async/await и все, что вы боялись спросить

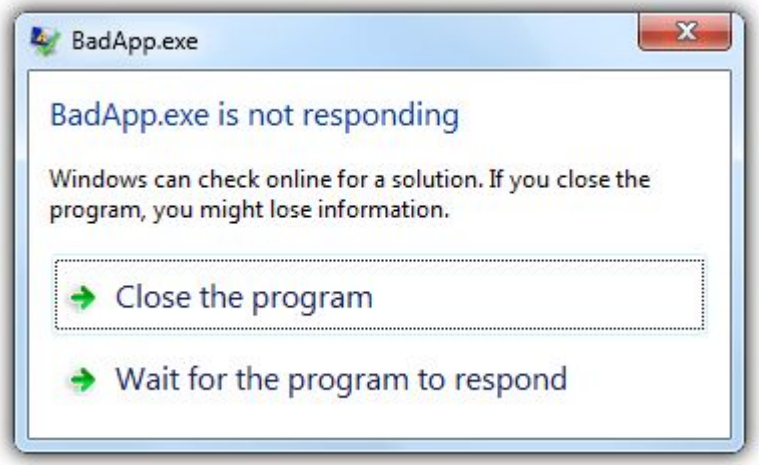
Гришечко Егор

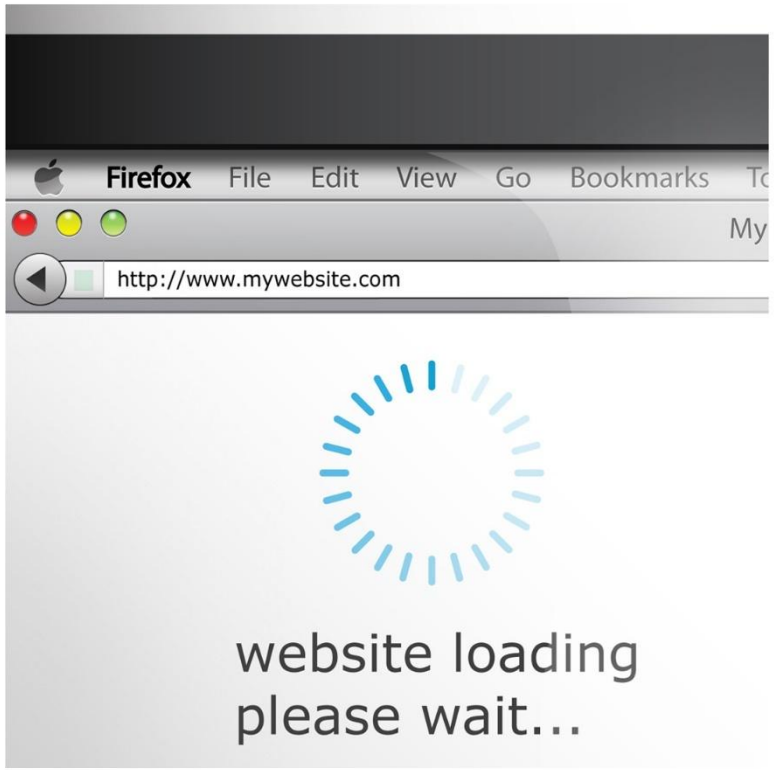
CodeBeavers

О чем буду разглагольствовать

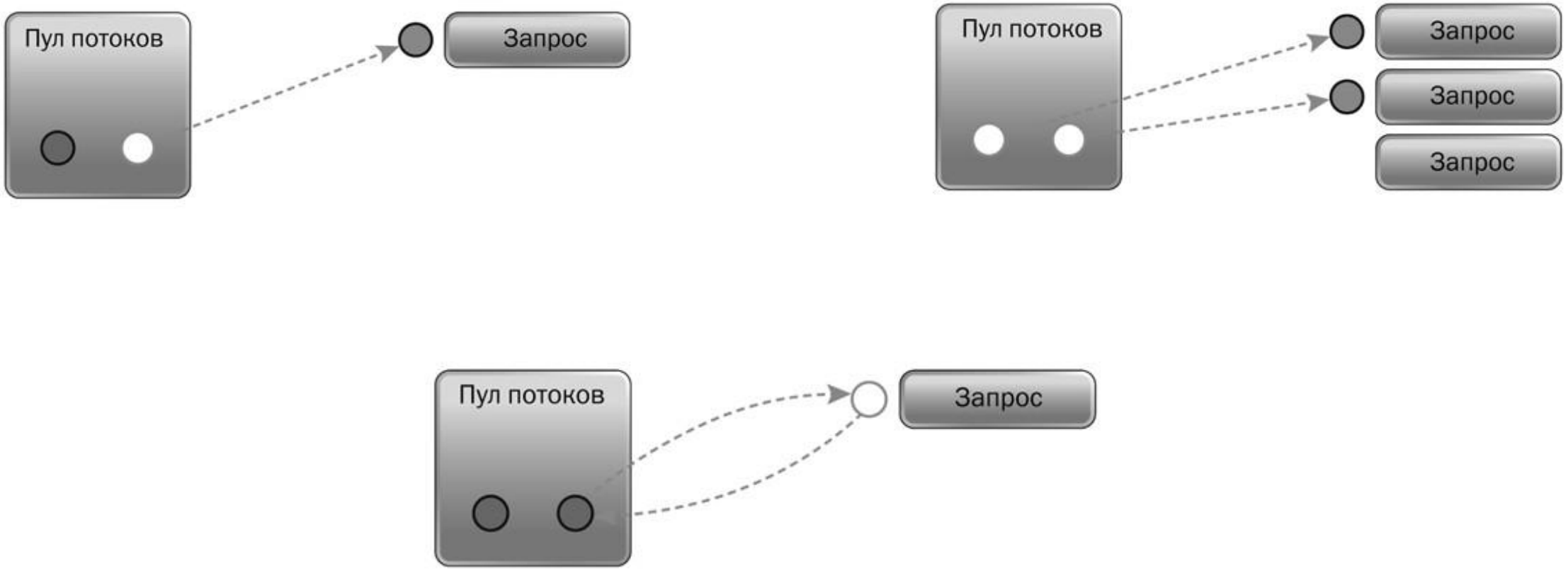
- Обосную, зачем этот доклад нужен
- Немного основ
- Копнем вглубь StateMachine
- Копнём вглубь контекстов и всего, что с ними связано
- Поумилияемся .NET Core
- Расскажу пару примеров из жизни
- Поспорим про паттерны и подходы

Зачем нам нужна асинхронность?





Зачем это нужно на сервере?



https://docs.microsoft.com/en-us/dotnet/standard/async-in-depth

Немногие основы

Как было?

