AsyncPage

<%@ Page Language="C#" Async="true" Trace="true" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="AsyncPages.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Асинхронная страница</title>

</head>

<body>

<form id="form1" runat="server">

<div>

Ссылки с msdn.microsoft.com

<asp:Label ID="Output" runat="server" />

</div>

</form>

</body>

</html>

public partial class Default : System.Web.UI.Page

{

private WebRequest \_request = null;

protected void Page\_Load(object sender, EventArgs e)

{

// Регистрация методов обработчиков начала и завершения асинхронной операции.

AddOnPreRenderCompleteAsync(new BeginEventHandler(BeginAsyncOperation),

new EndEventHandler(EndAsyncOperation));

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> AddOnPreRenderCompleteAsync in Page\_Load");

}

private IAsyncResult BeginAsyncOperation(object sender, EventArgs e, AsyncCallback cb, object state)

{

// Вывод ID потока, который начал выполнение операции. ID потока, который завершит асинхронную операцию будет отличатся.

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> BeginAsyncOperation");

\_request = WebRequest.Create("http://msdn.microsoft.com/");

return \_request.BeginGetResponse(cb, null); // Асинхронный запрос к удаленному ресурсу.

}

private void EndAsyncOperation(IAsyncResult ar)

{

string text;

using (WebResponse response = \_request.EndGetResponse(ar))

{

Stream stream = response.GetResponseStream();

using (StreamReader reader = new StreamReader(stream))

{

text = reader.ReadToEnd();

}

}

Regex regex = new Regex("href=\"(\\S\*)\"");

MatchCollection collection = regex.Matches(text);

StringBuilder builder = new StringBuilder();

foreach (Match match in collection)

{

builder.Append(match.Groups[1]);

builder.Append("<br />");

}

Output.Text += builder.ToString();

// Вывод ID потока завершившего выполнение асинхронной страницы.

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> EndAsyncOperation");

}

protected void Page\_PreRenderComplete(object sender, EventArgs e)

{

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> PreRenderComplete");

}

}

}

Web conf

<configuration>

<system.web>

<compilation debug="true" targetFramework="4.5" />

<httpRuntime targetFramework="4.5" />

</system.web>

</configuration>

\*\*\*

<%@ Page Language="C#" Async="true" Trace="true" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="\_002\_AsyncPage.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Асинхронная страница</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Label runat="server" ID="Output" />

</div>

</form>

</body>

</html>

public partial class Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Trace.Write("Page\_Load " + Thread.CurrentThread.ManagedThreadId.ToString());

// Регистрация асинхронной операции.

AddOnPreRenderCompleteAsync(BeginAsyncOperation, EndAsyncOperation);

}

// Начало асинхронной операции.

private IAsyncResult BeginAsyncOperation(object sender, EventArgs e, AsyncCallback cb, object extraData)

{

Trace.Write("BeginAsyncOperation " + Thread.CurrentThread.ManagedThreadId.ToString());

Func<int, int> del = new Func<int, int>(DoSomeWork);

return del.BeginInvoke(8, cb, null);

}

// Завершение асинхронной операции.

private void EndAsyncOperation(IAsyncResult ar)

{

Trace.Write("EndAsyncOperation " + Thread.CurrentThread.ManagedThreadId.ToString());

AsyncResult arObject = ar as AsyncResult;

// Получение экземпляра делегата на котором производился асинхронный вызов.

Func<int, int> del = (Func<int, int>)arObject.AsyncDelegate;

// Получение результата работы асинхронного метода.

int result = del.EndInvoke(ar);

Output.Text = result.ToString();

}

// Задача, которая выполняется другим потоком (не из пула потоков приложения)

private int DoSomeWork(int arg)

{

Trace.Write("DoSomeWork " + Thread.CurrentThread.ManagedThreadId.ToString());

Thread.Sleep(2000);

return arg + 2;

}

protected void Page\_PreRenderComplete(object sender, EventArgs e)

{

Trace.Write("PreRenderComplete " + Thread.CurrentThread.ManagedThreadId.ToString());

}

}

\*\*\*

PageAsyncTask

<%@ Page Language="C#" Async="true" Trace="true" AsyncTimeout="10" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="\_003\_PageAsyncTasks.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Использование PageAsyncTask</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Label runat="server" ID="OutputLabel" />

</div>

</form>

</body>

</html>

// Для того, что бы класс PageAsyncTask работал в 4.5 framework необходимо добавить следующий фрагмент кода в файл web.config

// <appSettings>

// <add key="aspnet:UseTaskFriendlySynchronizationContext" value="false" />

// </appSettings>

public partial class Default : System.Web.UI.Page

{

private WebRequest \_request = null;

protected void Page\_Load(object sender, EventArgs e)

{

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> AddOnPreRenderCompleteAsync in Page\_Load");

// Асинхронная задача для данной страницы.

