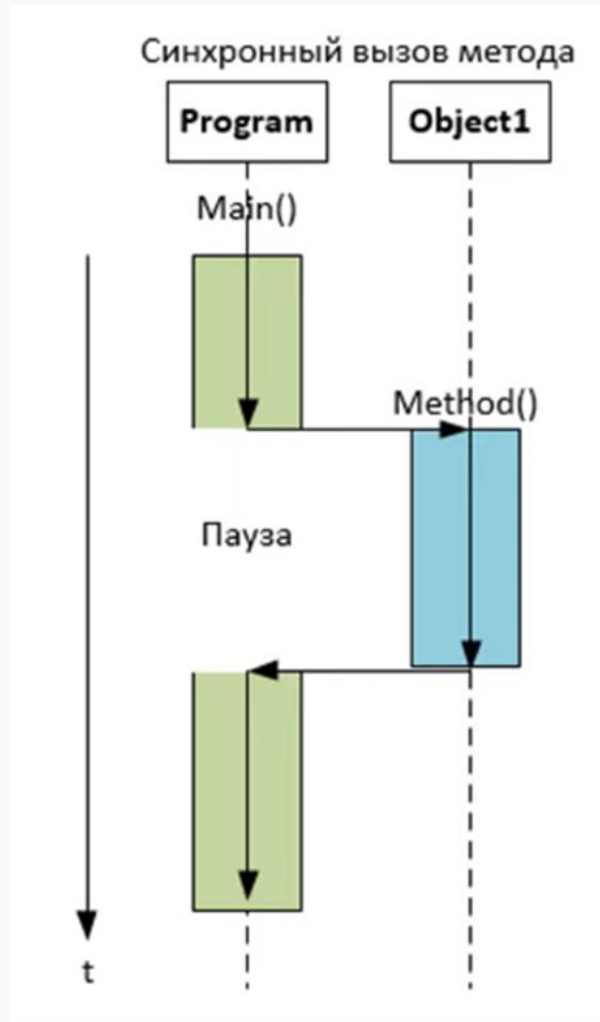
План занятия

- Синхронность и асинхронность
- Асинхронные методы
- Применение асинхронности
- Обработка исключений и отмена операций
- Контекст синхронизации

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band across the middle, which contains a white network diagram of interconnected nodes and lines. The title text is centered within this band.

