عنوان: **استفاده از async و await در برنامه‌های ASP.NET Web forms 4.5**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۱۲ ۱۲:۳۰
آدرس: www.dotnettips.info
گروه‌ها: C#, Asynchronous Programming

سؤال: چه زمانی از متدهای async و چه زمانی از متدهای همزمان بهتر است استفاده شود؟

از متدهای همزمان متداول برای انجام امور ذیل استفاده نمائید:
- جهت پردازش اعمالی ساده و سریع
- اعمال مدنظر بیشتر قرار است بر روی CPU اجرا شوند و از مرزهای IO سیستم عبور نمی‌کنند.

و از متدهای غیرهمزمان برای پردازش موارد زیر کمک بگیرید:
- از وب سرویس‌هایی استفاده می‌کنید که متدهای نگارش async را نیز ارائه داده‌اند.
- عمل مدنظر network-bound و یا I/O-bound است بجای CPU-bound. یعنی از مرزهای IO سیستم عبور می‌کند.
- نیاز است چندین عملیات را به موازات هم اجرا کرد.
- نیاز است مکانیزمی را جهت لغو یک عملیات طولانی ارائه دهید.

مزایای استفاده از متدهای async در ASP.NET

استفاده از await در ASP.NET، ساختار ذاتی پروتکل HTTP را که اساسا یک synchronous protocol، تغییر نمی‌دهد. کلاینت، درخواستی را ارسال می‌کند و باید تا زمان آماده شدن نتیجه و بازگشت آن از طرف سرور، صبر کند. نحوه‌ی تهیه‌ی این نتیجه، خواه async باشد و یا حتی همزمان، از دید مصرف کننده کاملا مخفی است. اکنون سؤال اینجا است که چرا باید از متدهای async استفاده کرد؟
- **پردازش موازی:** می‌توان چند Task را مثلا توسط Task.WhenAll به صورت موازی با هم پردازش کرده و در نهایت نتیجه را سریعتر به مصرف کننده بازگشت داد. اما باید دقت داشت که این Taskها اگر I/O bound باشند، ارزش پردازش موازی را دارند و اگر compute bound باشند (اعمال محاسباتی)، صرفا یک سری ترد را ایجاد و مصرف کرده‌اید که می‌توانسته‌اند به سایر درخواست‌های رسیده پاسخ دهند.
- **خالی کردن تردهای در حال انتظار:** در اعمالی که disk I/O و یا network I/O دارند، پردازش موازی و اعمال async به شدت مقیاس پذیری سیستم را بالا می‌برند. به این ترتیب worker thread جاری (که تعداد آن‌ها محدود است)، سریعتر آزاد شده و به worker pool بازگشت داده می‌شود تا بتواند به یک درخواست دیگر رسیده سرویس دهد. در این حالت می‌توان با منابع کمتری، درخواست‌های بیشتری را پردازش کرد.

ایجاد Asynchronous HTTP Handlers در ASP.Net 4.5

در نگارش‌های پیش از دات نت 4.5، برای نوشتن فایل‌های ashx غیرهمزمان می‌بایستی اینترفیس IHttpAsynchHandler پیاده سازی می‌شد که نحوه‌ی کار با آن از مدل APM پیروی می‌کرد؛ نیاز به استفاده از یک سری callback داشت و این عملیات باید طی دو متد پردازش می‌شد. اما در دات نت 4.5 و با معرفی امکانات async و await، نگارش سازگاری با پیاده سازی کلاس پایه HttpTaskAsyncHandler فراهم شده است.
برای آزمایش آن، یک برنامه‌ی جدید ASP.NET Web forms نگارش 4.5 یا بالاتر را ایجاد کنید. سپس از منوی پروژه، گزینه‌ی Add new item یک Generic handler به نام LogRequestHandler.ashx را به پروژه اضافه نمائید.
زمانیکه این فایل به پروژه اضافه می‌شود، یک چنین امضایی را دارد:

```
public class LogRequestHandler : IHttpHandler
```

IHttpHandler آن‌را اکنون به HttpTaskAsyncHandler تغییر دهید. سپس پیاده سازی ابتدایی آن به شکل زیر خواهد بود:

```
using System;
```

```
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Web;

namespace Async14
{
    public class LogRequestHandler : HttpTaskAsyncHandler
    {
        public override async Task ProcessRequestAsync(HttpContext context)
        {
            string url = context.Request.QueryString["rssfeedURL"];
            if (string.IsNullOrWhiteSpace(url))
            {
                context.Response.Write("Rss feed URL is not provided");
            }

            using (var webClient = new WebClient {Encoding = Encoding.UTF8})
            {
                webClient.Headers.Add("User-Agent", "LogRequestHandler 1.0");
                var rssfeed = await webClient.DownloadStringTaskAsync(url);
                context.Response.Write(rssfeed);
            }
        }

        public override bool IsReusable
        {
            get { return true; }
        }

        public override void ProcessRequest(HttpContext context)
        {
            throw new Exception("The ProcessRequest method has no implementation.");
        }
    }
}
```

