Планировщик задач. Дочерние задачи.

№ урока: 3 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Цель данного урока в том, чтобы познакомить студента с работой планировщика задач. Мы подробно рассмотрим класс TaskScheduler и рассмотрим несколько популярных реализаций этого абстрактного класса. Задача второй части урока состоит в разборе дочерних и вложенных задач, для понимания различий этих типов задач, сказывающихся на их поведении.

Изучив материал данного занятия, учащийся сможет:

- Понимать назначение методов и свойств класса TaskScheduler
- Использовать планировщик задач
- Создавать свои планировщики задач
- Создавать дочерние задачи
- Создавать вложенные задачи
- Уметь различать дочерние и вложенные задачи

Содержание урока

1. Планировщик задач
2. Разбор свойств и методов планировщика
3. Примеры популярных планировщиков задач
4. Вложенные задачи
5. Дочерние задачи
6. Отличия вложенных и дочерних задач

Резюме

- Планировщик задач – это механизм, который позволяет настроить выполнение задач указанным вами способом и методами.
- Планировщик задач (TaskScheduler) является абстрактным классом. Реализация конкретной логики работы планировщика полностью ложится на программиста-пользователя.
- Стандартный планировщик ThreadPoolTaskScheduler можно получить, вызвав статическое свойство TaskScheduler.Default.
- Абстрактные методы планировщика задач, которые нужно реализовать своими силами:
  - QueueTask(Task task) – помещение переданной в параметрах задачи в очередь выполнения. Через этот метод производится запрос на запуск задачи. В этом методе необходимо решать, как и где будет выполнена ваша задача. Вы сами должны организовать асинхронное или же синхронное выполнение полученой задачи.
  - GetScheduledTasks() – возвращает коллекцию задач, приведенную к базовому интерфейсному типу IEnumerable<Task>.
  - TryExecuteTaskInline(Task task, bool taskWasPreviouslyQueued) – запрашивает возможность выполнить задачу синхронно.
- Дочерние задания – создание задач и их дальнейшее прикрепление к другой задачи, которая будет считаться родителем. Настройка связи Родитель-Потомок.

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

- Чтобы присоединить задачу к родительской, нужно при создании указать флаг перечисления TaskCreationOptions.AttachedToParent.
- Задача может запретить присоединение дочерних задач, указав при создании флаг перечисления TaskCreationOptions.DenyChildAttach.
- Вложенная задача – создание задачи в теле другой задачи.
- Родительская задача ожидает завершения дочерней.
- Внешняя задача НЕ ожидает завершения вложенной.
- Состояние родительской задачи зависит от состояния дочерней задачи.
- Состояние внешней задачи НЕ зависит от состояния вложенной задачи.
- Родительская задача передает исключения дочерней задаче.
- Внешняя задача НЕ передает исключения вложенных задач.

Закрепление материала

- Что такое планировщик задач?
- Какие методы или свойства нужно реализовать в своем планировщике задач, наследуясь от абстрактного класса TaskScheduler?
- Какой тип планировщика содержится в свойстве TaskScheduler.Default?
- Зачем нужен планировщик задач?
- Что такое дочерняя задача?
- Что такое вложенная задача?
- В чем отличия дочерней и вложенной задачи?

Дополнительное задание

Задание
Создайте проект по шаблону "WPF". Переместите из элементов управления (ToolBox) на форму два текстовых поля (TextBox) и кнопку (Button). Дайте имена для ваших элементов управления, чтобы к ним можно было обращаться из кода. Например, текстовое поле 1 – txtResult, текстовое поле 2 - txtLoop, а кнопка – btnStart.

Перенесите в это приложение метод FindLastFibonacciNumber из домашнего задания #4 предыдущего урока. Создайте и зарегистрируйте обработчик события по нажатию на кнопку btnStart. Он должен создать и запустить задачу, которая будет выполнять метод FindLastFibonacciNumber. 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 в текстовое поле txtResult. Сделайте это с помощью продолжения.

Помните, что к элементам управления можно обращаться только из потоков, в которых они были созданы. Поэтому выполните продолжение с помощью планировщика задач SynchronizationContextTaskScheduler. Его можно получить из статического метода TaskScheduler.FromCurrentSynchronizationContext().

Самостоятельная деятельность учащегося

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

Задание 2
Создайте проект по шаблону "Console Application". Создайте свой планировщик задач, производный от класса TaskScheduler с названием StackTaskScheduler. Ваш планировщик будет

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

выполнять первоочередно только поступившие задачи, то есть самые «свежие». Поэтому, внутри него используйте для хранения задач коллекцию Stack<T>. Реализуйте добавление задачи при запуске в вашу коллекцию.

Также, вам необходимо создать метод, который будет перебирать коллекцию задач и изымать задачи на выполнение.

Создайте коллекцию задач из 40 задач. Каждая из задач должна вывести на экран консоли, что она выполнилась и свой порядковый номер при запуске. Запустите все задачи в цикле с вашим планировщиком. Посмотрите на результат работы.

Задание 3
Создайте проект по шаблону "Console Application". Создайте свой планировщик задач, производный от класса TaskScheduler с названием DelayTaskScheduler. Ваш планировщик будет выполнять задачи с задержкой в 2 секунды. То есть, при запуске задачи, она должна подождать 2 секунды прежде, чем запустится. Для решения такой ситуации можно воспользоваться классом Timer или методом ThreadPool.RegisterWaitForSingleObject(), которые позволят вам выполнить вашу задачу в контексте пула потоков, но при этом с задержкой, указанной вами.

Не забудьте заглушить абстрактный метод TryExecuteTaskInline (необходимо просто всегда возвращать false). Из-за задержки в выполнении, задача может часто пытаться выполнится синхронно к потоку вызова.

Создайте задачу, которая выведет на экран консоли в каком потоке она отработала и являлся ли он потоком из пула потоков (для этого используйте свойство - Thread.CurrentThread.IsThreadPoolThread). После этого запустите задачу в контексте вашего планировщика DelayTaskScheduler. После создайте цикл while и, при условии, что свойство IsCompleted вашей задачи возвращает false, выводите на экран консоли звездочку с задержкой в 100 миллисекунд. Код примерно такой:

```
while (task.IsCompleted == false)
{
   Console.Write($"* ");
   Thread.Sleep(100);
}
```

Если вы указали, что задачи должны выполнятся с задержкой в 2 секунды, то у вас должно быть выведено на экран консоли 20 звездочек.

Рекомендуемые ресурсы

MSDN: Планировщик задач
https://docs.microsoft.com/ru-ru/dotnet/api/system.threading.tasks.taskscheduler?view=netframework-4.7.2

MSDN: Планировщик задач использующий пул потоков
https://docs.microsoft.com/ru-ru/dotnet/api/system.threading.tasks.taskscheduler?view=netframework-4.7.2#Default

MSDN: Дочерние и вложенные задачи
https://docs.microsoft.com/ru-ru/dotnet/standard/parallel-programming/attached-and-detached-child-tasks

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

Samples for Parallel Programming with the .NET Framework
https://code.msdn.microsoft.com/ParExtSamples

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