عنوان: **چطور مسیریابی‌های ASP.NET MVC را دیباگ کنیم؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۸ ۸:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MVC

سؤال: من برای تهیه sitemap برنامه، یک route سفارشی نوشته‌ام تا یک فایل xml ایی را که در وب سرور، وجود خارجی ندارد، در آدرس‌های سایت قابل دسترسی کند. برای مثال:

```
routes.MapRoute(
                "SiteMap_route", // Route name
                "sitemap.xml", // URL with parameters
                new { controller = "Sitemap", action = "index", name = UrlParameter.Optional, area = ""
} // Parameter defaults
            );
```

با استفاده از این مسیریابی خاص، قرار است هر زمانیکه آدرس http://site/sitemap.xml در مرورگر وارد شد، برنامه در پشت صحنه، به صورت خودکار به کنترلر sitemap و اکشن متد index آن مراجعه کرده و یک محتوای پویای XML ایی را تولید کند و بازگشت دهد. اما ... کار نمی‌کند! یعنی آدرس یاد شده اصلا پاسخ نمی‌دهد. چرا؟ نحوه‌ی ثبت مسیریابی سفارشی تعریف شده نیز به صورت زیر است:

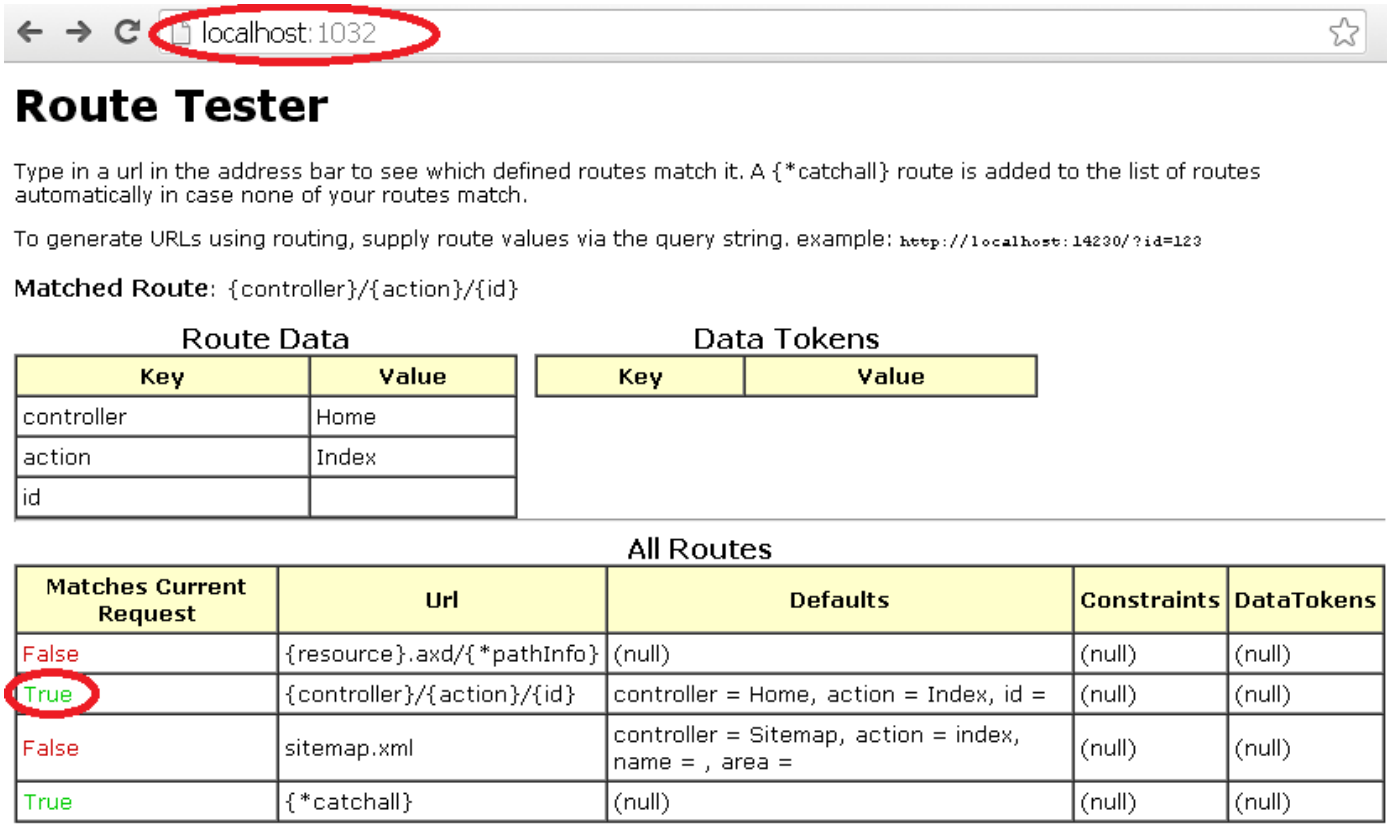
```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

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

            routes.MapRoute(
                "SiteMap_route", // Route name
                "sitemap.xml", // URL with parameters
                new { controller = "Sitemap", action = "index", name = UrlParameter.Optional, area = ""
} // Parameter defaults
            );
        }
```

