مقدمه:

متد RedirectToAction جهت انتقال روند اجرای برنامه از یک اکشن به اکشن دیگری استفاده می گردد. اما این انتقال دارای نکاتی می باشد که عدم رعایت آن می تواند باعث بروز مشکلاتی گردد و در این ترفند قصد بررسی این موضوع را داریم.

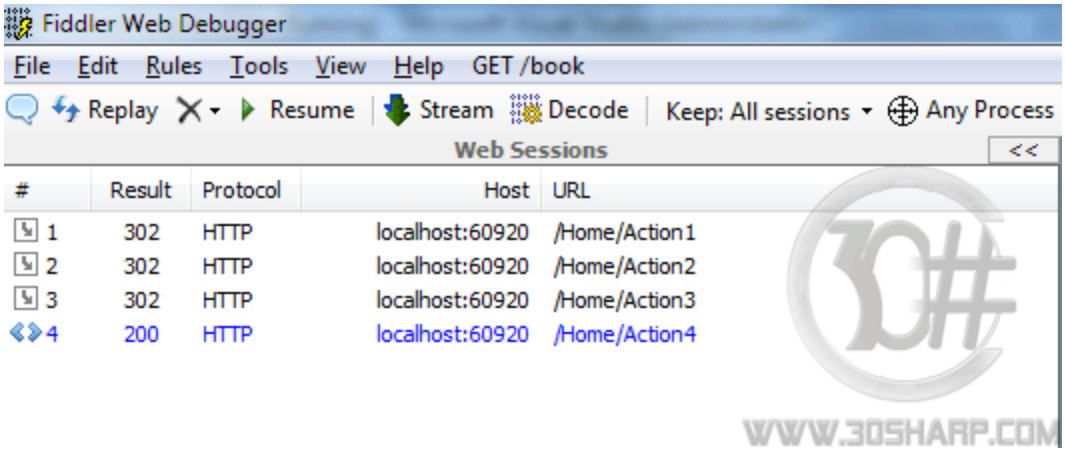
آغاز:

به قطعه کد زیر توجه کنید.

```
public class HomeController : Controller
{
    // Action 1
    public ActionResult Action1()
    {
        return RedirectToAction("Action2");
    }

    // Action 2
    public ActionResult Action2()
    {
        return RedirectToAction("Action3");
    }

    // Action 3
    public ActionResult Action3()
    {
        return RedirectToAction("Action4");
    }

    // Action 4
    public ActionResult Action4()
    {
        return View();
    }
}
```

در قطعه کد بالا ٤ اکشن وجود دارد که هر کدام از اکشن ها با استفاده از فرمان RedirectToAction اجرای دستورات را به اکشن بعدی منتقل می کنند و در نهایت اکشن آخری یعنی Action4 یک View را رندر می کند. اگر با استفاده از دیباگر نرم افزار Visual Studioروند اجرای کدها رو دنبال کنید، تصور خواهید کرد که تمامی این اعمال در سمت سرور و به ترتیب اتفاق می افتند.

اما حقیقت اینگونه نیست و اگر با ابزار Fiddler ترافیک HTTP را مانیتور کنید، تصویری را مشابه شکل زیر مشاهده خواهید نمود.



تصویر فوق نشانگر این واقعیت می باشد که به جای ۱ درخواست که مورد نظر ما بوده است، ٤ درخواست برای وب سایت صادر شده است. در حقیقت پس از اجرای فرمان RedirectToAction یک کد وضعیت (Status Code) به شماره ۳۰۲ به سمت مرورگر کاربر ارسال می گردد و بیانگر این مطلب می باشد که این صفحه به آدرس جدید انتقال داده شده است. سپس مرورگر کاربر درخواست جدیدی را برای صفحه جدید (اکشن بعدی) ارسال می نماید. به همین دلیل است که در شکل بالا ٤ درخواست صادر شده است که فقط کد وضعیت در خواست آخری یعنی اکشن ٤ برابر ۲۰۰ می باشد یعنی این اکشن یک View را رندر نموده است.