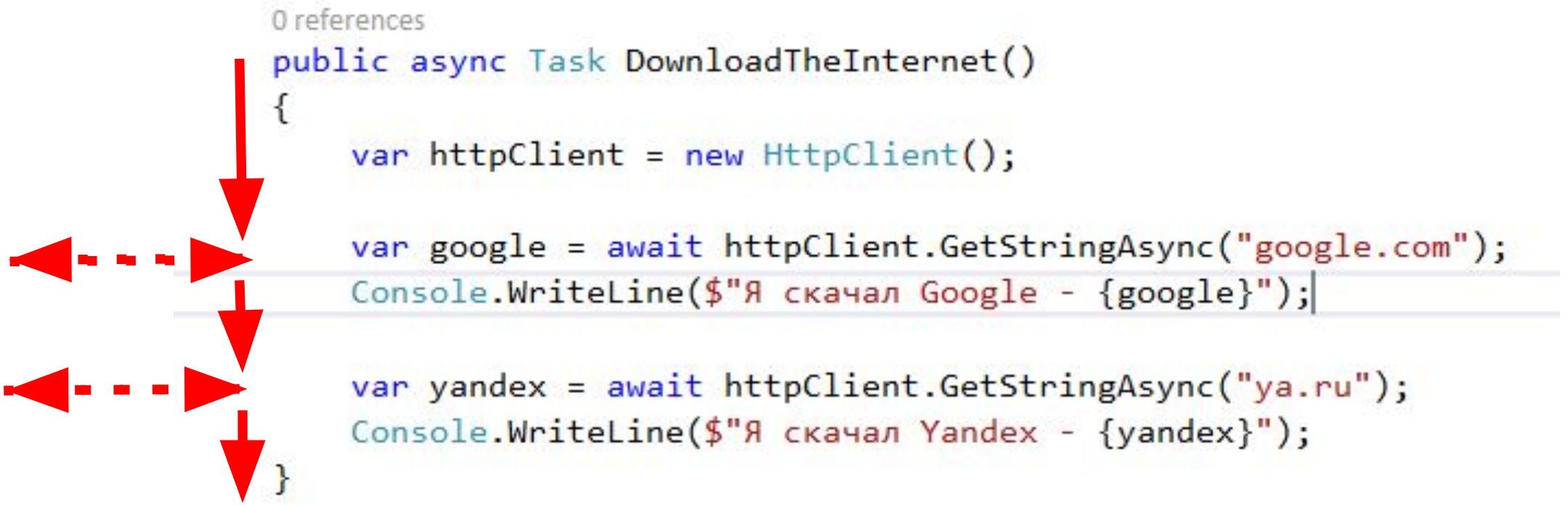
- Asynchronous Programming Model (**Begin***OperationName*/**End***OperationName*)
- Event-based Asynchronous Programming (*OperationName***Async**()/*OperationName***Completed**)

Как стало?

- Task-base Asynchronous Pattern (TAP) (**async/await и Task/Task<T>**)

- **Асинхронное программирование** – стиль программирование, в котором основной поток выполнения не блокируется. А выполнение кода превращается из последовательного в поток обратных вызовов (call back).

Поехали ☺️



```
0 references
public async Task DownloadTheInternet()
{
    var httpClient = new HttpClient();

    var google = await httpClient.GetStringAsync("google.com");
    Console.WriteLine($"Я скачал Google - {google}");

    var yandex = await httpClient.GetStringAsync("ya.ru");
    Console.WriteLine($"Я скачал Yandex - {yandex}");
}
```

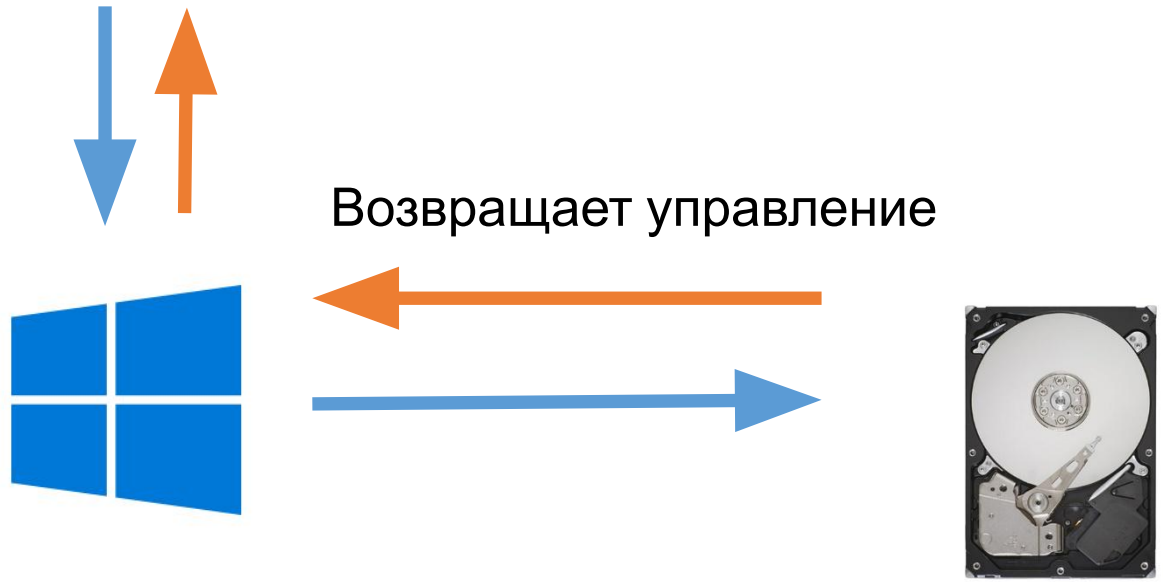
Как работает (синхронное)

```
0 references
static void Main(string[] args)
{
    var fileStream = new FileStream("testfile", FileMode.Open);
    byte[] output = new byte[250];
    fileStream.Read(output, 0, 250);
}
```