پاسخ: اگر با تقدم و تاخر و معنای مسیریابی‌های تعریف شده آشنایی داشته باشید، شاید بلافاصله بتوانید مشکل را حدس بزنید. اما اگر تعداد مسیریابی‌های سفارشی تعریف شده زیاد باشد، اینکار ساده نیست و حتما نیاز به ابزار دیباگ دارد تا بتوان تشخیص داد که در صفحه جاری کدامیک از مسیریابی‌های تعریف شده کار را تمام کرده‌اند و نوبت به دیگری نرسیده است.

برای این منظور می‌توان از افزونه‌ای به نام RouteDebug نوشته یکی از اعضای سابق تیم ASP.NET MVC استفاده کرد:
RouteTester.zip
کار کردن با آن نیز بسیار ساده است.
الف) ارجاعی را به اسمبلی RouteDebug.dll (حاصل از کامپایل پروژه فوق) به پروژه جاری ASP.NET MVC خود اضافه کنید.
ب) سپس به فایل Global.asax.cs خود مراجعه و در سطر آخر متد Application_Start آن، فراخوانی ذیل را اضافه نمائید:

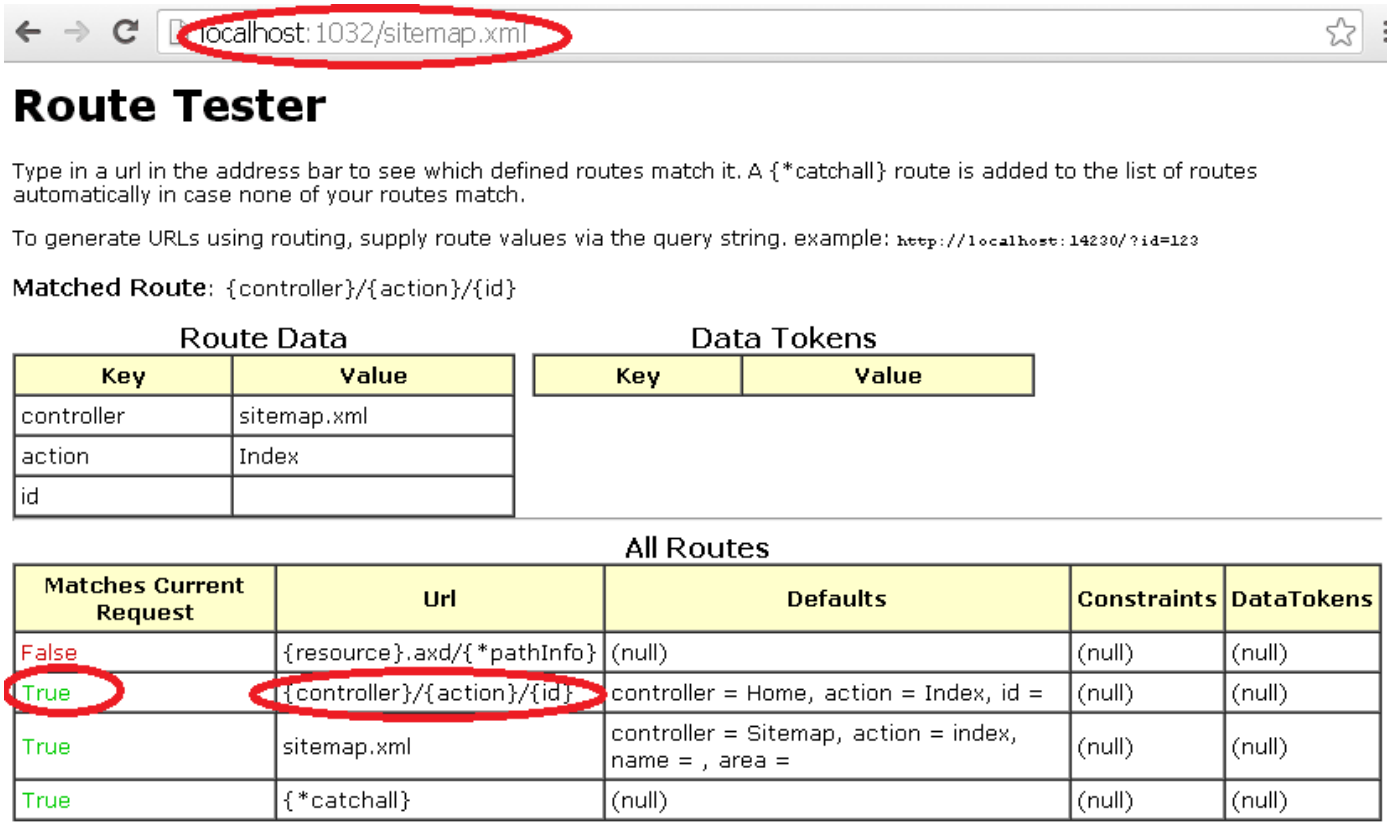
```
RouteDebug.RouteDebugger.RewriteRoutesForTesting(RouteTable.Routes);
```

اکنون هر صفحه و آدرسی را که باز کنید، بجای محتوای اصلی صفحه، مسیریابی‌های فعال و برنده آن‌را مشاهده خواهید کرد. برای مثال در صفحه اول برنامه داریم:



نکته مهمی که در این تصویر باید به آن دقت داشت، اولین True سبز رنگی است که نمایش می‌دهد. یعنی اولین مسیریابی که کار هدایت و نمایش صفحه جاری را برعهده دارد. در اجرای عادی ASP.NET MVC، همینجا کار پردازش سیستم مسیریابی صفحه جاری خاتمه خواهد یافت و نوبت به سایرین نخواهد رسید.

در مورد صفحه sitemap.xml چطور؟ اگر این آدرس را در مرورگر، بدون فعال سازی افزونه RouteDebug وارد کنیم، پیام 404 را دریافت می‌کنیم. اگر افزونه را فعال کنیم، اینبار به صفحه زیر خواهیم رسید:



localhost:1032/sitemap.xml

Route Tester

Type in a url in the address bar to see which defined routes match it. A {*catchall} route is added to the list of routes automatically in case none of your routes match.

