عنوان: **استفاده از چند Routing در یک پروژه ASP.NET MVC بدون درد و خونریزی**
نویسنده: مرتضی دلیل
تاریخ: ۱۳۹۳/۰۴/۱۳ ۱۰:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, routing

کار کردن با مسیریابی برای یک پروژه ساده ، نیاز به طراحی پیچیده ندارد. مسیریابی پیش فرض موجود در فایل RoutConfig.cs برای کارهای ابتدایی کافیست. اما اگر کمی کار پیچیده شود و صفحات مختلفی با منطق‌های متفاوتی ایجاد کنیم، ممکن است با مشکل روبرو شویم. در MVC5 به کمک دخالت ویژگی‌ها در مسیریابی ، کار ساده شده است اما در MVC4 و قبل از آن چه باید کرد؟ پیش از بسط مساله، ابتدا این سوال را پاسخ میدهیم که **چگونه صفحه‌ی start پروژه انتخاب میشود؟**

مسیریابی پیش فرض یک پروژه MVC به شکل زیر است :

```
routes.MapRoute(
           name: "Default",
           url: "{controller}/{action}/{id}",
           defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
       );
```

وقتی یک پروژه MVC را بررسی کنید، مشاهده می‌کنید که در شاخه‌ی اصلی آن، فایل index یا default وجود ندارد و اصولا منطق کار با اکشن‌ها و صدا زدن آنها و دیدن پاسخ‌ها توسط View، این اتفاق را توجیه میکند. پس وقتی درخواستی به سمت شاخه‌ی اصلی ما فرستاده میشود، مسیریابی وارد عمل می‌شود. وقتی پروژه را اجرا میکنید، متد RegisterRoutes از شیءایی که از کلاس RoutingConfig.cs ساخته شده (به فایل global.asax نگاه کنید) فراخوانی می‌شود و شیء Routes از " قالب آدرس " هایی که ما تعیین کرده ایم پُر می‌شود. بخشی از فایل RoutConfig.cs را در تصویر زیر می‌بینیم.

به Url کد فوق نگاه کنید که در حقیقت آدرسی نسبی است. آدرس ما به طور کامل به شکل زیر قابل تعریف است:

```
http://mydomain.com/{controller}/{action}/{id}
```

واضح است که درخواست اولیه به سایت ما دارای بخش‌های controller و action و بخش اختیاری id نیست. پس به شکل پیشفرض بر اساس آنچه در جلوی خصوصیت defaults نوشته شده است، فراخوانی خواهد شد. یعنی اکشن Index از داخل کنترلر Home صدا زده می‌شود و چون id نداریم، هیچ نوع id به متد (اکشن) index ارسال نخواهد شد.

یک روتینگ دیگر به شکل زیر در بالای کد فوق اضافه کنید :

```
routes.MapRoute(
          name: "Default1",
          url: "{controller}/{action}/{id}",
          defaults: new { controller = "News", action = "Index", id = UrlParameter.Optional }
      );
```

حتما متوجه شدید که اینبار با اجرای پروژه، متد(اکشن) Index از کنترلر News فراخوانی خواهد شد. در حقیقت اولین روتینگ ساخته شده همان چیزیست که Asp.NET MVC پس از دریافت ریکوئست به آن رجوع میکند. **دقت کنید که مقدار name در دو MapRouting فوق متفاوت است.**

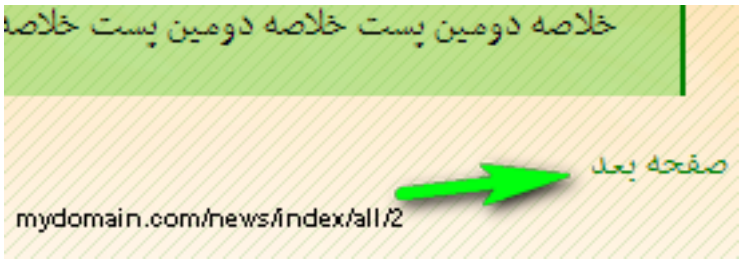
اما این اسم‌ها(name ها) به چه دردی می‌خورند؟

قبل از توضیح در این مورد، default1 را از پروژه حذف میکنیم و با همان MapRouting پیش فرض به کار خود ادامه می‌دهیم. ابتدا مروری بر عملکرد MapRouting انجام میدهیم.