Поток заблокировался

Как работает (асинхронное)

```
0 references
static async void Main(string[] args)
{
    var fileStream = new FileStream("testfile", FileMode.Open);
    byte[] output = new byte[250];
    await fileStream.ReadAsync(output, 0, 250);
}
```



Поехали?

```
public class Example
{
    public async Task ShowStateMachine()
    {
        FirstSync();
        await Task.Delay(1000);
        SecondSync();
    }

    public void FirstSync()
    {
        Console.WriteLine("First");
    }

    public void SecondSync()
    {
        Console.WriteLine("Second");
    }
}
```

```
public Task ShowStateMachine()
{
    Example.ShowStateMachine stateMachine = new Example.ShowStateMachine()
    {
            That = this,
            Builder = AsyncTaskMethodBuilder.Create(),
            State = -1
    };
    stateMachine.Builder.Start(ref stateMachine);
    return stateMachine.Builder.Task;
}

public void FirstSync()
{
  Console.WriteLine("First");
}

public void SecondSync()
{
  Console.WriteLine("Second");
}
```

А что с Core?

```
public class Example
{
    public async Task ShowStateMachine()
    {
        FirstSync();
        await Task.Delay(1000);
        SecondSync();;
    }

    public void FirstSync()
    {
        Console.WriteLine("FCore");
    }

    public void SecondSync()
    {
        Console.WriteLine("SCore");
    }
}
```

```
public Task ShowStateMachine()
{
    Example.ShowStateMachine stateMachine = new Example.ShowStateMachine()
    {
            That = this,
            Builder = AsyncTaskMethodBuilder.Create(),
            State = -1
    };
    stateMachine.Builder.Start(ref stateMachine);
    return stateMachine.Builder.Task;
}

public void FirstSync()
{
  Console.WriteLine("FCore");
}

public void SecondSync()
{
  Console.WriteLine("SCore");
}
```

Вниз, к центру стэйт машины

```
namespace System.Runtime.CompilerServices
{
  public interface IAsyncStateMachine
  {
    void MoveNext();

    void SetStateMachine(IAsyncStateMachine stateMachine);
  }
}
```

Вниз, к центру стэйт машины

```
private sealed class ShowStateMachine : IAsyncStateMachine
{
  public int State;
  public AsyncTaskMethodBuilder Builder;
  public Example That;
  private TaskAwaiter _awaiter;

  void IAsyncStateMachine.MoveNext()
  {
      // Куча полезного кода
  }

  void IAsyncStateMachine.SetStateMachine(IAsyncStateMachine stateMachine)
  {
      Builder.SetStateMachine(stateMachine)
  }
}
```

```
void IAsyncStateMachine.MoveNext()
{
  try
  {
    if (State != 0)
    {
      That.FirstSync();
      _awaiter = Task.Delay(1000).GetAwaiter();
      if (!_awaiter.IsCompleted)
      {
        State = 0;
        Buider.AwaitUnsafeOnCompleted(ref _awaiter, ref this);
        return;
      }
    }
    else
    {
      State = -1;
    }
    awaiter.GetResult();
    That.SecondSync();
  }
  catch (Exception ex)
  {
    State = -2;
    Builder.SetException(ex);
    return;
  }
  State = -2;
  Builder.SetResult();
}
```

Совсем упростим

```
FirstSync();

var t = Task.Delay(10000);

var currentContext = SynchronizationContext.Current;
t.ContinueWith(task =>
{
    if (currentContext == null)
        SecondSync();
    else
        currentContext.Post(delegate { SecondSync(); }, null);
}, TaskScheduler.Current);
```

