

.BeginInvoke на делегате

var myDelegate = new **Action**(Method);

myDelegate.BeginInvoke(null, null); - Выполнение метода Method в отдельном потоке, взятом из пула потоков.

1й аргумент ссылка на CallBack метод

2й аргумент в CallBack метод

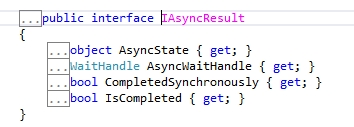
Интерфейс IAsyncResult EndInvoke дождаться завершения метода

ВОЗВРАЩАЕТ ОБЪЕКТ ТИПА class AsyncResult приведенный к ин-су IAsyncResult

IAsyncResult asyncResult = myDelegate.BeginInvoke(null, null); - IAsyncResult - представляет состояние асинхронной операции.

Чтобы дождаться завершения асинхронной операции в основном потоке, следует вызвать на экземпляре делегата метод EndInvoke(передать объект реализующий интерфейс IAsyncResult)

myDelegate.EndInvoke(asyncResult); - Ожидание завершения асинхронной операции.



Передача аргументов и получения результата метода

var myDelegate = new **Func**<int,int,int>(Add);

IAsyncResult asyncResult = myDelegate.BeginInvoke(1, 2, null, null); - Так как класс делегата сообщается с методами, которые принимают два целочисленных параметра, метод BeginInvoke также начинает принимать два дополнительных параметра, кроме двух последних постоянных аргументов.

int result = myDelegate.EndInvoke(asyncResult); - Ожидание завершения асинхронной операции и получение результата работы метода.

// Метод для выполнения в отдельном потоке.

static int Add(int a, int b)

{

Thread.Sleep(2000);

return a + b;

}

Второй способ дождаться завершения метода asyncResult.AsyncWaitHandle.WaitOne();

asyncResult.AsyncWaitHandle.WaitOne(); - AsyncWaitHandle типа WaitHandle, переходит в сигнальное состояние при завершении асинхронной операции.

int result = myDelegate.EndInvoke(asyncResult); - Получение результата асинхронной операции.

Console.WriteLine(asyncResult.AsyncWaitHandle.GetType());

Третий способ дождаться асинхронной опеации

asyncResult.IsCompleted – возвращает true по завершению метода

Выполнение цикла до тех пор, пока работает асинхронная операция.

while (!asyncResult.IsCompleted)

{

Thread.Sleep(100);

Console.Write(".");

}

Метод который будет вызван по завершению операции делегат **AsyncCallback**

var callback = new **AsyncCallback**(HandleCompletion); - Делегат, метод которого будет запущен по завершению асинхронной операции.

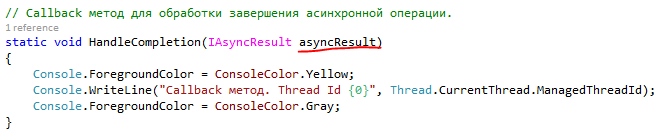
// Первый параметр:

Принимает метод обратного вызова, который должен сработать по завершению асинхронной операции.

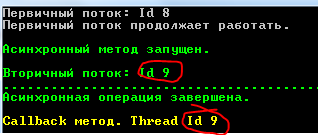
// Второй параметр:

Дополнительный объект хранящий состояние, который будет доступен в методе обратного вызова.

myDelegate.BeginInvoke(callback, null);



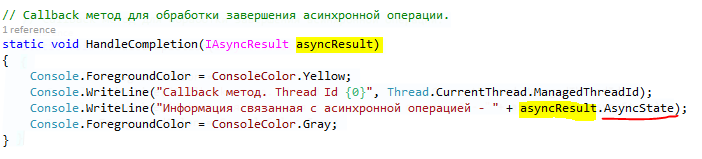
Метод выполняется в том же потоке пула потоков что и основной асинхронный метод



Метод обратного вызова и передача в него пользовательских аргументов .AsyncState

myDelegate.BeginInvoke(callback, "Hello world"); - 2м параметром передаем дополнительный объект в метод обратного вызова

получить пользовательский аргумент в методе обратного вызова можно через объект IAsyncResult asyncResult вызвав свойство asyncResult.AsyncState



Класс AsyncResult

.AsyncDelegate

Получение делегата асинхронного метода (делегат на котором в основном потоке был вызван .BeginInvoke())

// Получение экземпляра делегата, на котором была вызвана асинхронная операция.

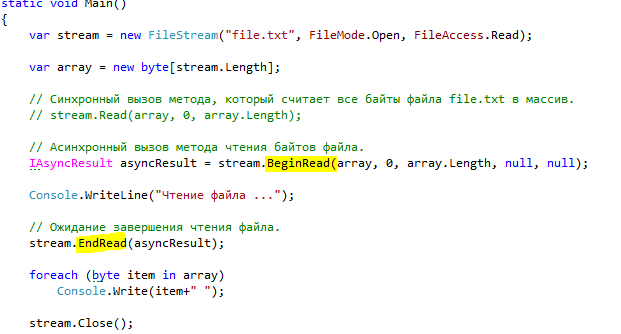
var asyncResult = iAsyncResult as AsyncResult; - привести объект к базовому классу AsyncResult

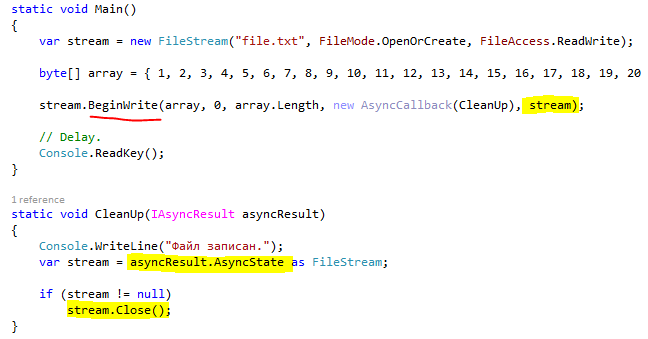
var caller = (**Func**<int,int,int>)asyncResult.AsyncDelegate; - привести его к классу делегат на котором был вызван метод .BeginInvoke()

int sum = caller.EndInvoke(iAsyncResult); - Получение результатов асинхронной операции.

FileStream

.BeginRead .EndRead





МОЖНО ИЗМЕНИТЬ РЕЖИМ IsBackground ПОТОКА

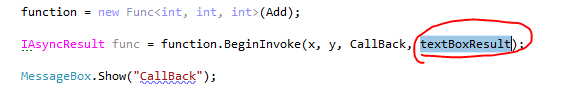
Так как по умолчанию все потоки из пула имеют значение IsBackground true (фоновые)

Thread.CurrentThread.IsBackground = false;

НО ДЕЛАТЬ ЭТОГО НЕ РЕКОМЕНДУЕТЬСЯ

Правильно будет создать обычный поток и в нем производить нужные действия

В WinForms в случае если необходимо использовать какой либо элемент управления, то его необходимо передать в качестве аргумента в CallBack метод либо в метод вызывающийся асинхронно



Но обратиться к нему можно только так

