عنوان: **مدیریت خطا ها در ASP.NET MVC**
نویسنده: حسین رضیئی
تاریخ: ۱۳۹۱/۰۴/۲۲ ۲۰:۴۰
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net MVC, Handle Errors

احتمال بوجود اومدن خطا در اجرای یک برنامه همیشه هست. خطاهایی که برنامه نویس ممکنه هیچ وقت فکر نکنه چنین خطایی در اجرای برنامه ای که نوشته بوجود بیاد و هیچ وقت هم از اون خطا اطلاع پیدا نکنه. ثبت و بررسی خطاهایی که در برنامه بوجود میاد میتونه مفید باشه اما آیا باید همه‌ی خطاها رو ثبت کرد؟
خیر. ممکنه برخی از این خطاها سلامت سیستم رو به خطر نندازه و اصلا ارتباطی هم به برنامه نداشته باشه . به طور مثال زمانی که کاربر یک صفحه ای و یا فایلی رو درخواست میده که وجود نداره.
توسط رویداد OnException میتونیم به خطاهای بوجود اومده در سطح یک Controller دسترسی داشته باشیم و اون رو مدیریت کنیم.

```
protected override void OnException(ExceptionContext filterContext)
{
    // Bail if we can't do anything
    if (filterContext == null)
        return;

    // log
    var ex = filterContext.Exception ??
            new Exception("No further information exists.");
    // save log ex.Message

    filterContext.ExceptionHandled = true;

    filterContext.Result = View("ViewName");
    base.OnException(filterContext);
}
```

ما تمام خطا هایی که در سطح یک کنترل اتفاق می‌افته رو با دوباره نویسی(override) متد OnException مدیریت میکنیم. اما اگه بخواهیم محدوده‌ی این مدیریت رو بیشتر کنیم و کاری کنیم که روی تمام نرم افزار اعمال بشه باید چه کار کنیم؟
رویداد Application_Error در Global.asax محل مناسبی برای انجام این کار هست اما باید چند مسئله رو در نظر بگیریم:

Server.Transfer : این متد یک فایل (میتونه یک صفحه باشه)رو به خروجی میفرسته بدون اینکه آدرس تغییر کنه(کاربر به صفحه دیگه ای منتقل بشه) نکته ای که وجود داره اینه که برای انتقال نیاز به وجود یک فایل فیزیکی بر روی سیستم داره تا اون فایل رو به خروجی منتقل کنه ، در پروژه‌های MVC فایل فیزیکی (به شکلی که در ASP.NET وجود داره) وجود نداره و بوسیله routeها ، controllerها و view ها یک صفحه(خروجی) تولید میشه بنابراین ما نمیتونیم در ASP.NET MVC از این متد استفاده کنیم و باید به دنبال راه حلی برای فرستادن پاسخ مناسب باشیم.
Response.Redirect : این متد کاربر رو به یک صفحه‌ی دیگه منتقل میکنه و StatusCode رو برابر با 301 مقدار دهی میکنه که معنای انتقال صفحه هست و برای مشخص کردن "خطای داخلی سرور" (Internal Server Error) با کد 500 و پیدا نشدن فایل با کد 404 مناسب نیست .این کد(StatusCode) برای موتورهای جستجو اهمیت زیادی داره لیست کامل این کدها و توضیحاتشون رو میتونید در این آدرس مطالعه کنید.
Response.Clear : این متد باید فراخوانی بشه تا خروجی تولید شده تا این لحظه رو پاک کنیم.
Server.ClearError : این متد باید فراخوانی بشه تا از ایجاد صفحه زرد رنگ خطا جلوگیری کنه.
با توجه به نکات بالا، با طی کردن مراحل زیر میتونیم خطاهای ایجاد شده رو مدیریت کنیم.

بدست آوردن آخرین خطای ایجاد شده.
بدست آوردن کد خطای ایجاد شده.
ثبت خطا برای بررسی (ممکن هست شما برخی خطاها مثل پیدا نشدن فایل رو ثبت نکنید).
پاک کردن خروجی ایجاد شده تا این لحظه.
پاک کردن خطای ایجاد شده در سرور.
فرستادن یک خروجی مناسب بدون تغییر مسیر.

```
protected void Application_Error(object sender, EventArgs e)
{
    var error = Server.GetLastError();
    var code = (error is HttpException) ? (error as HttpException).GetHttpCode() : 500;

    if (code != 404)
    {
        // save log error.Message
    }

    Response.Clear();
    Server.ClearError();

    string path = Request.Path;
    Context.RewritePath(string.Format("~/Errors/Http{0}", code), false);
    IHttpHandler httpHandler = new MvcHttpHandler();
    httpHandler.ProcessRequest(Context);
    Context.RewritePath(path, false);
}
```

خروجی مناسب به کاربر از طریق از یکی از Action هایی انجام میشه که در ErrorsController هست. باید توجه داشته باشید که اگه در خود این ErrorsController خطایی رخ بده ، یک حلقه‌ی بی پایان بین این کنترلر و رویداد Application_Error اتفاق خواهد افتاد.

```
public class ErrorsController : Controller
{
    [HttpGet]
    public ActionResult Http404()
    {
        Response.StatusCode = 404;
        return View();
    }

    [HttpGet]
    public ActionResult Http500()
    {
        Response.StatusCode = 500;
        return View();
    }
}
```

نظرات خوانندگان

نویسنده: eli
تاریخ: ۱۳۹۱/۰۴/۲۵ ۱۱:۳۷

عالی بود مرسی ممنون