SynchronizationContext – это важно

Веселая задачка

```
public class HomeController : Controller
{
    public string Index()
    {
        var service = new Service();
        var result = service.Perfom().Result;
        return result;
    }
}
public class Service
{
    public async Task<string> Perfom()
    {
        var manager = new Manager();
        return await manager.Perfom();
    }
}
public class Manager
{
    public async Task<string> Perfom()
    {
        await Task.Delay(100);
        return "Bang";
    }
}
```


MOTORS
TV

Нам нужно на следующий уровень (Демо)

Parallel Programming with .NET

All about Async/Await, System.Threading.Tasks, System.Collections.Concurrent, System.Linq, and more...

ExecutionContext vs SynchronizationContext

★★★★☆

June 15, 2012 by Stephen Toub - MSFT // 38 Comments

Share 2 | 5 | 0

I've been asked a few times recently various questions about ExecutionContext and SynchronizationContext, for example what the differences are between them, what it means to "flow" them, and how they relate to the new async/await keywords in C# and Visual Basic. I thought I'd try to tackle some of those questions here.

WARNING: This post goes deep into an advanced area of .NET that most developers never need to think about.

What is ExecutionContext, and what does it mean to flow it?

ExecutionContext is one of those things that the vast majority of developers never need to think about. It's kind of like air: it's important that it's there, but except at some crucial times (e.g. when something goes wrong with it), we don't think about it being there. ExecutionContext is actually just a container for other contexts. Some of these other contexts are ancillary, while some are vital to the execution model of .NET, but they all follow the same philosophy I described for ExecutionContext: if you have to know they're there, either you're doing something super advanced, or something's gone wrong.

ExecutionContext is all about "ambient" information, meaning that it stores data relevant to the current environment or "context" in which you're running. In many systems, such ambient information is maintained in thread-local storage (TLS), such as in a ThreadStatic field or in a ThreadLocal<T>. In a synchronous world, such thread-local information is sufficient: everything's happening on that one thread, and thus regardless of what stack frame you're in on that thread, what function is being executed, and so forth, all code running on that thread can see

ExecutionContext – это тоже важно

ExecutionContext

```
public sealed class ExecutionContext
{
    // Куча полезного кода

    public static ExecutionContext Capture()

    public static void Run(ExecutionContext executionContext, ContextCallback callback, object state)

    // Куча полезного кода
}
```

ExecutionContext

```
public static class FuncExtensions
{
    public static TResult InvokeWith<TResult>(this Func<TResult> function, ExecutionContext executionContext)
    {
        if (executionContext == null)
        {
            return function();
        }

        TResult result = default(TResult);
        // See: System.Runtime.CompilerServices.AsyncMethodBuilderCore.MoveNextRunner.Run()
        ExecutionContext.Run(executionContext, _ => result = function(), null);
        return result;
    }
}
```

https://weblogs.asp.net/dixin/understanding-c-sharp-async-await-3-runtime-context

SynchronizationContext

```
public class SynchronizationContext
{
    // Выполняем некую единицу работы
    public virtual void Send(SendOrPostCallback d, object state) {}
    public virtual void Post(SendOrPostCallback d, object state) {}

    public static void SetSynchronizationContext(SynchronizationContext syncContext) {}
    public static SynchronizationContext Current
    public virtual SynchronizationContext CreateCopy() {}
}
```

SynchronizationContext

```
public static void DoWork()
{
        TextView.Text = "В UI потоке"
        var sc = SynchronizationContext.Current;
        ThreadPool.QueueUserWorkItem(delegate
        {
                // выполняется в рамках ThreadPool
                sc.Post(delegate
                {
                        // выполняем в рамках другого контекста, например UI
                        TextView.Text = "Снова в UI потоке"
                }, null);
        });
}
```

Веселая задачка №2

```
public class HomeController : Controller
{
    public async Task<ActionResult> Index()
    {
        if (System.Web.HttpContext.Current == null)
            throw new Exception("Все пропало");

        await Task.Delay(100).ConfigureAwait(false);

        if (System.Web.HttpContext.Current == null)
            throw new Exception("Все совсем пропало");

        return View();
    }
}
```