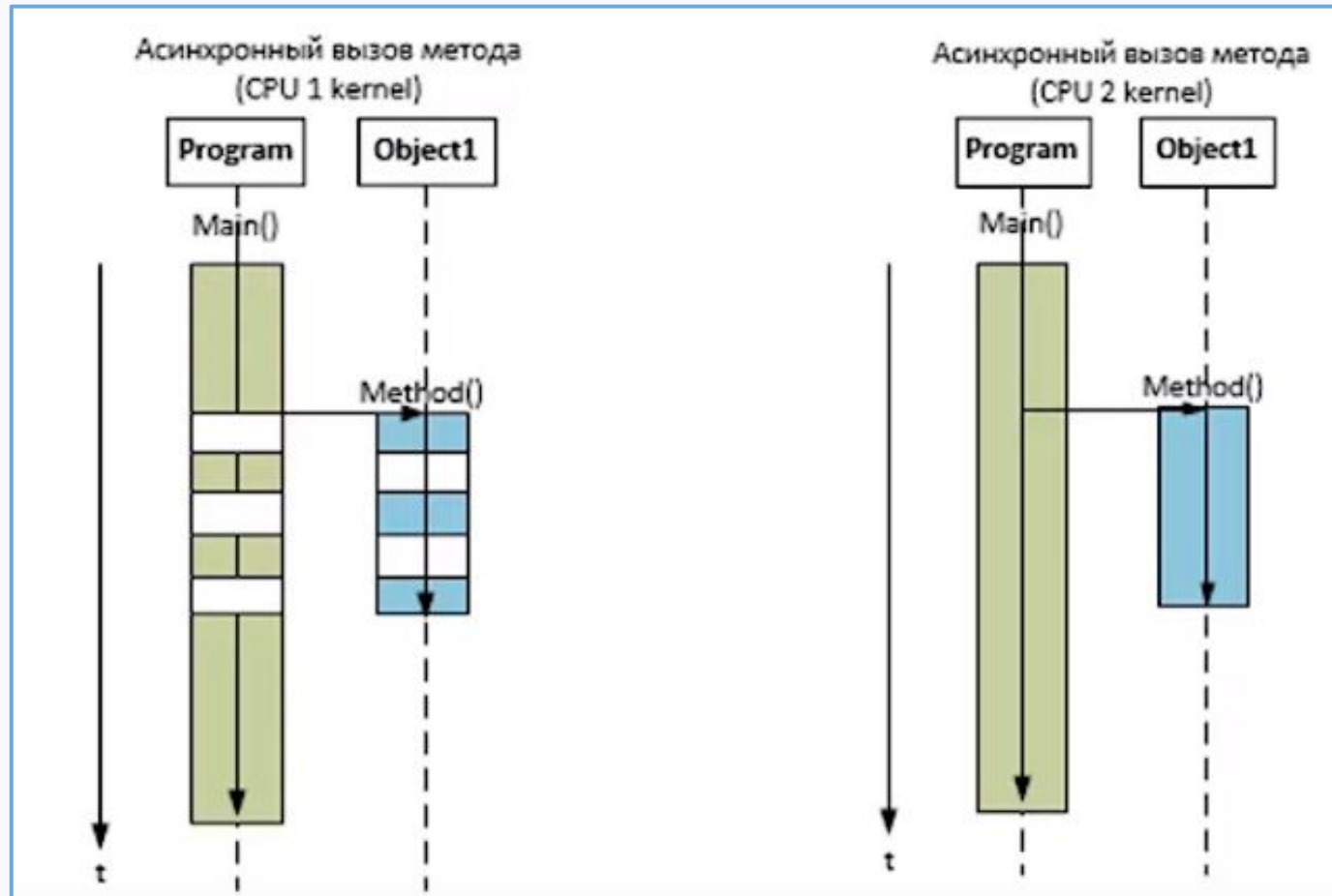
Синхронность и асинхронность

Синхронность



- Выполняем операцию в другом потоке
- Первый поток ставится на паузу
- Ждем, пока не выполнится операция

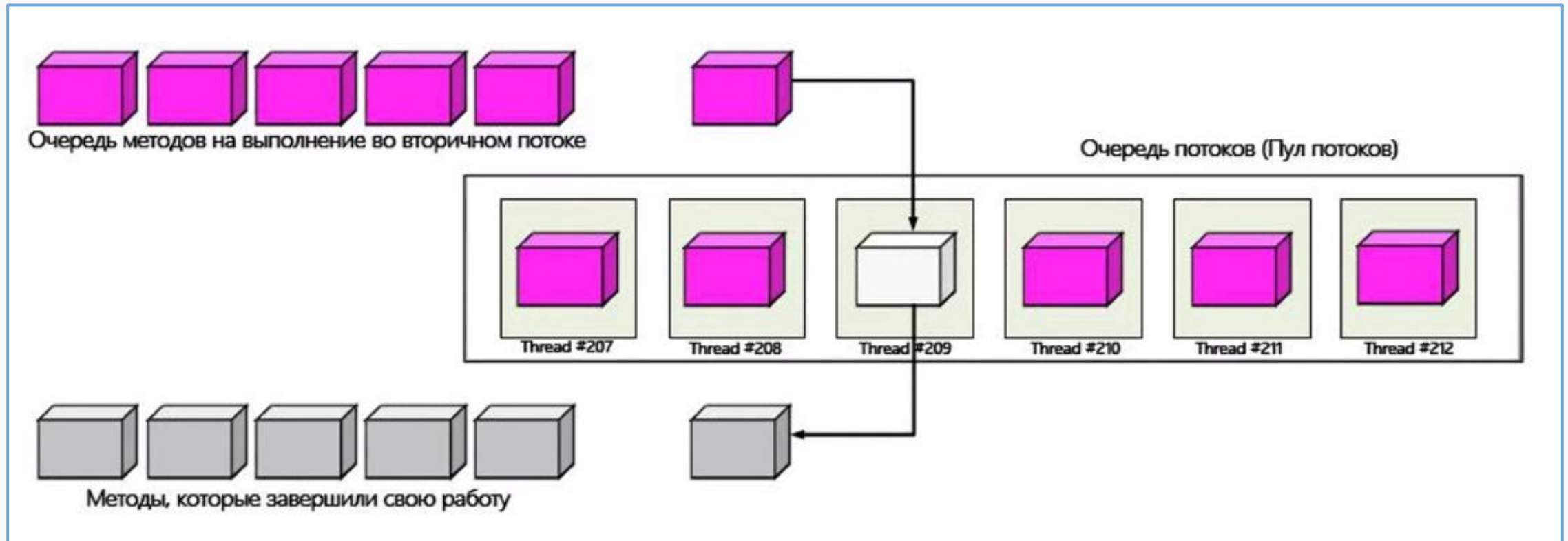
Асинхронность



- Выполняем операцию в другом потоке
- Первый поток “освобождается”
- Нет блокировки вызова