To generate URLs using routing, supply route values via the query string. example: http://localhost:14230/?id=123

Matched Route: {controller}/{action}/{id}

Route Data

Key	Value
controller	sitemap.xml
action	Index
id	

Data Tokens

Key	Value

All Routes

Matches Current Request	Url	Defaults	Constraints	DataTokens
False	{resource}.axd/{*pathInfo}	(null)	(null)	(null)
True	{controller}/{action}/{id}	controller = Home, action = Index, id =	(null)	(null)
True	sitemap.xml	controller = Sitemap, action = index, name = , area =	(null)	(null)
True	{*catchall}	(null)	(null)	(null)

بله. همانطور که مشاهده می‌کنید، مسیریابی پیش فرض، اینبار نیز برنده بوده است و اولین تطابق صورت گرفته با آن صورت می‌گیرد. بنابراین اصلا کار به استفاده از مسیریابی سفارشی تعریف شده توسط ما نخواهد رسید.
بنابراین محل تعریف این مسیریابی را اکنون به پیش از مسیریابی پیش فرض انتقال می‌دهیم:

```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                "SiteMap_route", // Route name
                "sitemap.xml", // URL with parameters
                new { controller = "Sitemap", action = "index", name = UrlParameter.Optional, area = ""
} // Parameter defaults
            );

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

در ادامه اگر مجددا مسیر sitemap.xml را درخواست کنیم، به تصویر ذیل خواهیم رسید:

Route Tester

Type in a url in the address bar to see which defined routes match it. A {*catchall} route is added to the list of routes automatically in case none of your routes match.

To generate URLs using routing, supply route values via the query string. example: http://localhost:14230/?id=123

Matched Route: sitemap.xml

Route Data

Key	Value
controller	Sitemap
action	index
name	
area	

Data Tokens

Key	Value

