Работа контекста синхронизации с async await
Роль async await в ASP.NET

№ урока: 5 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Урок познакомит вас с работой ключевых слов async await в приложениях WPF и ASP.NET. Задача урока состоит в том, чтобы рассмотреть работу продолжений оператора await при взаимодействии с контекстом синхронизации, планировщиком задач и пулом потоков для выполнения в их контексте продолжений. Будет рассмотрены способы повлиять на работу продолжений оператора await, особенности работы типа возвращаемого значения void с модификатором async, а также асинхронные лямбда выражения.

Изучив материал данного занятия, учащийся сможет:

- Понимать, какую особенность дают async await в WPF приложениях.
- Работать с контекстом синхронизации.
- Понимать работу продолжений оператора await.
- Конфигурировать работу продолжений оператора await.
- Понимать опасность использования async void.
- Создавать асинхронные лямбда выражения.
- Понимать, какую особенность дают async await в ASP.NET приложениях.

Содержание урока

1. Рассмотрение работы ключевых слов async await в WPF приложениях
2. Рассмотрение контекста синхронизации
3. Принцип работы продолжений оператора await
4. Продолжение await в контексте синхронизации
5. Продолжение await в пуле потоков
6. Конфигурирование продолжения await
7. Модификатор async и тип возвращаемого значения void
8. Асинхронные лямбда выражения
9. Рассмотрение работы ключевых слов async await в ASP.NET приложениях

Резюме

- Ключевые слова async await упрощают асинхронное программирование в приложениях WPF, благодаря выполнению продолжений оператора await в контексте синхронизации WPF. Вам не приходится самостоятельно (вручную) передавать данные из вторичного потока в основной UI поток.
- Контекст синхронизации – абстрактный механизм, который позволяет выполнить код вашего приложения в определенном месте. В .NET представлен классом System.Threading.SynchronizationContext.
- Для установки контекста синхронизации необходимо создавать производный класс от базового класса SynchronizationContext (использование базового класса SynchronizationContext скорее всего не то, что вам нужно) и установка его экземпляра в

CyberBionic systematics

CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

окружение потока через статический метод SyncrhonizationContext.SetSynchronizationContext().

- Продолжения оператора await могут выполняться с помощью разных механизмов. Это зависит от окружения оператора await. Но, при наличии нескольких механизмов одновременно, всегда будет выбран только один из них, наиболее приоритетный. Приоритетность определяется по порядку следования проверок наличия механизмов.
- Если в окружении оператора await находится контекст синхронизации, то внутренние механизмы оператора await его захватят. Если он не является обычным базовым типом SynchronizationContext, то продолжение будет выполнено через контекст синхронизации и его метод Post.
- Если в окружении оператора await находится планировщик задач, то внутренние механизмы оператора await его получат. Если он не является планировщиком по умолчанию (свойство TaskScheduler.Default), то он будет использован для выполнения продолжения. Так как планировщики работают только с задачами, для делегата продолжения будет создана новая задача.
- Если основные механизмы (контекст синхронизации, планировщик задач) не были найдены или было порекомендовано отказаться от захвата окружения (контекста синхронизации, планировщика задач) вызывающего потока, то продолжение будет выполнено либо синхронно в месте, где была завершена задача, либо в контексте пула потоков.
- Для конфигурирования ожидания (возможность отказаться от захвата окружения контекста синхронизации и планировщика задач) используется метод ConfigureAwait. Он принимает булевый параметр, который советует системе, каким образом вы хотите выполнить продолжение оператора await. Значение True – означает выполнение с захватом контекста синхронизации и планировщика задач, если они есть. Значение False – указывает, что необходимо отказаться от захвата контекста синхронизации и планировщика задач.
- По умолчанию продолжения оператора await ведут себя так, как будто для них вызывали явно метод ConfigureAwait(true).
- Если на момент применения оператора await асинхронная задача уже будет завершена, то код продолжит выполняться синхронно в том же потоке. Это значит, что указание ConfigureAwait(false) не сработает. Поэтому, его указание не гарантирует 100% выполнение указанным вами способом, это скорее способ рекомендации для выполнения системе.
- Контекст выполнения – это объект, который представляет контейнер для хранения информации потока выполнения. В .NET представлен классом System.Threading.ExecutionContext.
- Возвращаемый тип void при использовании модификатора async имеет взаимодействие с контекстом синхронизации. Оно описано на уровне строителя асинхронных методов AsyncVoidMethodBuilder.
- При выполнении асинхронного метода с типом void, при захвате контекста синхронизации, вначале будет вызван метод OperationStarted, по завершении OperationCompleted. Если во время выполнения асинхронного метода с типом void произойдет исключение, оно будет выброшено в метод Post контекста синхронизации. Если контекста синхронизации нет и получается, что он не был захвачен, то методы OperationStarted и OperationCompleted не будут вызваны, а если произойдет исключение – оно будет выброшено в пул потоков.
- Лямбда выражения можно сделать асинхронными, добавив модификатор async перед набором параметров лямбда выражения. На них накладываются все правила и особенности асинхронных методов.
- Используя асинхронное программирование в ASP.NET, вы можете увеличить пропускную способность входящих запросов веб приложения.

CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Для повышения пропускной способности используйте асинхронность с запросами ввода-вывода (файловая система, сетевые запросы, удаленная база данных, удаленные веб-сервисы).

Закрепление материала

- Чем именно ключевые слова async await упрощают асинхронное программирование в WPF?
- Что такое контекст синхронизации?
- Где могут выполняться продолжения оператора await?
- Как можно сконфигурировать продолжения оператора await?
- Что такое контекст выполнения?
- В чем особенности работы возвращаемого типа void для асинхронных методов?
- В чем особенность работы ключевых слов async await в приложениях ASP.NET?

Дополнительное задание

Задание
Создайте приложение по шаблону WPF Application. Переместите из элементов управления (ToolBox) на форму текстовое поле и кнопку. Создайте асинхронный обработчик события по нажатию на кнопку. Создайте метод Addition, который принимает два параметра и возвращает результат их сложениях. Из асинхронного обработчика события вызовите этот метод через Task.Run<int>. На возвращаемом значении метода Run вызовите метод ConfigureAwait с указанием параметра false. Примените оператор await к этому выражению и примите результат работы задачи в целочисленную переменную. Ее значение выведите в текстовое поле.

Посмотрите на результаты работы.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные конструкции и понятия, рассмотренные на уроке.

Задание 2
Создайте приложение по шаблону Console Application. Создайте свой контекст синхронизации и переопределите метод Post. Метод Post должен выполнять полученный делегат в контексте потока (экземпляр класса Thread). Дайте потоку, созданному для выполнения делегата в методе Post, имя. Установите в начале работы метода Main созданный контекст синхронизации. Создайте асинхронный метод, который в контексте задачи считает факториал числа через цикл. Используйте оператор await для ожидания завершения этой задачи. До и после оператора await выведите на экран консоли в каком id потока (ManagedThreadId) выполняется эта часть, имя у потока (Name) и является ли поток потоком из пула (IsThreadPoolThread).

Задание 3
Создайте приложение по шаблону Console Application. Возьмите предыдущий пример (Задание 2), не убирая/не изменяя контекст синхронизации выполните продолжение оператора await в контексте рабочего потока пула потоков.

Задание 4
Создайте приложение по шаблону Console Application. Создайте контекст синхронизации, который сможет обрабатывать ошибки, возникшие в асинхронных методах с возвращаемым типом void. Установите созданный контекст синхронизации и проверьте вызовом асинхронного

CyberBionic systematics
CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

метода с типом void, обрабатывается ли ваша ошибка в контексте. Уберите установку контекста синхронизации. Сделайте выводы на счет использования асинхронных методов с типом void.

Задание 5
Создайте приложение по шаблону ASP.NET MVC Application. Создайте веб-сервис с контроллером News. Сделайте GET метод, который отдает экземпляр News из контроллера. При этом, перед возвратом News сделайте задержку в 3 секунды. В веб приложении, создайте HomeController с асинхронным методом действия Index, который асинхронно запрашивает данные из веб-сервиса. В представлении Index выведите информацию полученую от веб-сервиса.

В классе News сделайте 3 автосвойства с названием новости, ее описанием и датой происшествия.

Рекомендуемые ресурсы

MSDN: SynchronizationContext
https://docs.microsoft.com/en-us/dotnet/api/system.threading.synchronizationcontext?view=netframework-4.8

MSDN: TaskScheduler
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskscheduler?view=netframework-4.8

MSDN: ExecutionContext
https://docs.microsoft.com/en-us/dotnet/api/system.threading.executioncontext?view=netframework-4.8

Developer's blog PFX team: SynchronizationContext vs ExecutionContext
https://devblogs.microsoft.com/pfxteam/executioncontext-vs-synchronizationcontext/

MSDN: async void
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/async-return-types#BKMK_VoidReturnType

MSDN: async lambdas
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions#async-lambdas

MSDN: ASP.NET 4.5 asynchronous methods
https://docs.microsoft.com/en-us/aspnet/web-forms/overview/performance-and-caching/using-asynchronous-methods-in-aspnet-45

CyberBionic systematics
CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com