همانطور که می‌دانید برای ایجاد یک لینک از طریق ویوی Razor از چیزی شبیه دستور زیر می‌توانیم استفاده کنیم:

```
<a href="@Url.Action("post", "News", new { id = @item.PostName }, null)"> @item.PostTitle </a>
```

فرض کنید یک سایت خبری داریم و گروه‌های مختلف آن شامل مطالب متفاوتی هستند. کنترلری به نام News ایجاد کرده‌ایم، که دارای یک اکشن به نام post است که نام مطلب را دریافت کرده و یک View به ما برمی‌گرداند. بخش هایی از صفحه اول را در تصاویر زیر می‌بینید. تصویر آخر، بخش پایین صفحه اخبار است، که بوسیله لینک (لینک های) موجود صفحات جابجا میشوند.



وقتی کد فوق در یک صفحه فراخوانی میشود، کالکشن Routes بررسی می‌شود و لینک مورد نظر، با توجه به ورودی‌های Url.Action ساخته می‌شود. ما یک MapRouting به نام Default در کالکشن Routes داریم، پس Url.Action لینک زیر را می‌سازد:

```
http://mydomain.com/news/post/نام-مقاله
```

استفاده از MapRouting مثل یک جعبه است که ورودی آن مقادیر موجود در Url.Action می‌باشد و آنچه به ما بر می‌گرداند یک آدرس برای فراخوانی اکشن مورد نیاز است. (آخرین پارامتر، null قرار داده شده که برای تعیین نوع پروتکل استفاده میشود که http یا https میتواند باشد.)

برای فراخوانی صفحه‌ی اخبار (لیست اخبار) می‌توانید دستور زیر را بنویسید :

```
@Html.ActionLink("اخبار", "Index", "News", null, new { @class="btn btn-success"})
```

این دستور تگ anchor را نیز می‌سازد و به کمک MapRouting آدرسی شبیه آدرس زیر را به ما بر می‌گرداند. (به تفاوت دو دستور دقت کنید):

```
<a href="/News">اخبار</a>
```

واضح است که چون MapRouting یک حالت پیش فرض درونی دارد که دارای اکشن دیفالت index است، پس ActionLink اگر ببیند لینکش در صفحه قرار است به شکل news/index/ تعریف شود خود بخود بخش index را حذف میکند.

فرض کنید بعد از کلیک روی لینک فوق، اکشن index ، یک آبجکتِ لیستی با 10 خبر آخر، ایجاد میکند. پس ما میخواهیم قابلیت صفحه بندی را برای لیست اخبار فعال کنیم و در هر صفحه 10 مطلب را نمایش دهیم. مشکل از همینجا آغاز میشود. MapRouting فعلی جوابگوی ما نخواهد بود و آدرس را به شکل زیر نمایش میدهد.

```
<a href="/News/index?pid=2">صفحه بعد</a>
```

و آنچه ما در View نوشته‌ایم چیزی شبیه کد زیر است :

```
@Html.ActionLink("صفحه بعد", "index", "News", new { pid = …. }, null)
```

