عنوان: **آشنایی با Defensive programming**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۳۱ ۱۳:۴۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Tips

تصادف برای یک راننده حتی در صورت داشتن بیمه نامه‌ای معتبر، گران تمام خواهد شد (از لحاظ جانی/مادی/...). بنابراین صرف نظر از اینکه شرکت بیمه کننده چه میزان از خسارت راننده را جبران خواهد کرد، باید تا حد ممکن از تصادفات بر حذر بود (defensive driving).

در برنامه نویسی، استثناءها (Exceptions) مانند تصادفات هستند و مدیریت استثناءها (exception handling)، همانند بیمه خودرو می‌باشند. هر چند مدیریت استثناءها جهت بازگردان برنامه شما به ادامه مسیر مهم هستند، اما جایگزین خوبی برای Defensive programming به شمار نمی‌روند. استثناءها و مدیریت آن‌ها برای برنامه گران تمام می‌شوند (خصوصا از لحاظ میزان مصرف منابع سیستمی و سربارهای مربوطه). بنابراین در برنامه باید توجه خاصی را به این موضوع معطوف داشت که چه زمانی، چگونه و در کجا ممکن است استثنائی رخ دهد و علاج واقعه را پیش از وقوع آن نمود.

اصل اول Defensive programming : همیشه ورودی دریافتی را تعیین اعتبار کنید
به مثال زیر دقت بفرمائید:

```
public void LogEntry(string msg)
    {
        string path = GetPathToLog();
        using (StreamWriter writer = File.AppendText(path))
        {
            writer.WriteLine(DateTime.Now.ToString(CultureInfo.InstalledUICulture));
            writer.WriteLine("Entry: {0}", msg);
            writer.WriteLine("--------------------");
        }
    }
```

قرار هست رخدادهای برنامه را توسط این متد، لاگ کنیم. اکنون لحظه‌ای دقت نمائید که این تابع در چه مواقعی ممکن است دچار مشکل شود:
path می‌تواند یک رشته خالی باشد.
path می‌تواند نال باشد.
path می‌تواند حاوی کاراکترهای غیرمجازی باشد.
path می‌تواند فرمت نادرستی داشته باشد.
path می‌تواند به محلی ناصحیح اشاره نماید.
path می‌تواند اصلا وجود نداشته باشد.
فایل مورد نظر ممکن است readonly باشد.
برنامه ممکن است دسترسی لازم را برای نوشتن در مسیر ذکر شده، نداشته باشد.
فایل مورد نظر ممکن است توسط پروسه‌ای دیگر قفل شده باشد.
ممکن است در لحظه نوشتن یا خواندن بر روی فایل، هارد دیسک دچار مشکل گردد.
و ...

رخ دادن هر کدام از موارد ذکر شد منجر به بروز یک استثناء خواهد شد.

چگونه این وضعیت را بهبود بخشیم؟
فرض کنید متد GetPathToLog قرار است مسیر ذخیره سازی لاگ‌ها را از کاربر در یک برنامه ASP.Net دریافت کند. برای این

منظور باید حداقل دو مورد را منظور کرد.

```
<asp:TextBox ID="txtPath" runat="server" MaxLength="248" />

<asp:RequiredFieldValidator ID="reqval_txtPath" runat="server" ControlToValidate="txtPath"
ErrorMessage="Path is required." />

<asp:RegularExpressionValidator ID="regex_txtPath" runat="server" ControlToValidate="txtPath"
ErrorMessage="Path is invalid." ValidationExpression='^([a-zA-
Z]\:)(\\{1}|((\\{1})[^\\]([^/:*?<>”|]*(?<![ ]))+)$' />
```

برای تکست باکس ارائه شده، ابتدا یک RequiredFieldValidator در نظر گرفته شده تا مطمئن شویم که کاربر حتما مقداری را وارد خواهد کرد. اما این کافی نیست. سپس با استفاده از عبارات باقاعده و RegularExpressionValidator بررسی خواهیم کرد که آیا فرمت ورودی صحیح است یا خیر.

تا اینجا 4 مورد اول مشکلاتی که ممکن است رخ دهند (موارد ذکر شده فوق)، کنترل می‌شوند بدون اینکه احتمال رخ دادن این استثناءها در برنامه وجود داشته باشد. Defensive programming به این معنا است که طراحی برنامه باید به گونه‌ای باشد که در اثر استفاده‌ی غیر قابل پیش بینی از آن، در عملکرد برنامه اختلالی رخ ندهد.

نظرات خوانندگان

نویسنده: افشار محبی
تاریخ: ۱۳۸۸/۰۶/۰۱ ۰۹:۱۴:۰۱

ولی بعضی وقت‌ها اعتبارسنجی مقادیر ورودی آنقدر زمان‌گیر هستند و پیچیده که بهتر است بگذاریم خطا به وجود بیاید و وقتی به وجود آمد آن را «مدیریت» کنیم.

نویسنده: SirAsad
تاریخ: ۱۳۸۸/۰۶/۰۱ ۱۰:۰۰:۲۱

ولی بعضا نمیشه کل خطاها رو کنترل کردبه خصوص در برنامه های Enterprise چونکه اگر بخوای مدیریت کنی باید به قول امریکن ها Rest of you life ات رو IF بنویسی .