Microsoft Logo

Gray Pipe

search clear

Recommended Version

IAsyncResult Interface

.NET Framework 1.1

Other Versions

- .NET Framework (current version)
- Visual Studio 2010
- .NET Framework 4
- Silverlight

- .NET Framework 3.5
- Visual Studio 2008
- .NET Framework 3.0
- .NET Framework 2.0

Represents the status of an asynchronous operation.

For a list of all members of this type, see IAsyncResult Members.

[Visual Basic]

Public Interface IAsyncResult

[C#]

public interface IAsyncResult

[C++]

public __gc __interface IAsyncResult

[JScript]

public interface IAsyncResult

Classes that Implement IAsyncResult

Class	Description
AsyncResult	Encapsulates the results of an asynchronous operation on an asynchronous delegate.
WebClientAsyncResult	Provides an implementation of **IAsyncResult** for use by XML Web service proxies to implement the standard asynchronous

	method pattern.

Remarks

The **IAsyncResult** interface is implemented by classes containing methods that can operate asynchronously. It is the return type of methods that initiate an asynchronous operation, such as FileStream.BeginRead, and is the type of the third parameter of methods that conclude an asynchronous operation, such as FileStream.EndRead. **IAsyncResult** objects are also passed to methods invoked by AsyncCallback delegates when an asynchronous operation completes.

An object that supports the **IAsyncResult** interface stores state information for an asynchronous operation, and provides a synchronization object to allow threads to be signaled when the operation completes.

For a detailed description of how the **IAsyncResult** interface is used, see the Asynchronous Programming Overview topic.

Example

[Visual Basic, C#, C++] The following sample demonstrates using an **IAsyncResult** to obtain the return value of an asynchronous operation.

[Visual Basic]

```
' Asynchronous Callback method.
Public Shared Sub MyCallback(ar As IAsyncResult)
   ' Obtains the last parameter of the delegate call.
   Dim value As Integer = Convert.ToInt32(ar.AsyncState)

   ' Obtains return value from the delegate call using EndInvoke.
   Dim aResult As AsyncResult = CType(ar, AsyncResult)
   Dim temp As SampSyncSqrDelegate = CType(aResult.AsyncDelegate,
SampSyncSqrDelegate)
   Dim result As Integer = temp.EndInvoke(ar)

   Console.Write("Simple.SomeMethod (AsyncCallback): Result of ")
   Console.WriteLine("{0} in SampleSynchronized.Square is {1} ", value, result)
End Sub 'MyCallback

[C#]
// Asynchronous Callback method.
```

```
public static void MyCallback(IAsyncResult ar) {

   // Obtains the last parameter of the delegate call.
   int value = Convert.ToInt32(ar.AsyncState);

   // Obtains return value from the delegate call using EndInvoke.
   AsyncResult aResult = (AsyncResult)ar;
   SampSyncSqrDelegate temp = (SampSyncSqrDelegate)aResult.AsyncDelegate;
   int result = temp.EndInvoke(ar);

   Console.Write("Simple.SomeMethod (AsyncCallback): Result of ");
   Console.WriteLine("{0} in SampleSynchronized.Square is {1} ", value, result);
}

[C++]
// Asynchronous Callback method.
public:
static void MyCallback(IAsyncResult* ar)
{
   // Obtains the last parameter of the delegate call.
   int value = Convert::ToInt32(ar->AsyncState);

   // Obtains return value from the delegate call using EndInvoke.
   AsyncResult* aResult = dynamic_cast<AsyncResult*>(ar);
   SampSyncSqrDelegate* temp = static_cast<SampSyncSqrDelegate*>(aResult-
>AsyncDelegate);
   int result = temp->EndInvoke(ar);

   Console::Write(S"Simple::SomeMethod (AsyncCallback): Result of ");
   Console::WriteLine(S" {0} in SampleSynchronized::Square is {1} ", __box(value),
__box(result));
}
```

