Асинхронный шаблон программирования
Task-based Asynchronous Pattern

№ урока: 2 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Урок познакомит вас с работой библиотеки TPL – Task Parallel Library. Вы подробно ознакомитесь с работой класса Task/Task<TResult>, который лежит в основе асинхронного программирования. На уроке будет рассмотрено большинство свойств и методов классов Task и Task<TResult>. Также в этом уроке рассматривается нововведения в библиотеке, а именно структура ValueTask/ValueTask<TResult>.

Изучив материал данного занятия, учащийся сможет:

- Использовать класс Task/Task<TResult>.
- Разбираться в различных видах настройки задач.
- Использовать метод Task.Run() с лямбдой.
- Понимать работу большинства свойств и методов класса Task/Task<TResult>.
- Создавать продолжения для задач.
- Использовать фабрику задач.
- Использовать структуру ValueTask/ValueTask<TResult>.
- Понимать различия между Task/Task<TResult> и ValueTask/ValueTask<TResult>.

Содержание урока

1. Рассмотрение библиотеки TPL
2. Рассмотрение класса Task
3. Способы создания экземпляра класса Task
4. Рассмотрение настроек задач
5. Продолжения задач
6. Фабрика задач
7. Рассмотрение структуры ValueTask
8. Рассмотрение различий между Task/Task<TResult> и ValueTask/ValueTask<TResult>
9. Исправление синхронных примеров первого урока

Резюме

- TAP – Task-based Asynchronous Pattern. Шаблон асинхронного программирования, в основе которого лежит задача (Task).
- TPL – Task Parallel Library. Библиотека параллельных задач. Появилась с .NET Framework 4.0 и получила обновление в .NET Framework 4.5. Содержит классы для параллельного и асинхронного выполнения кода в языке C#.
- Task – класс, объектно-ориентированное представление задачи. Задача – конструкция, которая реализует модель параллельной обработки, основанной на обещаниях (Promise).
- Task<TResult> - класс, наследуется от базового класса Task. В отличие от родителя имеет возможность удобно вернуть возвращаемое значение асинхронной операции.
- TaskStatus – перечисление, с помощью которого можно отслеживать состояние задачи, в котором она находится. Возможно побитовое сложение флагов перечисления.

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

- TaskCreationOptions – перечисление, с помощью которого можно задать дополнительные параметры для выполнения задачи. Возможно побитовое сложение флагов перечисления.
- Методы ожидания – Wait(), WaitAll(), WaitAny() позволяют подождать выполнения указанных вами задач.
- Метод Task.Run() – добавлен в .NET Framework 4.5 и служит для быстрого создания горячих задач. Замена метода из фабрики задач StartNew().
- Метод RunSynchronously() – выполняет метод, сообщенный с задачей синхронно.
- Continuation – продолжение задачи. Представляет собой метод настройки задачи с указанием, что после своего завершения задача должна продолжится и выполнить указанный ей метод. Некий вариант так называемых Callback методов.
- TaskContinuationOptions – перечисление, с помощью которого можно задать дополнительные параметры для выполнения продолжений, связанных с задачей. Возможно побитовое сложение флагов перечисления.
- TaskFactory – класс, который представляет фабрику задач. Фабрика задач – механизм, который позволяет настроить набор сгруппированных задач, которые находятся в одном состоянии.
- Можно создать экземпляр класса TaskFactory и настроить его нужными параметрами для создания экземпляров класса Task с этими параметрами. Удобность применения TaskFactory состоит в отсутствии необходимости указания этих параметров при каждом создании экземпляров класса Task.
- ValueTask – структура, которая представляет собой обертку над задачей. Создан для уменьшения потребления ресурсов кучи. В некоторых случаях считается сомнительной оптимизацией.
- В большинстве случаев лучше использовать Task или Task<TResult> и рассматривать использование ValueTask или ValueTask<TResult> только в том случае, когда профилирование кода с помощью инструмента анализа производительности показывает, что выделения памяти, связанные с типом Task, являются проблемой для вашего приложения. ValueTask сможет просто вам просигнализировать, что операция завершена или, если используется ValueTask<TResult>, вернуть значение из асинхронной операции, которая все-таки завершилась синхронно.
- Метод AsTask() возвращает из обертки ValueTask задачу, которая была помещена в него. Нужен в случае, если асинхронная операция все-таки произошла, и она была завернута в структуру ValueTask.