AspNetSynchronizationContext

```
internal sealed class AspNetSynchronizationContext : AspNetSynchronizationContextBase {
        // Куча важного кода
        public override void Post(SendOrPostCallback callback, Object state) {
            _state.Helper.QueueAsynchronous(() => callback(state));
        }
        // Куча важного кода
}
```

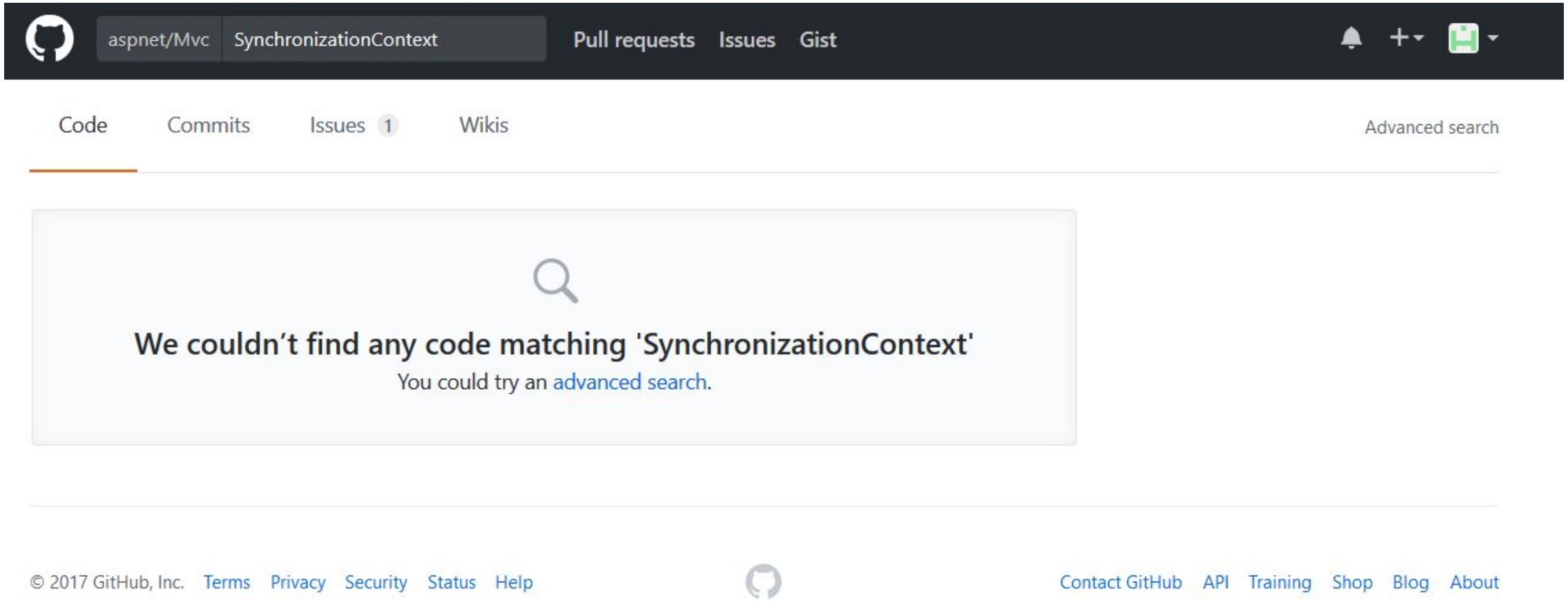
```
public void QueueAsynchronous(Action action) {
    CheckForRequestStateIfRequired(checkForReEntry: true);
    ChangeOperationCount(+1);

    // This method only schedules work; it doesn't itself do any work. The lock is held for a very
    // short period of time.
    lock (_lockObj) {
        Task newTask = _lastScheduledTask.ContinueWith
        (
                                                    _ => SafeWrapCallback(action),
                                                    TaskScheduler.Default

        );
        _lastScheduledTask = newTask; // the newly-created task is now the last one
    }
}
```