[Visual Basic, C#, C++] The following sample demonstrates waiting for an asynchronous operation to complete.

```
[Visual Basic]
Imports System
Imports System.Threading
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Contexts
Imports System.Runtime.Remoting.Messaging


'
' Context-Bound type with Synchronization Context Attribute
'
<Synchronization()> Public Class SampleSyncronized
  Inherits ContextBoundObject

  ' A method that does some work - returns the square of the given number
  Public Function Square(i As Integer) As Integer
    Console.Write("SampleSyncronized.Square called.  ")
    Console.WriteLine("The hash of the current thread is: {0}",
Thread.CurrentThread.GetHashCode())
    Return i * i
  End Function 'Square
End Class 'SampleSyncronized



'
' Async delegate used to call a method with this signature asynchronously
'
Delegate Function SampSyncSqrDelegate(i As Integer) As Integer


'Main sample class
Public Class AsyncResultSample

  Public Shared Sub Main()
```

```
    Dim callParameter As Integer = 0
    Dim callResult As Integer = 0

    'Create an instance of a context-bound type SampleSynchronized
    'Because SampleSynchronized is context-bound, the object sampSyncObj
    'is a transparent proxy
    Dim sampSyncObj As New SampleSyncronized()


    'call the method synchronously
    Console.Write("Making a synchronous call on the object.  ")
    Console.WriteLine("The hash of the current thread is: {0}",
Thread.CurrentThread.GetHashCode())
    callParameter = 10
    callResult = sampSyncObj.Square(callParameter)
    Console.WriteLine("Result of calling sampSyncObj.Square with {0} is {1}.",
callParameter, callResult)
    Console.WriteLine("")
    Console.WriteLine("")


    'call the method asynchronously
    Console.Write("Making an asynchronous call on the object.  ")
    Console.WriteLine("The hash of the current thread is: {0}",
Thread.CurrentThread.GetHashCode())
    Dim sampleDelegate As New SampSyncSqrDelegate(AddressOf
sampSyncObj.Square)
    callParameter = 17

    Dim aResult As IAsyncResult = sampleDelegate.BeginInvoke(callParameter,
Nothing, Nothing)

    'Wait for the call to complete
    aResult.AsyncWaitHandle.WaitOne()
```

```
      callResult = sampleDelegate.EndInvoke(aResult)

      Console.WriteLine("Result of calling sampSyncObj.Square with {0} is {1}.",
callParameter, callResult)

   End Sub 'Main

End Class 'AsyncResultSample

[C#]
using System;
using System.Threading;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Contexts;
using System.Runtime.Remoting.Messaging;

//
// Context-Bound type with Synchronization Context Attribute
//
[Synchronization()]
public class SampleSyncronized : ContextBoundObject
{
   // A method that does some work - returns the square of the given number
   public int Square(int i)
   {
      Console.Write("SampleSyncronized.Square called.  ");
      Console.WriteLine("The hash of the current thread is: {0}",
Thread.CurrentThread.GetHashCode());
      return i*i;
   }
}



//
// Async delegate used to call a method with this signature asynchronously
```

```
//
public delegate int SampSyncSqrDelegate(int i);

//Main sample class
public class AsyncResultSample
{
   public static void Main()
   {
      int callParameter = 0;
      int callResult = 0;

      //Create an instance of a context-bound type SampleSynchronized
      //Because SampleSynchronized is context-bound, the object sampSyncObj
      //is a transparent proxy
      SampleSyncronized sampSyncObj = new SampleSyncronized();


      //call the method synchronously
      Console.Write("Making a synchronous call on the object.  ");
      Console.WriteLine("The hash of the current thread is: {0}",
Thread.CurrentThread.GetHashCode());
      callParameter = 10;
      callResult = sampSyncObj.Square(callParameter);
      Console.WriteLine("Result of calling sampSyncObj.Square with {0} is {1}.\n\n",
callParameter, callResult);


