عنوان: **مروری بر کاربرد DoEvents**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۴ ۱۶:۱۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Windows forms

چند روز قبل هنگام استفاده از DoEvents در یک برنامه windows forms ، ناگهان پیغام stack overflow ظاهر شد! برای علت یابی و رفع آن کمی جستجو کردم که خلاصه‌ی آن به شرح زیر است:

DoEvents چیست؟

DoEvents یکی از متدهای کلاس Application در فضای نام System.Windows.Forms است.
ویندوز جهت مدیریت رخدادهای مختلف از یک صف استفاده می‌کند. رخدادهایی مانند کلیک ماوس، تغییر اندازه‌ی یک فرم و مواردی شبیه به آن ابتدا در یک صف قرار می‌گیرند و سپس پردازش می‌شوند. زمانیکه کنترلی مشغول پاسخ دهی به یک رخداد می‌گردد، سایر رخدادها هنوز در صف هستند و پردازش نخواهند شد. بنابراین اگر برنامه‌ی شما در یک روال رخدادگردان کلیک، عملیاتی طولانی را در حال انجام باشد، بدلیل عدم پردازش سایر رخدادها اینطور به نظر خواهد رسید که هنگ کرده است.
روش صحیح پردازش یک عملیات طولانی استفاده از یک ترد دیگر می‌باشد تا ترد اصلی برنامه که کار مدیریت رابط کاربر برنامه را به عهده دارد، درگیر این عملیات طولانی نشده و پاسخگوی رخدادهای رسیده باشد.
راه میان‌بر و ساده‌ای که اینجا وجود دارد استفاده از DoEvents می‌باشد (بدون ایجاد یک ترد جدید). برای مثال اگر در روال رخ دادگردان کلیک یک برنامه، حلقه‌ای طولانی در حال پردازش است، هر از چندگاهی این متد فراخوانی شود، رخدادهای در صف قرار گرفته فرصت ارسال به ترد اصلی برنامه را یافته و برنامه در حالت هنگ به نظر نخواهد رسید.
برای نمونه مثال زیر را در دو حالت با Application.DoEvents و بدون آن اجرا کنید:

```
private void btnProcessWithDoEvents_Click(object sender, EventArgs e)
{
for (int i = 0; i < 100000; i++)
{
  TextBox1.Text = "Processing " + i.ToString();
  Application.DoEvents();
}
}
```

در حالت بدون استفاده از Application.DoEvents ، تنها آخرین عبارت پردازش شده را در TextBox1 مشاهده خواهید کرد و همچنین در این حین، برنامه در حالت هنگ به نظر می‌رسد و برعکس.

مشکلات احتمالی حاصل از استفاده از Application.DoEvents :

الف) حس غلط پایان یافتن عملیات پیش از موعد
در مثال فوق در حین استفاده از Application.DoEvents ، دکمه‌ی btnProcessWithDoEvents مجددا فعال شده و قابل کلیک کردن می‌شود ولی آیا این بدین معنا است که پردازش قبلی به پایان رسیده است؟ به یک سری از کاربرها هم click-happy user گفته می‌شود! یعنی از کلیک کردن مجدد لذت می‌برند! در این حالت حتما باید دکمه‌ی btnProcessWithDoEvents را در ابتدای پردازش غیرفعال کرد و سپس در انتهای آن باید مجددا فعال شود.
مورد مشکل کلیک مجدد حتی می‌تواند منجر به تخریب اطلاعات در حال پردازش شود. فرض کنید برنامه در حال ذخیره‌ی اطلاعات در یک فایل است و کاربر مرتبا بر روی دکمه‌ی پردازش مربوطه کلیک کنید. فایل نهایی از یک سری اطلاعات ناهماهنگ و بی‌ربط پر خواهد شد.

ب) مشکل stack overflow
اگر علاقمند باشید، این مورد را می‌توان به صورت زیر شبیه سازی کرد:

یک تایمر را به برنامه اضافه کنید و یک دکمه. در روال رخدادگردان کلیک مربوط به دکمه، دستورات زیر را اضافه کنید:

```
private void btnStartTimer_Click(object sender, EventArgs e)
{
  this.timer1.Enabled = true;
  this.timer1.Start();
  this.timer1.Interval = 20;

}
```

و در روال tick مربوط به تایمر، دستورات زیر را اضافه کنید:

```
private void timer1_Tick(object sender, EventArgs e)
{
  Thread.Sleep(50);
  Application.DoEvents();
}
```

برنامه را اجرا کرده و یکی دو دقیقه صبر کنید، حتما با پیغام خطای stack overflow مواجه خواهید شد. چرا؟
فواصل زمانی اجرای تایمر به 20 میلی ثانیه تنظیم شده است اما در روال رخداد گردان tick آن، نیاز به 50 میلی ثانیه (بیش از 20 میلی ثانیه) یا بیشتر برای اجرا دارد. با رسیدن به Application.DoEvents ، رخداد در صف قرار گرفته‌ی دیگر tick بلافاصله اجرا می‌شود و همینطور الی آخر، تا بالاخره stack overflow حاصل خواهد شد.

پس چه باید کرد؟

الف) هنگام استفاده از Application.DoEvents به موارد فوق حتما دقت داشته باشید.
ب) بجای استفاده از این روش که در بیشتر موارد یک ضعف برنامه نویسی محسوب می‌شود، شروع به استفاده از روش‌های غیرهمزمان نمائید. برای مثال استفاده از :

BackgroundWorker

Asynchronous delegates

Threads

تنها موردی را که هنگام کار با تردها باید در نظر داشت این است که امکان دسترسی به کنترل‌های یک فرم را از ترد دیگری که آن کنترل را ایجاد نکرده است، ندارید و برای این مورد راه حل‌های زیادی موجود است.
همچنین بخاطر داشته باشید در یک ترد استفاده از Application.DoEvents هیچ معنایی ندارد. ترد اصلی برنامه وظیفه‌ی به روز رسانی رابط کاربر برنامه و پاسخگویی به رخدادهای رسیده را به عهده دارد. زمانیکه پردازش در تردی دیگر صورت می‌گیرد، ترد اصلی برنامه تا پایان پردازش متد شما قفل نخواهد شد که نیازی به استفاده از این متد باشد. در این حالت استفاده از Application.DoEvents ، سبب بالا رفتن مصرف حافظه‌ی برنامه و همچنین بالا رفتن میزان مصرف CPU خواهد شد.

جهت مطالعه بیشتر

Keeping your UI Responsive and the Dangers of Application.DoEvents

نظرات خوانندگان

نویسنده: افشار محبی
تاریخ: ۱۳۸۷/۱۲/۱۷ ۰۹:۰۵:۰۰

در wpf هم مشکل freez شدن UI وجود دارد با این تفاوت که استفاده از DoEvents امکان پذیر نیست و راه حل‌های مشابه هم چندان چنگی به دل نمی‌زنند. آیا راهی اصولی برای wpf وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۷ ۰۹:۳۰:۰۰

راه اصولی همان استفاده از ترد است. در wpf یک سری نکته ریز در این مورد هست که در مقاله زیر به آن اشاره شده است:
http://ascendedguard.com/2007/11/proper-multi-threading-in-wpf.html