مشکلی در ارجاع به صفحات وجود ندارد و با کلیک روی لینک "صفحه بعد" مقدار عدد 2 به اکشن index ارسال میشود و اگر کد نویسی را برای take و skip کردن لیست، درست انجام شده باشد، نتیجه مورد نظر نمایش داده خواهد شد. اما آدرس فوق آدرس زیبایی نیست. اولین فکری که به ذهن برنامه نویس میرسد، ایجاد یک مسیریابی دیگر است. فکر درستیست؛ اما اگر چند بار دیگر این اتفاق بیفتد و در بخش هایی از برنامه نیاز به روتینگ پیدا کنید و روتینگ‌های جدید ایجاد کنید متوجه خواهید شد که مدیریت این MapRouting ها کار خسته کننده و طاقت فرسایی خواهد شد، مخصوصا اگر بدانید که فقط مجاز به استفاده از یک پارامتر optional در هر MapRouting هستید! دست شما کاملا بسته است. لینک‌های بالای سایت را اصلاح میکنید ولی لینک‌های پایین سایت خراب میشوند و بالعکس.

به هر حال MapRouting زیر را به RoutConfig.cs اضافه میکنیم :

```
routes.MapRoute("PostPaging", "{controller}/{action}/{id}/{pid}",
            defaults: new
            {
                controller = "News",
                action = "Index",
                id = "all",
                pid = UrlParameter.Optional

            }
            );
```

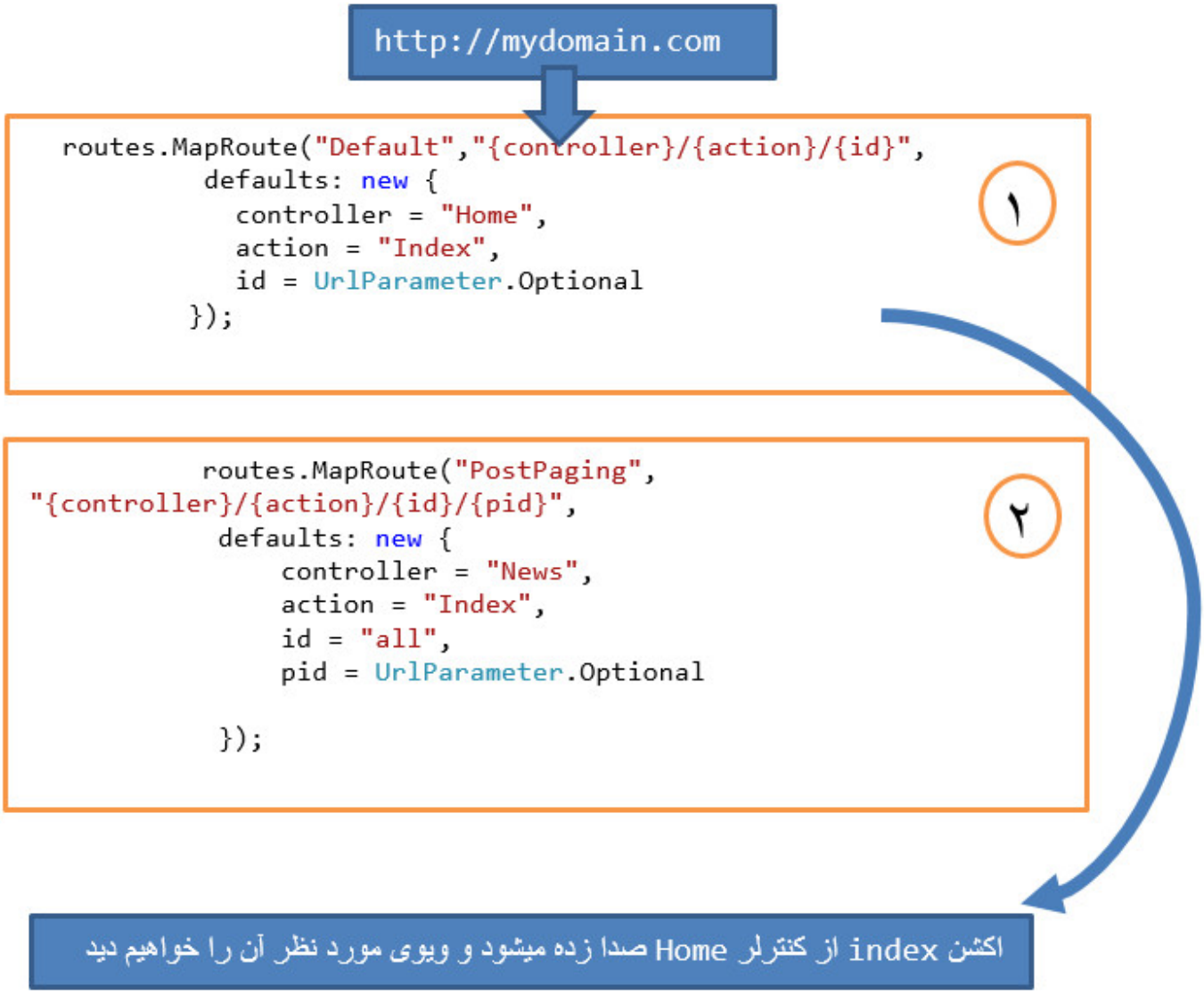
- اسم این MapRouting ، دیگر Default نیست.