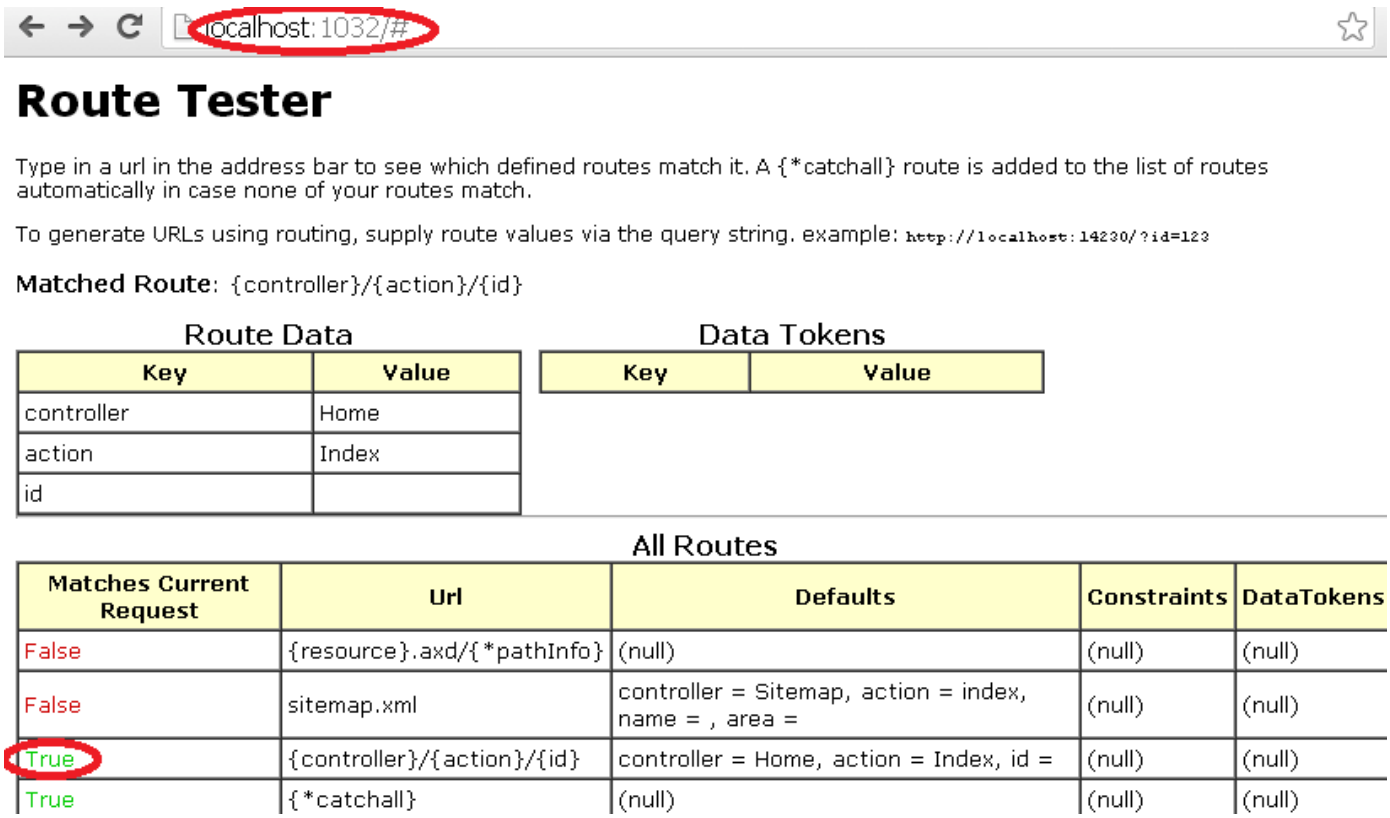
All Routes

Matches Current Request	Url	Defaults	Constraints	DataTokens
False	{resource}.axd/{*pathInfo}	(null)	(null)	(null)
True	sitemap.xml	controller = Sitemap, action = index, name = , area =	(null)	(null)
True	{controller}/{action}/{id}	controller = Home, action = Index, id =	(null)	(null)
True	{*catchall}	(null)	(null)	(null)

بله. با این تنظیم صورت گرفته، اینبار دیگر سیستم مسیریابی، برای تفسیر مسیر سفارشی تعریف شده، به سراغ مسیریابی پیش فرض نخواهد رفت و کار همینجا خاتمه می‌یابد.

سؤال: آیا اینکار تداخلی در عملکرد اصلی برنامه ایجاد نمی‌کند؟ مثلا اگر به مسیر `index` کنترلر `home` مراجعه کنیم، مشکلی نخواهد بود؟

پاسخ: خیر. برای آزمایش آن باز هم به افزونه `RouteDebug` مراجعه خواهیم کرد:



همانطور که مشخص است، مسیریابی برنده در این حالت، همان مسیریابی پیش فرض است و نه مسیریابی سفارشی آدرس خاص sitemap.xml سایت.

یک نکته تکمیلی

افزونه گلیمپس نیز امکان دیباگ Routeها را دارد؛ اما توانایی بررسی مشکلات Routing یک خطای 404 مانند مثال فوق را حداقل تا زمان نگارش این مطلب ندارد و همان افزونه RouteDebug یاد شده، بهتر عمل می‌کند.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۱۸ ۲۳:۵۲

یک نکته‌ی تکمیلی

افزونه‌ی Routing Assistant نیز برای این منظور مفید است.

نویسنده: محمود راستین
تاریخ: ۱۳۹۴/۰۶/۲۰ ۱۴:۲۶

مرسی جناب نصیری بابت این مطلب.

در اپدیت 2 دیگه نیازی به افزودن اون خط ذر فایل Global.asax.cs نیست. کافیه با دستور زیر در package manager console اون رو اضافه کنیم :

```
PM>Install-Package routedebugger
```

و به صورت خودکار خط زیر در appConfig اضافه میشه:

```
<add key="RouteDebugger:Enabled" value="true" />
```

اگر هم به صورت دستی اضافه کنیم باید این خط رو خودمون اضافه کنیم. حالا با مقدار true و false میشه RouteDebugger رو فعال یا غیر فعال کنیم.