بنابراین استفاده نابجا از فرمان RedirectToAction می تواند تاثیر منفی بر روی راندمان و کارایی برنامه بگذارد. اما این موضوع تنها مشکل موجود نیست و در ادامه مقاله مشکل دیگری را نیز بررسی می کنیم که خصوصا برنامه نویسان نو آموز ASP.NET MVC و افرادی که به تازگی از WebForm ها به این تکنولوژی مهاجرت می کنند، معمولا با آن روبرو می شوند.

اکنون فرض کنید فرمی طراحی نموده ایم که قرار است دانش آموزان با استفاده از آن ثبت نام کنند. پیاده سازی اکشن های مربوطه به شکل زیر می باشد.

```
public ActionResult Create()
{
    return View();
}

[HttpPost]
public ActionResult Create(Student student)
{
    if (!ModelState.IsValid)
    {
        // Add a general error message
        ModelState.AddModelError("", "There is something wrong!");

        return RedirectToAction("Create");
    }
```

```
    //Create new user and rest of implementation here...
}
```

اکشن اول وظیفه نمایش فرم ثبت نام را بر عهده دارد و اکشن دوم پس از تکمیل فرم ثبت نام و کلیک شدن دکمه ثبت نام فراخوانی می گردد. قبل از اینکه به مطالعه ادامه مطالب بپردازید، بهتر است لحظه ای تامل کنید! با توجه به مطالبی که تاکنون در این مقاله ارائه شده است، به عقیده شما چه مشکلی در اکشن دوم در قطعه کد بالا وجود دارد؟!

در اکشن دوم اگر اشکالی در اطلاعات وارد شده وجود داشته باشد و یا به عبارت بهتر اگر ModelState معتبر نباشد، ما یک پیام خطا برای نمایش به کاربر به آن اضافه نموده ایم و سپس با استفاده از دستور RedirectToAction ادامه دستورات را به اکشن اول که وظیفه آن نمایش فرم ثبت نام است، انتقال داده ایم. با اجرای قطعه کد بالا متوجه خواهید شد که اگر ModelState معتبر نباشد نه تنها پیامی به کاربر نمایش داده نمی شود، بلکه فیلدهایی که توسط کاربر پر شده بودند نیز همگی خالی شده و دانش آموز مجبور خواهد شد که مجددا همه موارد را پر نماید!

دلیل این امر نیز کاملا مشخص می باشد. همانگونه که در ابتدای این مقاله ذکر شد، هنگامی که از دستور RedirectToAction جهت انتقال بین اکشن دومی به اولی استفاده می کنیم، مرورگر کاربر یک درخواست جدید برای اجرای اکشن اول صادر می کند و بنابراین این صفحه کاملا از ابتدا و به شکل خام لود می شود و اطلاعات موجود در ModelState از بین خواهند رفت.

برای رفع این مشکل قطعه کد اکشن دوم را می توانیم به شکل زیر تغییر دهیم.

```
[HttpPost]
public ActionResult Create(Student student)
{
    if (!ModelState.IsValid)
    {
        // Add a generic error message
        ModelState.AddModelError("", "There is something wrong!");

        return Create();
    }

    //Create new user and rest of implementation here...
}
```

ملاحظه می کنید که اینبار مستقیما اکشن Create اولی را فراخوانی نموده و محتوای آن را برگردانده ایم و بنابراین اطلاعات ModelStateدر فرم ما نمایش داده خواهد شد. مسئله جالب اینجاست که اگر این موضوع را فراموش کنید، با استفاده از دیباگر ویژوال استودیو و قرار دادن BeakPoint در هر دو اکشن وTrace نمودن روند اجرای کدها، متوجه هیچ تفاوتی بین دو روش ذکر شده نمی شوید و این موضوع باعث اتلاف وقت فراوان خوهد شد.