- یک پارامتر pid اضافه‌تر از MapRouting اولی دارد.
- pid به عنوان یک پارامتر اختیاری تعریف شده است، پس "قالب آدرس" بسیار شبیه مپ روتینگ قبلی است.
- مقدار id اختیاری نیست، چون قرار است در آینده بتوانیم گروه‌های مختلف موجود در بخش اخبار را صفحه بندی کنیم و قرار نیست پشت سر هم MapRouting ایجاد کنیم و کافیست به جای id اسم گروه را بنویسیم. در حالتیکه اسمی از گروه درلینکهایمان نبرده باشیم به شکل پیشفرض all قرار داده میشود که یعنی کل اخبار مد نظر است. (در اکشن مربوطه باید این تصمیمات را لحاظ کنیم)
- حتما این MapRouting را بعد از MapRouting اولیه بنویسید، کمی پیشتر، علت این امر توضیح داده شد و گفته شد اولین چیزی که MVC پس از درخواست ما میبیند به عنوان Routing بررسی می‌کند (درخواست اولیه) و چون ساختار MapRouting فوق تا اندازه ای شبیه ساختار Default MapRouting است ممکن است با فراخوانی سایت مشکل ایجاد شود.
- میتوانید MapRouting را کمی خاص‌تر هم بنویسیم :

```
routes.MapRoute("NewsPaging", "News/index/{id}/{pid}",
            defaults: new
            {
                controller = "News",
                action = "Index",
                id = "all",
                pid = UrlParameter.Optional
            }
            );
```