Мы же в 2к17

Упс...


aspnet/Mvc
SynchronizationContext
Pull requests
Issues
Gist
Code
Commits
Issues 1
Wikis
Advanced search
We couldn't find any code matching 'SynchronizationContext'
You could try an advanced search.
© 2017 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
API
Training
Shop
Blog
About

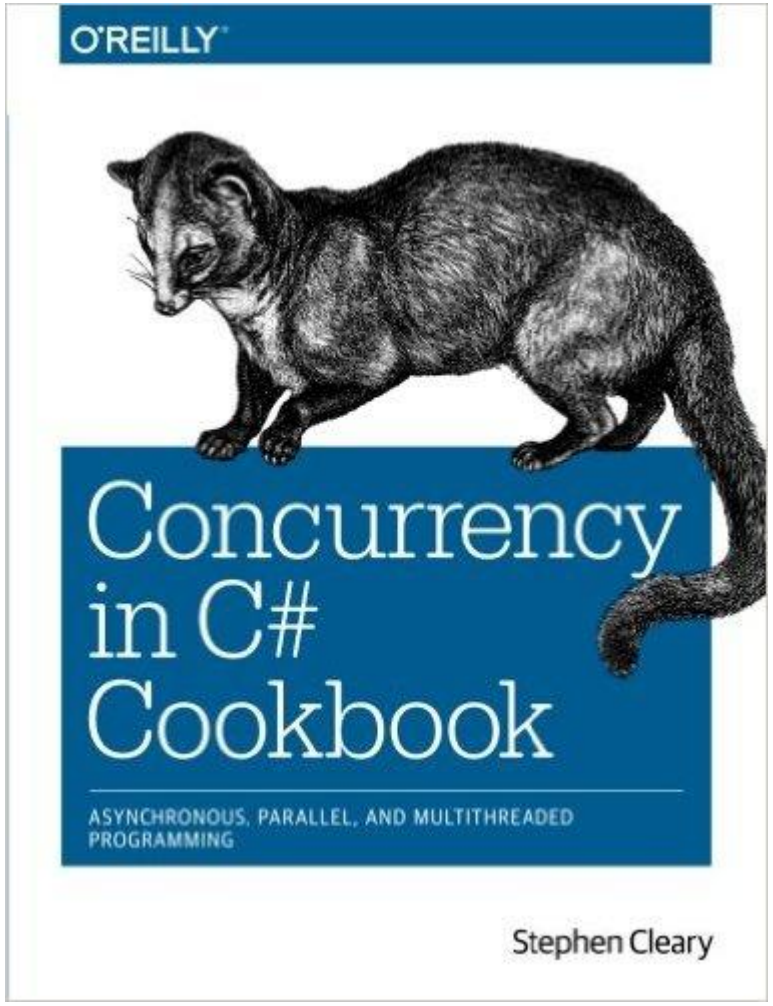
Немного опыта

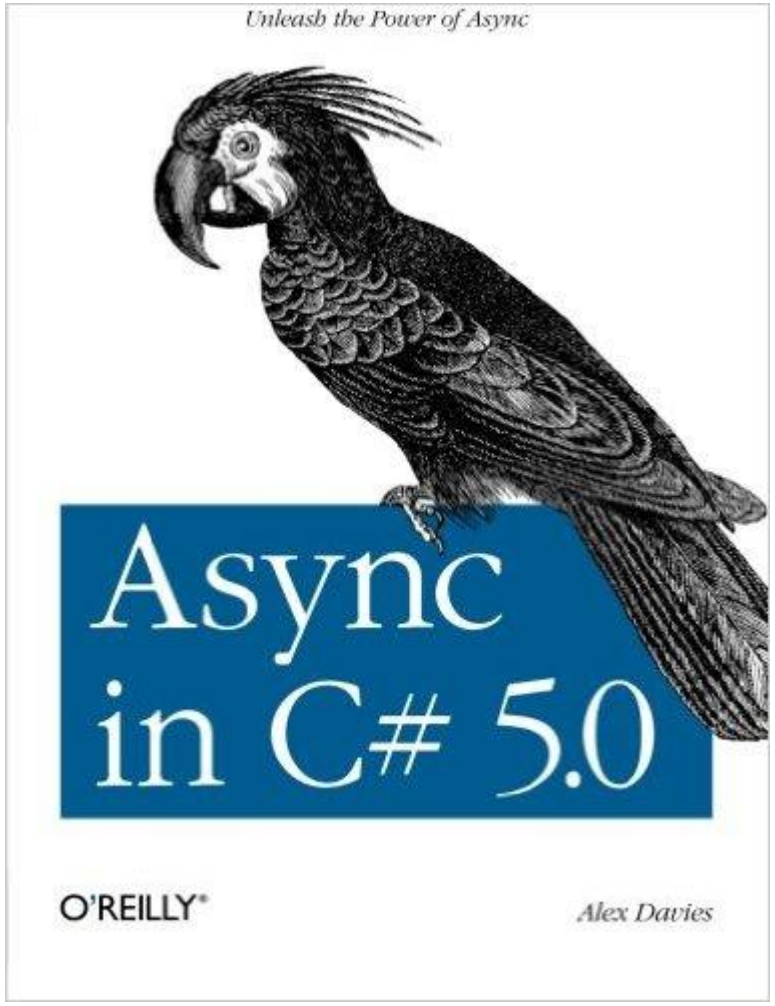
Похоливарим

- Не стоит оборачивать синхронный код в асинхронный
- Не стоит оборачивать асинхронный код в синхронный

https://blogs.msdn.microsoft.com/pfxteam/2012/04/13/should-i-expose-synchronous-wrappers-for-asynchronous-methods/
https://blogs.msdn.microsoft.com/pfxteam/2012/03/24/should-i-expose-asynchronous-wrappers-for-synchronous-methods/
https://docs.microsoft.com/en-us/dotnet/csharp/async

Что почитать?(книги)





Что еще?

Стать

и

- https://weblogs.asp.net/dixin/understanding-c-sharp-async-await-1-compilation – кратко и обо всем
- https://blog.stephencleary.com/2013/10/taskrun-etiquette-and-proper-usage.html – (Stephen Cleary)
- https://blogs.msdn.microsoft.com/pfxteam/ – команда отвечающая за async/await
- https://codeblog.jonskeet.uk/2011/05/08/eduasync-part-1-introduction/ – Jon Skeet
- http://vegetarianprogrammer.blogspot.ru/2012/12/understanding-synchronizationcontext-in.html – хорошая статья про контексты