Закрепление материала

- Какой основной класс лежит в основе шаблона TAP?
- Task<TResult> является наследником класса Task?
- Сколько существует способов создания экземпляра класса Task?
- Отличия между холодной и горячей задачами?
- Что такое продолжение задачи?
- Какая основная цель создания фабрики задач?
- TaskFactory<TResult> является наследником TaskFactory?
- Для чего используют структуру ValueTask?
- Можно ли полностью отказаться от класса Task, в пользу структуры ValueTask?

Дополнительное задание

Задание
Создайте проект по шаблону "WPF". Переместите из элементов управления (ToolBox) на форму текстовое поле (TextBox) и кнопку (Button). Дайте имена для ваших элементов управления, чтобы к ним можно было обращаться из кода. Например, текстовое поле – txtResult, а кнопка – btnStart.

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

Создайте и зарегистрируйте обработчик события по нажатию на кнопку btnStart. Он должен в цикле выводить в текстовое поле звездочки с задержкой в 300 миллисекунд. В текстовое поле должно быть выведено 100 звездочек. Чтобы форма не зависла на время вывода звездочек и могла отвечать на действия пользователя реализуйте выполнение с помощью задач.

Самостоятельная деятельность учащегося

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

Задание 2
Создайте проект по шаблону "Console Application". Создайте метод «private static void WriteChar(object symbol)». В теле метода создайте цикл for, размерностью 160 итераций, который в своем теле с задержкой в пол секунды выводит на экран консоли значение параметра symbol, приведенного к типу char. Вызовите метод WriteChar из метода Main в контексте задачи, передавая в качестве параметра значение "!". Все время, пока метод WriteChar выполняется, из метода Main выводите на экран консоли "$". Когда задача закончит свое выполнение выведите на экран консоли строку "Метод Main закончил свою работу".

Задание 3
Создайте проект по шаблону "Console Application". Создайте метод «private static int[] SortArray(bool isAscending, params int[] array)». Метод должен отсортировать массив и вернуть отсортированный массив в виде результата. Если параметр isAscending равен true - сортировать по возрастанию, если false - сортировать по убыванию. Как организовать алгоритм сортировки, полностью зависит от вашего выбора. Вызвать метод SortArray в контексте задачи для большого массива типа int. Результат задачи обработать в продолжении, где нужно вывести на экран консоли все элементы массива через запятую.

Задание 4
Создайте проект по шаблону "Console Application". Создайте метод с именем «private static double FindLastFibonacciNumber(int number)». Метод должен найти и вернуть последнее число из последовательности Фибоначчи. Для нахождения последнего числа из последовательности Фибоначчи в тело метода вставить следующий код:

```
Func<int, double> fib = null;
fib = (x) => x > 1 ? fib(x - 1) + fib(x - 2) : x;
return fib.Invoke(number);
```

Даже, если вы считаете, что этот код недостаточно оптимизирован, все равно нужно использовать его. В этом и смысл, что с помощью такого решения, последовательность числа Фибоначчи будет находится намного дольше и с более сильной затратой ресурсов. Поэтому, вам нужно вызвать из метода Main этот метод в контексте задачи. Но так как эта операция займет много времени, вам нужно использовать флаг TaskCreationOptions.LongRunning, чтобы задача выполнялась в контексте потока выполнения Thread и не занимала потоки из пула. Результат асинхронной задачи необходимо вывести на экран консоли. Сделайте это с помощью продолжения.

Рекомендуемые ресурсы

MSDN: Task-based Asynchronous Pattern
https://docs.microsoft.com/ru-ru/dotnet/standard/asynchronous-programming-patterns/task-based-asynchronous-pattern-tap

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

MSDN: Task Parallel Library
https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl

MSDN: Task
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task?view=netframework-4.7.2

MSDN: Task<TResult>
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task-1?view=netframework-4.7.2

MSDN: TaskStatus
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskstatus?view=netframework-4.7.2

MSDN: TaskCreationOptions
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskcreationoptions?view=netframework-4.7.2

MSDN: Continuations
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.continuewith?view=netframework-4.7.2

MSDN: TaskContinuationOptions
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskcontinuationoptions?view=netframework-4.7.2

MSDN: TaskFactory
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskfactory?view=netframework-4.7.2

MSDN: TaskFactory<TResult>
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskfactory-1?view=netframework-4.7.2

MSDN: ValueTask
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.valuetask?view=netcore-2.2

MSDN: ValueTask<TResult>
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.valuetask-1?view=netcore-2.2

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