Асинхронная модель программирования

№ урока: 13 **Курс:** C# Professional

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Главная идея асинхронного программирования заключается в том, чтобы запускать отдельные вызовы методов и параллельно продолжать выполнять другую работу без ожидания окончания вызовов. Локальные методы, вероятность исключений которых сведена к минимуму, не нуждаются в асинхронном подходе (например, изменение цвета шрифта текста или его размера), но другие методы (ожидание чтения файла или запуск web-службы) требуют его в самом начале разработки. Среда Common Language Runtime поддерживает достаточно широкое количество методов и классов асинхронного программирования.

Изучив материал данного занятия, учащийся сможет:

- Расширить свои знания о пуле потоков
- Понимать асинхронную природу делегатов
- Понимать механику асинхронных вызовов методов
- Применять callback-методы для эффективной организации работы асинхронной обработки

Содержание урока

1. Пул потоков
2. Шаблон асинхронного вызова методов
3. Асинхронная природа делегатов
4. Интерфейс `IAsyncResult`
5. Синхронизация вызывающего потока
6. Делегат **`AsyncCallback`**
7. Класс `AsyncResult`
8. Передача и прием специальных данных состояния

Резюме

- Классы, в которых есть встроенная поддержка асинхронной модели, имеют пару асинхронных методов для каждого из синхронных методов. Эти методы начинаются со слов `Begin` и `End`. Например, если мы хотим воспользоваться асинхронным вариантом метода `Read` класса `System.IO.Stream`, нам нужно использовать методы `BeginRead` и `EndRead` этого же класса.
- Для использования встроенной поддержки асинхронной модели программирования нужно вызвать соответствующий метод `BeginOperation` и выбрать модель завершения вызова. Вызов метода `BeginOperation` возвращает объект интерфейса `IAsyncResult`, с помощью которого определяется состояние выполнения асинхронной операции.
- Метод `EndOperation` применяется для завершения асинхронного вызова в тех случаях, когда основному потоку необходимо проделать большой объем вычислений, не зависящих от результатов вызова асинхронного метода. После того как основная работа сделана и приложение нуждается в результатах выполнения асинхронного метода для дальнейших действий, вызывается метод `EndOperation`. При этом основной поток будет приостановлен до завершения работы асинхронного метода.
- Способ завершения асинхронного вызова `Callback` используется в тех случаях, когда нужно предотвратить блокирование основного потока. При использовании `Callback` мы запускаем метод `EndOperation` в теле метода, который вызывается при завершении метода, работающего в параллельном потоке. Сигнатура вызываемого метода должна совпадать с сигнатурой делегата **`AsyncCallback`**.
- Вызов асинхронных делегатов позволяет неявно помещать потоки в `ThreadPool`, тем самым избавляя программиста от необходимости работать с ним напрямую.

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- Сигнатура метода `BeginInvoke` не соответствует методу `Invoke`. Это объясняется тем, что нужен некоторый способ идентификации определенного элемента работы, который только что был отложен вызовом `Beginlnvoke`. Таким образом, `Beginlnvoke` возвращает ссылку на объект, реализующий интерфейс `IAsyncResult`. Этот объект подобен cookie-набору, который сохраняется для идентификации выполняющегося элемента работы. Через методы интерфейса `IAsyncResult` можно проверять состояние операции, например, ее готовность.
- Когда поток, запрошенный для выполнения операции, завершит свою работу, он вызывает `Endlnvoke` на делегате. Однако, поскольку метод должен иметь способ идентификации асинхронной операции, результат которой нужно получить, ему должен быть передан объект, полученный из метода `Beginlnvoke`.
- Если в процессе асинхронного выполнения в пуле потоков целевого кода делегата будет сгенерировано исключение, оно сгенерируется повторно, когда инициирующий поток вызовет `EndInvoke`.
- Пул потоков обладает следующими преимуществами:
  - Пул потоков управляет потоками эффективно, уменьшая количество создаваемых, запускаемых и останавливаемых потоков.
  - Используя пул потоков, можно сосредоточиться на решении задачи, а не на инфраструктуре потоков приложения.
- Ситуации, в которых предпочтительно ручное управление потоками:
  - Если нужны потоки переднего плана, или должен быть установлен приоритет потока. Внимание: Потоки из пула всегда являются фоновыми с приоритетом по умолчанию (`ThreadPriority.Normal`).
  - Если требуется поток с фиксированной идентичностью, чтобы можно было прерывать его или находить по имени.
- Интерфейс `IAsyncResult` реализован с помощью классов, содержащих методы, которые могут работать асинхронно. Объект, который обеспечивает работу интерфейса `IAsyncResult`, хранит в себе сведения о состоянии асинхронной операции и предоставляет объект синхронизации, сигнализирующий потоку о завершении операции.
- Для обработки результатов асинхронной операции в отдельном потоке используется делегат **`AsyncCallback`**. Делегат **`AsyncCallback`** представляет метод обратного вызова, который вызывается при завершении асинхронной операции. Метод обратного вызова принимает параметр `IAsyncResult`, который впоследствии используется для получения результатов асинхронной операции.
- Класс `AsyncResult` используется в сочетании с асинхронными вызовами методов с помощью делегатов. `IAsyncResult`, возвращенный из делегатского метода `BeginInvoke`, можно привести к `AsyncResult`. `AsyncResult` имеет свойство `AsyncDelegate`, содержащее объект делегата, к которому был направлен асинхронный вызов.

Закрепление материала

1. В чем заключается разница между синхронным и асинхронным вызовом методов?
2. Объясните механику асинхронных вызовов на уровне пула потоков.
3. Для чего применяется интерфейс `IAsyncResult`?
4. В чем заключается асинхронная природа делегатов?
5. Каково назначение делегата **AsyncCallback**?
6. В чем заключается роль класса AsyncResult?
7. Как организовать передачу/прием специальных данных состояния?

Дополнительное задание

Создайте приложение WindowsForms. На главной форме приложения разместите 3 кнопки с названиями: IsComplete, End, Callback. Организуйте обработчики нажатия на кнопки таким образом, чтобы они инициировали асинхронное выполнение некоторого метода (метод определите сами, можно воспользоваться чем-то вроде Add или более абстрактного Compute). Для каждой из кнопок завершение асинхронного метода должно отслеживаться соответствующим образом:

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- IsComplete – с использованием значения свойства IsComplete
- End – просто применяя EndInvoke
- Callback – с использованием callback метода

Самостоятельная деятельность учащегося

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

Задание 2
Создайте консольное приложение, в котором организуйте асинхронный вызов метода. Используя конструкцию `BeginInvoke` передайте в поток некоторую информацию (возможно, в формате строки). Организуйте обработку переданных данных в callback методе.

Задание 3
Зайдите на сайт MSDN.
Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

MSDN: Интерфес `IAsyncResult`
http://msdn.microsoft.com/ru-ru/library/system.iasyncresult.aspx

MSDN: Делегат **`AsyncCallback`**
http://msdn.microsoft.com/ru-ru/library/system.asynccallback.aspx

MSDN: Класс `AsyncResult`
http://msdn.microsoft.com/ru-ru/library/system.runtime.remoting.messaging.asyncresult.aspx

MSDN: Асинхронное программирование с использованием делегатов
http://msdn.microsoft.com/ru-ru/library/22t547yb.aspx

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com