عنوان: **جلوگیری از deadlock در برنامه‌های async**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۱۱ ۱۵:۹
آدرس: www.dotnettips.info
گروه‌ها: C#, Asynchronous Programming

توضیح مطلب جاری نیاز به یک مثال دارد. به همین جهت یک برنامه‌ی WinForms یا WPF را آغاز کنید (تفاوتی نمی‌کند). سپس یک دکمه و یک برچسب را در صفحه قرار دهید. در ادامه کدهای فرم را به نحو ذیل تغییر دهید.

```
using System;
using System.Net.Http;
using System.Threading.Tasks;
using System.Windows.Forms;
using Newtonsoft.Json.Linq;

namespace Async13
{
    public static class JsonExt
    {
        public static async Task<JObject> GetJsonAsync(this Uri uri)
        {
            using (var client = new HttpClient())
            {
                var jsonString = await client.GetStringAsync(uri);
                return JObject.Parse(jsonString);
            }
        }
    }

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnGo_Click(object sender, EventArgs e)
        {
            var url =
                "http://api.geonames.org/citiesJSON?north=44.1&south=-9.9&east=-
22.4&west=55.2&lang=de&username=demo";
            txtResult.Text = new Uri(url).GetJsonAsync().Result.ToString();
        }
    }
}
```

این کدها برای کامپایل نیاز به نصب بسته‌ی

```
PM> Install-Package Newtonsoft.Json
```

و همچنین افزودن ارجاعی به اسمبلی استاندارد System.Net.Http نیز دارند.
در اینجا قصد داریم اطلاعات JSON دریافتی را در یک TextBox نمایش دهیم. کاری که انجام شده، فراخوانی متد async ایی است به نام GetJsonAsync و سپس استفاده از خاصیت Result این Task برای صبر کردن تا پایان عملیات.
اگر برنامه را اجرا کنید و بر روی دکمه‌ی دریافت اطلاعات کلیک نمائید، برنامه قفل خواهد کرد. چرا؟
البته تفاوتی هم نمی‌کند که این یک برنامه‌ی دسکتاپ است یا یک برنامه‌ی وب. در هر دو حالت یک deadlock کامل را مشاهده خواهید کرد.

علت بروز deadlock در کدهای async چیست؟

همواره نتیجه‌ی await، در context فراخوان آن بازگشت داده می‌شود. اگر برنامه‌ی دسکتاپ است، این context همان ترد اصلی UI برنامه می‌باشد و اگر برنامه‌ی وب است، این context، زمینه‌ی درخواست در حال پردازش می‌باشد.
خاصیت Result و یا استفاده از متد Wait یک Task، به صورت همزمان عمل می‌کنند و نه غیرهمزمان. متد GetJsonAsync یک Task ناتمام را که فراخوان آن باید جهت پایان‌اش صبر کند، بازگشت می‌دهد. سپس در همینجا کد فراخوان، تردجاری را توسط

فراخوانی خاصیت Result قفل می‌کند. متد GetJsonAsync منتظر خواهد ایستاد تا این ترد آزاد شده و بتواند به کارش که بازگردان نتیجه‌ی عملیات به context جاری است، ادامه دهد.
به عبارتی، کدهای async منتظر پایان کار Result هستند تا نتیجه را بازگردانند. در همین لحظه کدهای همزمان برنامه نیز منتظر کدهای async هستند تا خاتمه یابند. نتیجه‌ی کار یک deadlock است.

روش‌های جلوگیری از deadlock در کدهای async؟

الف) در مورد متد ConfigureAwait در قسمت‌های قبل بحث شد و به عنوان یک best practice مطرح است:

```
public static class JsonExt
    {
        public static async Task<JObject> GetJsonAsync(this Uri uri)
        {
            using (var client = new HttpClient())
            {
                var jsonString = await
client.GetStringAsync(uri).ConfigureAwait(continueOnCapturedContext: false);
                return JObject.Parse(jsonString);
            }
        }
    }
```

با استفاده از ConfigureAwait false سبب خواهیم شد تا نتیجه‌ی عملیات به context جاری بازگشت داده نشود و نتیجه بر روی thread pool thread ادامه یابد. با اعمال این تغییر، کدهای متد btnGo_Click بدون مشکل اجرا خواهند شد.

