عنوان: **آشنایی با Defensive programming - قسمت دوم**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۰۱ ۱۷:۵۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Tips

در ادامه یک سری از خط مشی‌های متداول در defensive programming را با هم مرور خواهیم کرد:

-1 بررسی نال بودن اشیاء
سعی در استفاده از اشیاء نال، به یک NullReferenceException منتهی خواهد شد. اگر به هر دلیلی امکان نال بودن یک شیء وجود داشت، پیش از استفاده از آن، حتما این وضعیت را بررسی نمائید.
بهترین ابزاری هم که برای این منظور می‌توان استفاده کرد، نگارش جدید افزونه‌ی ReSharper است که زیر شیءایی را که احتمال نال بودن آن می‌رود یک خط آبی رنگ می‌کشد.

```
using (SqlDataReader myReader = sqlCommand.ExecuteReader())
{
    while (myReader.Read())
        {  Possible 'System.NullReferenceException'
```

-2 بررسی آرگومان‌های دریافتی
برای نمونه اگر متد شما تاریخی را بر اساس DateTime دریافت می‌کند، حتما حدود آن‌را بررسی نمائید. برای مثال دریافت تاریخ 31 اسفند از کاربر، به یک ArgumentOutOfRangeException منتهی خواهد شد. بنابراین آرگومان‌های دریافت شده باید انتظارات مربوطه را برآورده کنند و پیش از استفاده حتما بررسی گردند تا بتوان مشکلات را به کاربر به صورت واضحی گوشزد کرد. (خطای ArgumentOutOfRangeException برای کاربر نهایی بی‌معنی است)

-3 وضعیت اشیاء را بررسی کنید
برای مثال بستن یک کانکشن به دیتابیس در صورت بسته بودن آن، به یک InvalidOperationException منتهی می‌شود. بنابراین بهتر است ابتدا وضعیت این شیء بررسی شده و سپس عملیات خاصی بر روی آن صورت گیرد.

-4 هنگام کار با آرایه‌ها دقت کنید
اگر اندیس فراخوانی شده کمتر از صفر یا بیشتر از اندازه‌ی آرایه باشد به یک IndexOutOfRangeException بر خواهید خورد. بنابراین همواره بهتر است که این بررسی پیش از بروز واقعه انجام شود.

-5 مراقب الگوریتم‌های بازگشتی باشید
هر چند متدهای بازگشتی در بعضی از موارد کار راه انداز هستند اما اگر بدون دقت از آن‌ها استفاده شود ممکن است سبب ایجاد یک حلقه‌ی بی نهایت شده و نهایتا برنامه با یک StackOverFlowException خاتمه می‌یابد (این مورد در دات نت فریم ورک تنها حالتی است که با try و catch قابل مهار نیست).

-6 هنگام تبدیل اشیایی از نوع object مراقب باشید
اگر قصد تبدیل یک شیء را به نوعی داشته باشید که با مقدار ذخیره شده در آن مطابقت ندارد، یک InvalidCastException حاصل خواهد شد. بنابراین در اینگونه موارد بهتر است از اپراتورهای as و یا is استفاده کنید. هنگام استفاده از as اگر عملیات

تبدیل با موفقیت صورت نگیرد، حاصل عملیات تنها یک شیء نال خواهد بود و استثنایی رخ نخواهد داد.

-7 بجای متد Parse از TryParse استفاده کنید
برای مثال در دات نت جهت تبدیل یک رشته به مقداری عددی می‌توان از int.Parse و یا int.TryParse استفاده کرد. در حالت اول اگر عملیات تبدیل میسر نباشد حتما یک FormatException رخ خواهد داد اما در حالت دوم در صورت موفقیت آمیز بودن عملیات تبدیل، خروجی true خواهد بود و حاصل اصلی را با یک پارامتر از نوع out در اختیار شما قرار می‌دهد.

و به صورت خلاصه
- ورودی‌های کاربر را محدود کرده (مثلا اگر قرار است عددی را وارد کند، از یک تکست باکس عددی (masked edit controls) استفاده کنید) و یا آن‌ها را دقیقا بررسی نمائید تا احتمال بروز خطاهای بعدی را کاهش دهید.
- زمانیکه می‌توان از بروز یک exception جلوگیری کرد، بهتر است مدیریت آن‌را به قسمت catch بلاک try/catch واگذار نکرد.
- و هنگامیکه با برنامه نویسی نمی‌توانید تمامی خطاهای ممکن را پیش بینی کرده و آن‌ها را مدیریت کنید، برای مدیریت استثناءها برنامه داشته باشید.

## نظرات خوانندگان

نویسنده: Anonymous
تاریخ: ۱۳۸۸/۰۶/۰۱ ۲۳:۵۶:۱۷

سلام
خسته نباشید من تازه از جاوا به #c اومدم (یه ماهی هست شروع کردم) وبلاگ خیلی پرمحتوا و فعالی دارین
می خواستم اگر ممکن باشه در مورد masked edit controls یکم توضیح بدین.
مرسی و ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۰۲ ۰۰:۰۳:۴۸

سلام
یک نمونه masked edit control با استفاده از یکی از پلاگین‌های jQuery برای ASP.Net درست کرده‌ام که می‌تونید شرح آن‌را در آدرس زیر ملاحظه کنید
http://www.dotnettips.info/2008/11/jquery-aspnet.html