Пул потоков (Thread Pool)

Набор уже созданных потоков, готовых к выполнению задач



The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of light blue lines connecting various points, creating a mesh-like effect. The text is centered within this pattern.

Асинхронные паттерны

Асинхронные паттерны

- EAP
- APM
- TAP

EAP

- Event-based Asynchronous Pattern
- Операция выполняется по какому-то событию

```
1 public class MyClass
2 {
3     public void ReadAsync(byte [] buffer, int offset, int count);
4     public event ReadCompletedEventHandler ReadCompleted;
5 }
```



APM

- Asynchronous Programming Mode
- Иначе называют `IAsyncResult`

```
1 public class MyClass
2 {
3     public IAsyncResult BeginRead(
4         byte [] buffer, int offset, int count, AsyncCallback callback, object state);
5     public int EndRead(IAsyncResult asyncResult); |
6 }
```

ТАР (рекомендуемый способ)

- Task-based Asynchronous Pattern
- Можно применять `async/await`



```
1 public class MyClass
2 {
3     public Task<int> ReadAsync(byte [] buffer, int offset, int count);
4 }
```


TAP (рекомендуемый способ)

- Task-based Asynchronous Pattern
- Можно применять async/await

```
Task taskStart = new Task(() => Console.WriteLine("Run Task"));  
taskStart.Start();
```

```
Task taskRun = Task.Run(() => Console.WriteLine("Run Task"));
```

```
Task taskFactory = Task.Factory.StartNew(() => Console.WriteLine("Run Task"));
```




Асинхронные методы

План занятия

- Синхронность и асинхронность
- Асинхронные методы
- Применение асинхронности
- Обработка исключений и отмена операций
- Контекст синхронизации

Примеры асинхронных методов

```
public async Task PrintMeAsync()  
{  
    await Task.Run(() => Console.WriteLine("Printing"));  
}
```

```
public async Task<int> MultiplyMeAsync(int a, int b)  
{  
    return await Task.Run(function: () => a * b);  
}
```

```
public async void KillMeAsync()  
{  
    await Task.Run(() => Console.WriteLine("Nooooo"));  
}
```


Типы возвращаемых значений

- Task — для асинхронного метода, не возвращающего значение
- Task<TResult> — для асинхронного метода, возвращающего значение
- void — для обработчика событий (event handler)
- IEnumerable<T>* — для асинхронного метода, который возвращает асинхронный поток.

* - для c# версии 8.0 и выше



Применение асинхронности

Применение асинхронности

- Пользовательский интерфейс

Применение асинхронности

- Пользовательский интерфейс
- Второстепенные задачи

Применение асинхронности

- Пользовательский интерфейс
- Второстепенные задачи
- Одновременная обработка нескольких клиентских запросов

Применение асинхронности

- Пользовательский интерфейс
- Второстепенные задачи
- Одновременная обработка нескольких клиентских запросов
- Запросы в базу данных

Применение асинхронности

- Пользовательский интерфейс
- Второстепенные задачи
- Одновременная обработка нескольких клиентских запросов
- Запросы в базу данных
- Работа с файловой системой

Применение асинхронности

- Пользовательский интерфейс
- Второстепенные задачи
- Одновременная обработка нескольких клиентских запросов
- Запросы в базу данных
- Работа с файловой системой
- Сетевые запросы



LIVE



The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. This diagram consists of numerous small dots connected by thin white lines, creating a web-like structure that spans the width of the slide. Centered within this band is the main title in a large, white, sans-serif font.