PageAsyncTask task = new PageAsyncTask(

OnBegin, // Начало асинхронной операции.

OnEnd, // Завершение асинхронной операции.

OnTimeOut, // Действие в случае если вышло время ожидания указанное в атрибуте AsyncTimeout директивы @ Page.

null // Значение для аргумента extraData метода OnBegin.

);

// Регистрация задачи. По умолчанию, вызов задачи происходит после события PreRender.

RegisterAsyncTask(task);

}

private IAsyncResult OnBegin(object sender, EventArgs e, AsyncCallback cb, object extraData)

{

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> BeginAsyncOperation");

\_request = WebRequest.Create("http://msdn.microsoft.com/");

return \_request.BeginGetResponse(cb, extraData); // Асинхронный запрос к удаленному ресурсу.

}

private void OnEnd(IAsyncResult ar)

{

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> EndAsyncOperation");

string text;

using (WebResponse response = \_request.EndGetResponse(ar))

{

using (StreamReader reader = new StreamReader(response.GetResponseStream()))

{

text = reader.ReadToEnd();

}

}

Regex regex = new Regex("href=\"(\\S\*)\"");

MatchCollection collection = regex.Matches(text);

StringBuilder builder = new StringBuilder();

foreach (Match match in collection)

{

builder.Append(match.Groups[1]);

builder.Append("<br />");

}

OutputLabel.Text += builder.ToString();

}

private void OnTimeOut(IAsyncResult ar)

{

OutputLabel.Text = "Превышено время ожидания ответа.";

}

protected void Page\_PreRenderComplete(object sender, EventArgs e)

{

Trace.Write("Thread = " + Thread.CurrentThread.ManagedThreadId.ToString(), "-> PreRenderComplete");

}

}

}

Web config

<configuration>

<appSettings>

<add key="aspnet:UseTaskFriendlySynchronizationContext" value="false" />

</appSettings>

<system.web>

<compilation debug="true" targetFramework="4.5" />

<httpRuntime targetFramework="4.5" />

</system.web>

</configuration>

\*\*\*

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="\_004\_PageAsyncTasks.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>PageAsyncTask</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Label ID="OutputLabel" runat="server"></asp:Label>

</div>

</form>

</body>

</html>

public partial class Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

SlowTask slowTask1 = new SlowTask();

SlowTask slowTask2 = new SlowTask();

SlowTask slowTask3 = new SlowTask();

// Последний параметр:

// true - задачи выполняются параллельно.

// false - задачи выполняются последовательно.

PageAsyncTask task1 = new PageAsyncTask(slowTask1.OnBegin, slowTask1.OnEnd, slowTask1.OnTimeOut, null, false);

PageAsyncTask task2 = new PageAsyncTask(slowTask2.OnBegin, slowTask2.OnEnd, slowTask2.OnTimeOut, null, false);

PageAsyncTask task3 = new PageAsyncTask(slowTask3.OnBegin, slowTask3.OnEnd, slowTask3.OnTimeOut, null, false);

RegisterAsyncTask(task1);

RegisterAsyncTask(task2);

RegisterAsyncTask(task3);

// Запуск асинхронных операций на выполнение.

Page.ExecuteRegisteredAsyncTasks();

OutputLabel.Text = slowTask1.GetData() + "<br />" + slowTask2.GetData() + "<br />" + slowTask3.GetData();

}

}

Cs class

public class SlowTask

{

private delegate void AsyncDelegate();

AsyncDelegate \_delegate;

string \_data;

// Метод по сигнатуре подходит под делегат BeginEventHandler

public IAsyncResult OnBegin(object sender, EventArgs e, AsyncCallback cb, object extraData)

{

\_delegate = ExecuteAsyncTask;

return \_delegate.BeginInvoke(cb, extraData);

}

// Метод по сигнатуре подходит под делегат EndEventHandler

public void OnEnd(IAsyncResult ar)

{

\_delegate.EndInvoke(ar);

\_data += "Задача завершена " + DateTime.Now.Ticks.ToString() + " ";

}

// Метод по сигнатуре подходит под делегат EndEventHandler

public void OnTimeOut(IAsyncResult ar)

{

\_data = "Вышло время ожидания завершения задачи";

}

// Метод для получения результата.

public string GetData()

{

return \_data;

}

// Метод, который будет выполнятся в отдельном потоке.

private void ExecuteAsyncTask()

{

\_data += "Задача запущена " + DateTime.Now.Ticks.ToString() + " ";

Thread.Sleep(2000);

}

}

Web config

?xml version="1.0"?>

<configuration>

<appSettings>

<add key="aspnet:UseTaskFriendlySynchronizationContext" value="false" />

</appSettings>

<system.web>

<compilation debug="true" targetFramework="4.5"/>

<httpRuntime/>

</system.web>

</configuration>