واژه‌ی کلیدی async را نیز جهت استفاده از await به نسخه‌ی غیرهمزمان آن اضافه کرده‌ایم.
در این مثال آدرس یک فید RSS از طریق کوئری استرینگ rssfeedURL دریافت شده و سپس محتوای آن به کمک متد DownloadStringTaskAsync دریافت و بازگشت داده می‌شود.
برای آزمایش آن، مسیر ذیل را درخواست دهید:

```
http://localhost:4207/LogRequestHandler.ashx?rssfeedURL=http://www.dotnettips.info/feed/latestchanges
```

کاربردهای فایل‌های ashx برای مثال ارائه فیدهای XML ایی یک سایت، ارائه منبع نمایش تصاویر پویا از بانک اطلاعاتی، ارائه JSON برای افزونه‌های auto complete جی‌کوئری و امثال آن است. مزیت آن‌ها سربار بسیار کم است؛ زیرا وارد چرخه‌ی طول عمر یک صفحه‌ی aspx معمولی نمی‌شوند.

صفحات async در ASP.NET 4.5

در قسمت‌های قبل مشاهده کردیم که در برنامه‌های دسکتاپ، به سادگی می‌توان امضای روال‌های رخداد گردان را به async تغییر داد و ... برنامه کار می‌کند. به علاوه از مزیت استفاده از واژه کلیدی await نیز در آن‌ها برخوردار خواهیم شد. اما ... هرچند این روش در وب فرم‌ها نیز صادق است (مثلا public void Page_Load را به public async void Page_Load می‌توان تبدیل کرد) اما اعضای تیم ASP.NET آن‌را در مورد برنامه‌های وب فرم توصیه نمی‌کنند:
Async void event handlers تنها در مورد تعداد کمی از روال‌های رخدادگردان ASP.NET Web forms کار می‌کنند و از آن‌ها تنها برای تدارک پردازش‌های ساده می‌توان استفاده کرد. اگر کار در حال انجام اندکی پیچیدگی دارد، «باید» از PageAsyncTask استفاده نمائید. علت اینجا است که Async void یعنی fire and forget (کاری را شروع کرده و فراموشش کنید). این روش در برنامه‌های دسکتاپ کار می‌کند، زیرا این برنامه‌ها مدل طول عمر متفاوتی داشته و تا زمانیکه برنامه از طرف OS خاتمه نیابد، مشکلی نخواهند داشت. اما برنامه‌های بدون حالت وب متفاوتند. اگر عملیات async پس از خاتمه‌ی طول عمر صفحه پایان یابد، دیگر نمی‌توان اطلاعات صحیحی را به کاربر ارائه داد. بنابراین تا حد ممکن از تعاریف async void در برنامه‌های وب خودداری کنید.

تبدیل روال‌های رخدادگردان متداول وب فرم‌ها به نسخه‌ی async شامل دو مرحله است:

الف) از متد جدید RegisterAsyncTask که در کلاس پایه Page قرار دارد برای تعریف یک PageAsyncTask استفاده کنید:

```
using System;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Web.UI;

namespace Async14
{
    public partial class _default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            RegisterAsyncTask(new PageAsyncTask(LoadSomeData));
        }

        public async Task LoadSomeData()
        {
            using (var webClient = new WebClient { Encoding = Encoding.UTF8 })
            {
                webClient.Headers.Add("User-Agent", "LogRequest 1.0");
                var rssfeed = await webClient.DownloadStringTaskAsync("url");

                //listcontacts.DataSource = rssfeed;
            }
        }
    }
}
```

با استفاده از System.Web.UI.PageAsyncTask می‌توان یک async Task را در روال‌های رخدادگردان ASP.NET مورد استفاده قرار داد.

ب) سپس در کدهای فایل aspx، نیاز است خاصیت async را نیز true نمائید:

```
<%@ Page Language="C#" AutoEventWireup="true"
Async="true"
  CodeBehind="default.aspx.cs" Inherits="Async14._default" %>
```

تغییر تنظیمات IIS برای بهره بردن از پردازش‌های Async

اگر از ویندوزهای 7، ویستا و یا 8 استفاده می‌کنید، IIS آن‌ها به صورت پیش فرض به 10 درخواست همزمان محدود است.
بنابراین تنظیمات ذیل مرتبط است به یک ویندوز سرور و نه یک work station :
به IIS manager مراجعه کنید. سپس برگه‌ی Application Pools آن‌را باز کرده و بر روی Application pool برنامه خود کلیک راست نمائید. در اینجا گزینه‌ی Advanced Settings را انتخاب کنید. در آن Queue Length را به مثلا عدد 5000 تغییر دهید. همچنین در دات نت 4.5 عدد 5000 برای MaxConcurrentRequestsPerCPU نیز مناسب است. به علاوه عدد
connectionManagement/maxconnection را نیز به 12 برابر تعداد هسته‌های موجود تغییر دهید.