- https://msdn.microsoft.com/en-us/magazine/gg598924.aspx – еще статья про контекст
- https://blogs.msdn.microsoft.com/pfxteam/2012/06/15/executioncontext-vs-synchronizationcontext/ – ExecutionContext vs SynchronizationContext
- https://blogs.msdn.microsoft.com/pfxteam/2012/03/24/should-i-expose-asynchronous-wrappers-for-synchronous-methods/ – нужно ли оборачивать синхронное в асинхронное
- https://blogs.msdn.microsoft.com/pfxteam/2012/04/13/should-i-expose-synchronous-wrappers-for-asynchronous-methods/ – нужно ли оборачивать синхронное в асинхронное

Что еще?

Виде

о

- https://www.youtube.com/watch?v=lh8cT6qI-nA – Андрей Часовских — Async/await: собираем грабли
- https://channel9.msdn.com/Events/aspConf/aspConf/Async-in-ASP-NET – async в ASP.NET
- https://channel9.msdn.com/Events/BUILD/BUILD2011/TOOL-829T – The zen of async: Best practices for best performance
- https://channel9.msdn.com/Series/Three-Essential-Tips-for-Async – Six Essential Tips for Async
- https://channel9.msdn.com/Series/Three-Essential-Tips-for-Async – Tip 4: Async Library Methods Shouldn't Lie

Минутка тщеславия ☺️

Большое спасибо за внимание!

Ссылки
:

- https://github.com/egorikas/SpbDotNet – презентация и примеры
- egorikas.com – мой блог
- egorgrishechko@gmail.com