Обработка исключений

Обработка ошибок в асинхронных методах

Для обработки ошибок выражение `await` помещается в блок `try`

```
try
{
    await DoSomethingAsync();
}
catch (Exception e)
{
    Console.WriteLine(e);
}
```

```
Task task = null;
try
{
    task = DoSomethingAsync();
    await task;
}
catch (Exception e)
{
    Console.WriteLine("Exception: " + e.Message);
    Console.WriteLine("IsFaulted: " + task.IsFaulted);
}
```

```
Task allTasks = null;
try
{
    var task1 = DoSomethingAsync();
    var task2 = DoSomethingAsync();
    var task3 = DoSomethingAsync();

    allTasks = Task.WhenAll(task1, task2, task3);
    await allTasks;
}
catch (Exception e)
{
    Console.WriteLine("Exception: " + e.Message);
    Console.WriteLine("IsFaulted: " + allTasks.IsFaulted);
    foreach (var inx:Exception in allTasks.Exception.InnerExceptions)
    {
        Console.WriteLine("internal exception: " + inx.Message);
    }
}
```




LIVE



The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band across the middle, which contains a white network diagram of interconnected nodes and lines. The title text is centered within this band.

Отмена асинхронных операций

Отмена асинхронных операций

```
CancellationTokenSource cts = new CancellationTokenSource();  
CancellationToken token = cts.Token;
```

```
private async Task DoSmtAsync(CancellationToken token)  
{  
    Console.WriteLine("Executing");  
    while (true)  
    {  
        Console.Write(".");  
        await Task.Delay(TimeSpan.FromMilliseconds(200), token);  
    }  
}
```




LIVE





Контекст синхронизации

Контекст синхронизации

- Планировщик задач

Контекст синхронизации

- Планировщик задач
- Определяет, какой поток будет выполнять задачу

Контекст синхронизации

- Планировщик задач
- Определяет, какой поток будет выполнять задачу
- Разные платформы – разные переопределения

Контекст синхронизации

- Планировщик задач
- Определяет, какой поток будет выполнять задачу
- Разные платформы – разные переопределения
- отсутствует в ASP.NET Core

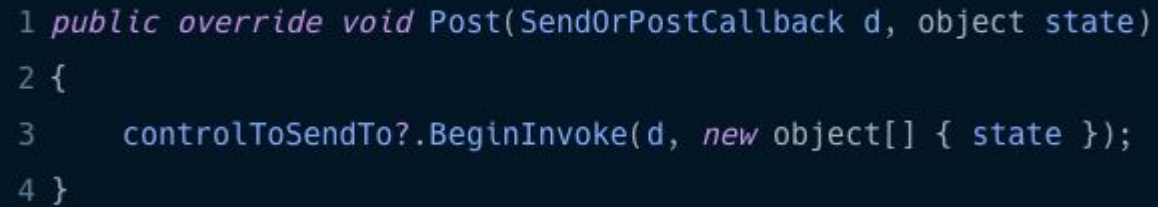
Стандартный КС

- <https://github.com/dotnet/runtime/blob/5e67c2480d8b9361923566243c1395a3d1a5d617/src/libraries/System.Private.CoreLib/src/System.Threading/SynchronizationContext.cs#L25>

```
1 public virtual void Post(SendOrPostCallback d, object? state) =>
    ThreadPool.QueueUserWorkItem(s => s.d(s.state), (d, state), preferLocal: false);
```

Windows Forms KC

- <https://github.com/dotnet/winforms/blob/94ce4a2e52bf5d0d07d3d067297d60c8a17dc6b4/src/System.Windows.Forms/src/System/Windows/Forms/WindowsFormsSynchronizationContext.cs#L88>



```
1 public override void Post(SendOrPostCallback d, object state)
2 {
3     controlToSendTo?.BeginInvoke(d, new object[] { state });
4 }
```


WPF Forms KC

- <https://github.com/dotnet/wpf/blob/ac9d1b7a6b0ee7c44fd2875a1174b820b3940619/src/Microsoft.DotNet.Wpf/src/WindowsBase/System/Windows/Threading/DispatcherSynchronizationContext.cs#L86>

```
1 /// <summary>
2 ///     Asynchronously invoke the callback in the SynchronizationContext.
3 /// </summary>
4 public override void Post(SendOrPostCallback d, Object state)
5 {
6     // Call BeginInvoke with the cached priority. Note that BeginInvoke
7     // preserves the behavior of passing exceptions to
8     // Dispatcher.UnhandledException unlike InvokeAsync. This is
9     // desirable because there is no way to await the call to Post, so
10    // exceptions are hard to observe.
11    _dispatcher.BeginInvoke(_priority, d, state);
12 }
```