      //call the method asynchronously
      Console.Write("Making an asynchronous call on the object.  ");
      Console.WriteLine("The hash of the current thread is: {0}",
Thread.CurrentThread.GetHashCode());
      SampSyncSqrDelegate sampleDelegate = new
SampSyncSqrDelegate(sampSyncObj.Square);
      callParameter = 17;
```

```
        IAsyncResult aResult = sampleDelegate.BeginInvoke(callParameter, null, null);

        //Wait for the call to complete
        aResult.AsyncWaitHandle.WaitOne();

        callResult = sampleDelegate.EndInvoke(aResult);
        Console.WriteLine("Result of calling sampSyncObj.Square with {0} is {1}.",
callParameter, callResult);
    }
}

[C++]
#using <mscorlib.dll>

using namespace System;
using namespace System::Threading;
using namespace System::Runtime::Remoting;
using namespace System::Runtime::Remoting::Contexts;
using namespace System::Runtime::Remoting::Messaging;

//
// Context-Bound type with Synchronization Context Attribute
//
public __gc class SampleSyncronized : public ContextBoundObject
{
    // A method that does some work - returns the square of the given number
public:
    int Square(int i)
    {
        Console::Write(S"SampleSyncronized::Square called.  ");
        Console::WriteLine(S"The hash of the current thread is: {0}",
__box(Thread::CurrentThread->GetHashCode()));
        return i*i;
    }
};
```

```
//
// Async delegate used to call a method with this signature asynchronously
//
__delegate int SampSyncSqrDelegate(int i);

//Main sample class
int main()
{
    int callParameter = 0;
    int callResult = 0;

    //Create an instance of a context-bound type SampleSynchronized
    //Because SampleSynchronized is context-bound, the Object* sampSyncObj
    //is a transparent proxy
    SampleSyncronized* sampSyncObj = new SampleSyncronized();



    //call the method synchronously
    Console::Write(S"Making a synchronous call on the Object*.  ");
    Console::WriteLine(S"The hash of the current thread is: {0}",
__box(Thread::CurrentThread->GetHashCode()));
    callParameter = 10;
    callResult = sampSyncObj->Square(callParameter);
    Console::WriteLine(S"Result of calling sampSyncObj.Square with {0} is {1}.\n\n",
__box(callParameter), __box(callResult));



    //call the method asynchronously
    Console::Write(S"Making an asynchronous call on the Object*.  ");
    Console::WriteLine(S"The hash of the current thread is: {0}",
__box(Thread::CurrentThread->GetHashCode()));
    SampSyncSqrDelegate* sampleDelegate = new
SampSyncSqrDelegate(sampSyncObj, SampleSyncronized::Square);
    callParameter = 17;
```

```
    IAsyncResult* aResult = sampleDelegate->BeginInvoke(callParameter, 0, 0);

    //Wait for the call to complete
    aResult->AsyncWaitHandle->WaitOne();

    callResult = sampleDelegate->EndInvoke(aResult);
    Console::WriteLine(S"Result of calling sampSyncObj.Square with {0} is {1}.",
__box(callParameter), __box(callResult));
}
```

[JScript] No example is available for JScript. To view a Visual Basic, C#, or C++ example, click the Language Filter button

Language Filter

in the upper-left corner of the page.

Requirements

Namespace: System

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP Home Edition, Windows XP Professional, Windows Server 2003 family, .NET Compact Framework

Assembly: Mscorlib (in Mscorlib.dll)

See Also

IAsyncResult Members | System Namespace

Syntax based on .NET Framework version 1.1.

Documentation version 1.1.1.

Send comments on this topic.

Show: Inherited Protected

Dev centers

- Windows
- Office
- Visual Studio
- Microsoft Azure

- More...

Learning resources

- Microsoft Virtual Academy
- Channel 9
- MSDN Magazine

Community

- Forums
- Blogs
- Codeplex

Support

- Self support

Programs

- BizSpark (for startups)
- Microsoft Imagine (for students)

United States (English)

- Newsletter
- Privacy & cookies
- Terms of use
- Trademarks

logo

© 2017 Microsoft