ب) راه حل دوم، عدم استفاده از خواص و متدهای همزمان با متدهای غیر همزمان است:

```
private async void btnGo_Click(object sender, EventArgs e)
        {
            var url =
                "http://api.geonames.org/citiesJSON?north=44.1&south=-9.9&east=-
22.4&west=55.2&lang=de&username=demo";
            var data = await new Uri(url).GetJsonAsync();
            txtResult.Text = data.ToString();
        }
```

ابتدا امضای متد رویدادگردان را اندکی تغییر داده و واژه‌ی کلیدی async را به آن اضافه می‌کنیم. سپس از await برای صبر کردن تا پایان عملیات متد GetJsonAsync استفاده خواهیم کرد. صبر کردنی که در اینجا انجام شده، یک asynchronous waits است؛ برخلاف روش همزمان استفاده از خاصیت Result یا متد Wait.

خلاصه‌ی بحث

Await را با متدهای همزمان Wait یا خاصیت Result بلاک نکنید. در غیراینصورت در ترد اجرا کننده‌ی دستورات، یک deadlock رخ‌خواهد داد؛ زیرا نتیجه‌ی await باید به context جاری بازگشت داده شود اما این context توسط خواص یا متدهای همزمان فراخوانی شده بعدی، قفل شده‌است.

نظرات خوانندگان

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۰۸ ۱۸:۳۱

با تشکر از شما، مطلب خیلی جالبی بود.
یک سوال فرض کنید در یک برنامه وب می‌خواهیم در داخل ویو تاریخ آخرین مراجعه کاربر به سایت را نمایش دهیم، برای این کار یک متد الحاقی نوشته‌ام که توسط User.Identity.GetLastActivity در دسترس باشد:

```
public static DateTime GetLastActivity(this System.Security.Principal.IIdentity user)
{
            var service = SmObjectFactory.Container.GetInstance<IApplicationUserManager>();
            return service.FindByIdAsync(int.Parse(user.GetUserId())).Result.LastActivity;
}
```

سپس در داخل ویو به راحتی در دسترس است. مورد فوق خروجی مورد نظر را ارائه میدهد اما با توجه به نکاتی که بیان کردید استفاده از خاصیت Result سبب بروز deadlock خواهد شد. از چه روشی برای این حالت بهتر است استفاده شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۸ ۱۸:۵۰

در این حالت خاص ضرورتی به استفاده از متدهای Async نیست و اگر ASP.NET Identity نمونه‌ی غیر async ارائه نداده‌است، خودتان ایجاد کنید. برای مثال در کلاس ApplicationUserManager ، شیء this.Users معادل همان users = >() _uow.Set<ApplicationUser است.

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۰۸ ۱۹:۱

در حالت فوق اولین چیزی که به ذهنم رسید رندر کردن خروجی با استفاده از ChildAction بود یعنی به این صورت:

```
[ChildActionOnly]
        public async Task<ActionResult> GetLastActivity()
        {
            var query = await _userManager.FindByIdAsync(User.Identity.GetUserId<int>());
            var result = query.LastActivity;
            return PartialView("GetLastActivity", result);
        }
```

ویو:

```
@model DateTime
@Model
```

اما به محض اجرای برنامه استثنای زیر صادر شد:

```
{"HttpServerUtility.Execute blocked while waiting for an asynchronous operation to complete."}
```

در نهایت بعد از کمی جستجو متوجه شدم که ChildActionها از async پشتیبانی نمی‌کنند(+)
در هر صورت ممنون، از روشی که پیشنهاد دادید استفاده خواهم کرد.