`.ConfigureAwait(false)`

.ConfigureAwait(false)

- Оборачивает оригинальную задачу новой структурой

.ConfigureAwait(false)

- Оборачивает оригинальную задачу новой структурой
- Предотвращает вызов коллбека в исходном контексте

.ConfigureAwait(false)

- Оборачивает оригинальную задачу новой структурой
- Предотвращает вызов коллбека в исходном контексте



```
1 object scheduler = null;
2 if (continueOnCapturedContext)
3 {
4     scheduler = SynchronizationContext.Current;
5     if (scheduler is null && TaskScheduler.Current != TaskScheduler.Default)
6     {
7         scheduler = TaskScheduler.Current;
8     }
9 }
```


The image features a blue-tinted aerial view of a city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue band with a white network pattern of dots and lines runs horizontally across the middle of the image. The text "To sum up" is centered within this band in a white, sans-serif font.

To sum up

Не делай

- Не используй `void`, если это не обработчик событий(event handler)

Не делай

- Не используй `void`, если это не обработчик событий(event handler)
- Никогда не блокируй асинхронные операции в асинхронном коде вызовом методов `GetResult()` или `Wait()`

Не делай

- Не используй `void`, если это не обработчик событий(event handler)
- Никогда не блокируй асинхронные операции в асинхронном коде вызовом методов `getResult()` или `Wait()`
- не применяй `.ConfigureAwait(false)` в коде WPF и WinForms – это нарушит работу контролов

Делай

- Используй `async` и `await` вместе

Делай

- Используй `async` и `await` вместе
- Возвращай `Task` из асинхронных методов

Делай

- Используй `async` и `await` вместе
- Возвращай `Task` из асинхронных методов
- Используй постфикс `...Async` для именования асинхронных методов. Так при чтении кода будет легче

Делай

- Используй `async` и `await` вместе
- Возвращай `Task` из асинхронных методов
- Используй постфикс `...Async` для именования асинхронных методов. Так при чтении кода будет легче
- Пишешь библиотеку – добавляй `.ConfigureAwait(false)`

Делай

- Используй `async` и `await` вместе
- Возвращай `Task` из асинхронных методов
- Используй постфикс `...Async` для именования асинхронных методов. Так при чтении кода будет легче
- Пишешь библиотеку – добавляй `.ConfigureAwait(false)`
- Используешь `ASP.NET Core` – можешь не применять `.ConfigureAwait(false)`

The background of the image is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this is a semi-transparent blue band that contains a white network diagram of interconnected nodes and lines. Centered within this band is the text "Демо-репозиторий" in a white, sans-serif font.

Демо-репозиторий

Демо-репозиторий

[https://github.com/gm-soft/
AsyncProgrammingDemo](https://github.com/gm-soft/AsyncProgrammingDemo)

The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this image is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure. The text "Список литературы" is centered in the middle of the slide in a white, sans-serif font.

Список литературы

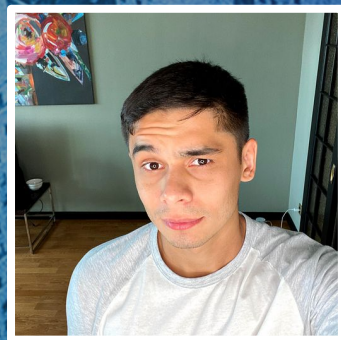
Список материалов

- <https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/>
- <https://github.com/davidfowl/AspNetCoreDiagnosticScenarios/blob/master/AsyncGuidance.md>
- <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async/task-asynchronous-programming-model>
- <https://devblogs.microsoft.com/pfxteam/executioncontext-vs-synchronizationcontext/>
- <https://habr.com/ru/post/416751/>
- <https://docs.microsoft.com/en-us/dotnet/api/system.threading.asynclocal-1?view=netcore-3.1>
- <https://ru.stackoverflow.com/a/461099>
- <https://ru.stackoverflow.com/a/750537>
- <https://habr.com/ru/post/482354/>
- <https://stackoverflow.com/a/39007110>
- <https://blog.stephencleary.com/2017/03/aspnetcore-synchronization-context.html>

Опрос о занятии по ссылке в чате



Ответы на вопросы



Максим Горбатюк

Lead Software Developer

TG: @maximgorbatyuk