اینکار بستگی به پروژه‌ی شما دارد، مپ روتینگ فوق این مزیت را دارد که با مپ روتینگ‌های دیگر به سختی قاطی میشود! یعنی حتا اگر قبل از مپ روتینگ دیفالت نوشته شود برنامه با مشکل مواجه نخواهد شد، چون اصلا شکل درخواست اولیه به سایت، چیزی شبیه این آدرس نیست. اما خاص بودن آن و همچنین نوع بهره گیری از آن با کمک Action یا ActioLink شاید شما را سردرگم خواهد کند.

اما مشکل این MapRouting ها چیست؟

درخواست به سایت آمده و قرار است سایت بارگذاری شود؛ ترتیب زیر در شیء routes ثبت شده است :



در صفحه اول ما لینکی به شکل زیر گذاشته ایم :

```
@Html.ActionLink("اخبار", "Index", "News", null, new { @class="btn btn-success"})
```

مشکلی پیش نخواهد آمد. روند فوق تکرار میشود و ActionLink برای ساختن لینک نهایی از MapRouting اول استفاده میکند (بدون اینکه به MapRouting دوم نگاه کند).

در صفحه اخبار، لینک "صفحه بعد" وجود دارد ، آیا این لینک به شکل صحیح نمایش داده میشود؟ خیر! نتیجه کار را ببینید :

```
<a href="/News/index?pid=2">صفحه بعد</a>
```

علت واضح است، ActionLink اصلا MapRouting دومی را نمی‌بیند و با همان MapRouting اولی لینک فوق را ساخته است. جای MapRouting ها را عوض کنیم؟ مطمئنید با اینکار درخواست مربوط به صفحه اول شما سلامت به مقصد میرسد؟ مطمئنید با اینکار بقیه Action ها و ActionLink های موجود در صفحه اول شما به درستی تبدیل میشوند؟
به نظر میرسد یک طرح دقیق برای آدرس دهی شاید این مسائل را حل کند ولی نه وقت داریم و نه اعصاب.

دقت کنید ما برای کار با روتینگ با دو مساله مواجه هستیم :
1. ساخته شدن لینک‌ها توسط هلپرها (که از مپ روتینگ‌های ثبت شده ما پیروی میکند)
2. واکنش پروژه به درخواست‌های دریافتی و هدایت آن به اکشن‌های مورد نظر که کاملا به ترتیب ثبت مپ روتینگ‌ها در کالکشن Routes بستگی دارد .

خوشبختانه یک هلپر به شکل زیر در MVC وجود دارد :

```
@Html.RouteLink("صفحه بعد", "PostPaging", new { action = "cat", controller = "News", id =. .., pid = ...})
```

مقدار اول متن لینک، مقدار دوم نام MapRouting ،مقدار سوم یک آبجکت است که نوع اکشن و کنترلر و پارامترهای مورد نظر MapRouting را تعیین میکند. به همین سادگی! این پاسخ به همان سوالیست که در ابتدای مقاله مطرح شد. **"نام گذاری مپ روتینگ‌ها به چه درد میخورد؟"**

نتیجه برای لینک موجود در صفحه اخبار چیزی شبیه شکل زیر خواهد شد :

```
<a href="/News/cat/all/2">صفحه بعد</a>
```

چند شکل دیگر از این هلپر را ببینید :

```
@Html.RouteLink("اخبار", "Default", new { controller = "news" })

@Html.RouteLink("درباره ما", "pages", new RouteValueDictionary(new { controller="Page", action="Index", pagename="درباره-ما"}), new Dictionary<string, Object> { { "data-toggle", "popover" }, { "data-placement", "top" } })
```

همانطور که میبینید در RoutLink اولی، اخبار را به کمک MapRouting با نام default بازنویسی میکنیم و نتیجه چیزی شبیه کد زیر خواهد شد :

```
<a href="/News">اخبار</a>
```

در RoutLink دومی اولا از یک RoutValueDictionary به جای یک آبجکت ساده استفاده کرده ایم و مقادیر را به شکل فوق به کنترلر و اکشن و ...نسبت داده ایم ثانیا برای بخش HTML نیز پراپرتی‌ها را به کمک یک دیکشنری ارسال میکنیم، به خاطر وجود "-" در یکی از خواص، راه دیگری غیر از اینکار نداریم.

اما دقت کنید که از یک MapRouting جدید استفاده کردیم که نامش pages است،

```
routes.MapRoute(
     "pages",
     "page/{pagename}",
      new { controller = "Page", action = "Index" }
 );
```

۳

این MapRoutnig را قبل از دیگر Routing ها می‌نویسیم؟ وسط دو MapRouting قبلی مینویسیم؟ آخر MapRouting ها مینویسیم؟ آیا فرقی میکند؟ اگر سریع بگوییم خـــیر! اشتباه کرده ایم. واقعا فرق میکند.

