عنوان: **انجام کارهای پس زمینه در ASP.NET 4.5.2**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۰۲ ۱۰:۲۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Schedule, MVC

دات نت 4.5.2 قابلیت توکاری را به نام در صف قرار دادن یک کار پس زمینه، اضافه کرده‌است که در ادامه خلاصه‌ای از آن‌را مرور خواهیم کرد.

روش متداول ایجاد کارهای پس زمینه

ساده‌ترین روش انجام کارهای پس زمینه در برنامه‌های دات نتی، استفاده از متدهایی هستند که یک ترد جدید را ایجاد می‌کنند مانند Task.Run, Task.Factory.StartNew, Delegate.BeginInvoke, ThreadPool.QueueUserWorkItem و امثال آن. اما ... این روش‌ها در برنامه‌های ASP.NET ایده‌ی خوبی نیستند!
از این جهت که موتور ASP.NET در این حالات اصلا نمی‌داند که شما کار پس زمینه‌ای را ایجاد کرده‌اید. به همین جهت اگر پروسه‌ی برنامه پس از مدتی recycle شود ، تمام کارهای پس زمینه‌ی موجود نیز از بین خواهند رفت.

معرفی HostingEnvironment.QueueBackgroundWorkItem

متد HostingEnvironment.QueueBackgroundWorkItem به دات نت 4.5.2 اضافه شده‌است تا بتوان توسط آن یک کار پس زمینه را توسط موتور ASP.NET شروع کرد و نه مانند قبل، بدون اطلاع آن. البته باید دقت داشت که این کارهای پس زمینه مستقل از هر نوع درخواستی اجرا می‌شوند.
در این حالت چون موتور ASP.NET از وجود کار پس زمینه‌ی آغاز شده مطلع است، در صورت فرا رسیدن زمان recycle شدن برنامه، کل AppDomain را به یکباره نابود نخواهد کرد. البته این مورد فقط به این معنا است که در صورت فرا رسیدن زمان recycle شدن پروسه، با تنظیم یک CancellationToken، اطلاع رسانی خواهد کرد. در این حالت حداکثر 30 ثانیه فرصت خواهید داشت تا کارهای پس زمینه را بدون مشکل خاتمه دهید. اگر کار پس زمینه در این مدت به پایان نرسد، همانند قبل، کل AppDomain نابود خواهد شد.

این متد دو overload دارد و در هر دو حالت، تنظیم خودکار پارامتر CancellationToken توسط ASP.NET، بیانگر آغاز زمان خاتمه‌ی کل برنامه است:

```
public static void QueueBackgroundWorkItem(Action<CancellationToken> workItem);
public static void QueueBackgroundWorkItem(Func<CancellationToken, Task> workItem);
```

در متد اول، یک متد معمولی از نوع void قابل پردازش است. در متد دوم، می‌توان متدهای async Task دار را که قرار است کارهای async را پردازش کنند، معرفی نمود.
علت استفاده از Action و Func در اینجا، امکان تعریف خلاصه و inline یک متد و ارسال پارامتری به آن از طرف برنامه است، بجای تعریف یک اینترفیس جدید، نیاز به پیاده سازی آن اینترفیس و بعد برای مثال ارسال یک مقدار از طرف برنامه به متد Stop آن (بجای تعریف یک اینترفیس تک متدی، از Action و یا Func نیز می‌توان استفاده کرد).

نمونه‌ای از نحوه‌ی فراخوانی این دو overload را در ذیل مشاهده می‌کنید:

```
HostingEnvironment.QueueBackgroundWorkItem(cancellationToken =>
{
        //todo: ...
});

HostingEnvironment.QueueBackgroundWorkItem(async cancellationToken =>
{
        //todo: ...
        await Task.Delay(20000, cancellationToken);
});
```

پشت صحنه‌ی HostingEnvironment.QueueBackgroundWorkItem

روش استاندارد ثبت و معرفی یک کار پس زمینه در ASP.NET، توسط پیاده سازی اینترفیسی به نام IRegisteredObject انجام می‌شود. سپس توسط متد HostingEnvironment.RegisterObject می‌توان این کلاس را به موتور ASP.NET معرفی کرد. در این حالت زمانیکه AppDomain قرار است خاتمه یابد، متد Stop اینترفیس IRegisteredObject کار اطلاع رسانی را انجام می‌دهد.
توسط QueueBackgroundWorkItem دقیقا از همین روش به همراه فراخوانی ThreadPool.QueueUserWorkItemجهت اجرای متد معرفی شده‌ی به آن استفاده می‌شود.
از مکانیزم IRegisteredObject در DNT Scheduler نیز استفاده شده‌است.

پیشنیازها

ابتدا نیاز است به خواص پروژه مراجعه کرده و Target framework را بر روی 5.4.2 قرار داد. اگر به روز رسانی دوم VS 2013 را نصب کرده باشید، این نگارش هم اکنون بر روی سیستم شما فعال است. اگر خیر، امکان دریافت و نصب آن، به صورت جداگانه نیز وجود دارد:

NET Framework 4.5.2.

Developer pack

محدودیت‌های QueueBackgroundWorkItem

- از آن در خارج از یک برنامه‌ی وب ASP.NET نمی‌توان استفاده کرد.
- توسط آن، خاتمه‌ی یک AppDomain تنها به مدت 30 ثانیه به تاخیر می‌افتد؛ تا فرصت داشته باشید کارهای در حال اجرا را با حداقل خسارت به پایان برسانید.
- یک work item، اطلاعاتی را از فراخوان خود دریافت نمی‌کند. به این معنا که مستقل از زمینه‌ی یک درخواست اجرا می‌شود.
- استفاده‌ی از آن الزاما به این معنا نیست که کار درخواستی شما حتما اجرا خواهد شد. زمانیکه که کار خاتمه‌ی AppDomain آغاز می‌شود، فراخوانی‌های QueueBackgroundWorkItem دیگر پردازش نخواهند شد.
- اگر برنامه به مقدار CancellationToken تنظیم شده توسط ASP.NET دقت نکند، جهت پایان یافتن کار در حال اجرا، صبر نخواهد شد.

نظرات خوانندگان

نویسنده: سعید شیرعلی
تاریخ: ۱۳۹۳/۰۹/۰۸ ۱۵:۱۲

با سلام؛ من می‌خوام یک کار در پس زمینه انجام بدم به صورتی که یک تابع را با چند آرگمان صدا بزنم و این تابع در یک تسک جدید اجرا بشود، به طوری که اگر من این تابع را هر چند بار که صدا بزنم همه آنها در یک تسک جدید و مجزا از هم اجرا بشود.
من چطوری میتونم این کار را انجام بدهم؟ با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۸ ۱۵:۱۸

- مطلب فوق در همین مورد است. `BackgroundWorkItem` به معنای در صف قرار دادن این کارها برای پردازش مجزا از هم **Queue** است.
- از آن استفاده کردید؟ به چه مشکلی برخوردید؟