دقت کنید موضوع MapRouting فقط ایجاد یک لینک‌تر و تمیز نیست؛ RoutLink یک لینک تمیز بر اساس مپ روتینگی که نامش برده شده ایجاد میکند اما تضمین نمیکند که با کلیک بر روی لینک به هدف برسیم و به خطای 404 برخورد نکنیم! اگر روی لینک

کلیک کنید آدرس شروع به تفسیر شدن میکند و این تفسیر اصلا ربطی به نامی که به RoutLink داده ایم ندارد و ترتیب موجود در کالکشن ایجاد شده **در RoutConfig تعیین کننده است.(آبجکت Routes) اگر MapRouting فوق را در انتهای بقیه بگذاریم صفحه اول لود میشود ولی با کلیک روی "درباره ما" صفحه پیغام خطا خواهد داد.**

باید به یاد داشته باشیم برای اجرای درخواست (کلیک روی لینک)، آنچه برای ASP.NET MVC اهمیت دارد، ترتیب قرار گیری MapRouting ها در RouteRegister است و ما به کمک RoutLink تنها مشکل ساخت لینک‌ها بر اساس قالب MapRouting مورد نظرمان را حل کردیم و این به ما تضمینی برای هدایت آن لینک به مکان درست را نخواهد داد.

اگر ترتیب به شکل زیر باشد :

1

2

3 باشد. درخواست اولیه برای بالا آمدن سایت به مشکل برخورد نمی‌کند چون همان مپ روتینگ 1 اجرا میشود. اما مشکل فوق به وجود خواهد آمد و خطای 404 با کلیک بر روی "درباره ما" نمایش داده خواهد شد چون با کلیک روی "درباره ما" مپ روتینگ شماره 1 وارد عمل میشود.

اگر ترتیب به شکل زیر باشد :

3

2

1 آیا اصلا صفحه اول سالم لود خواهد شد؟ خیر! درخواست نسبی " / " (یا به طور کامل http://mydomain.com) شماره 3 را به خیر پشت سر میگذارد، چون اصلا چیزی به نام page در آدرس وجود ندارد که از این MapRouting بخواهد پیروی کند. اما در شماره 2 گیر می‌افتد چون این فرمت را حفظ کرده است :

```
"{controller}/{action}/{id}/{pid}"
```

Pid که اختیاریست (بر اساس قوانین تعریف شده در مپ روتینگ شماره 2) بقیه موارد نیز حالت دیفالت دارند، یعنی اگر تعریف نشده باشند (مثل همین درخواست) خودبخود جایگذاری میشوند. پس به طور کلی صفحه اول ما تغییر می‌کند و اکشن index از کنترلر Home اجرا نمیشود و به جایش اکشن Index از کنترلر News اجرا میشود و صفحه اول را از دست میدهیم و صفحه اخبار را میبینیم! نتیجه اینکه نباید هرگز 2 را قبل از 1 قرار دهیم.

اگر ترتیب به شکل زیر باشد :

3

1

2 این همان چیزیست که مد نظر ماست. اولا 1 قبل از 2 است و صفحه اول برای لود شدن به مشکل برخورد نمیکند.
ثانیا وقتی روی "درباره ما" کلیک میکنیم همان شماره 3 فراخوانی میشود و بقیه مپ روتینگ‌ها اعمال نمیشوند.

تنظیم این مقاله با هدف آموزش صورت گرفته است. حتما با تفکر و به خاطر سپردن نکات با طرح ریزی ساختار صفحات به کمک کمترین تعداد MapRouting به بهترین نتایج خواهیم رسید.

نظرات خوانندگان

نویسنده: احمد اقامحمدی
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۶:۶

با سلام
یه سوالی داشتم، میخواستم بدونم با این روش میشه توی MVC چنین ادرسی ساخت؟ اگر نمیشه با چه روشی میشه درست کرد؟
http://www.sitename.com/books/gifting-store/box-sets/children/pr?sid=bks,cwa,0b9,hj7&otracker=ch_vn_giftingsto

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۶:۴۲

شبیه به مطلب مخفی کردن کوئری استرینگ‌ها در ASP.NET MVC توسط امکانات Routing هست.