عنوان: **تهیه خروجی RSS در برنامه‌های ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۰ ۱۷:۵۱
آدرس: www.dotnettips.info
برچسب‌ها: MVC, RSS, ASP.Net MVC

در این مطلب با کتابخانه تهیه شده جهت تولید فیدهای RSS سایت جاری آشنا خواهید شد. در این کتابخانه مسایل زیر لحاظ شده است:

1) تهیه یک ActionResult جدید به نام FeedResult برای سازگاری و یکپارچگی بهتر با ASP.NET MVC
2) اعمال زبان فارسی به خروجی نهایی (این مورد حداقل در IE محترم شمرده می‌شود و فید را، راست به چپ نمایش می‌دهد)
3) اعمال جهت‌های rtl و ltr به متون فارسی یا انگلیسی به صورت خودکار؛ به نحوی که خروجی نهایی در تمام فیدخوان‌ها یکسان به نظر می‌رسد.
4) اعمال کاراکتر یونیکد RLE به صورت خودکار به عناوین فارسی (این مساله سبب می‌شود تا عنوان‌های ترکیبی متشکل از حروف و کلمات فارسی و انگلیسی، در فیدخوان‌هایی که متون را، راست به چپ نمایش نمی‌دهند، صحیح و بدون مشکل نمایش داده شود.)
5) نیازی به کتابخانه اضافی خاصی ندارد و پایه آن فضای نام System.ServiceModel.Syndication دات نت است.
6) تنظیم صحیح ContentEncoding و ContentType جهت نمایش بدون مشکل متون فارسی

سورس کامل این کتابخانه به همراه یک مثال استفاده از آن را از اینجا می‌توانید دریافت کنید:
MvcRssApplication.zip

توضیحاتی در مورد نحوه استفاده از آن

کتابخانه کمکی MvcRssHelper به صورت یک پروژه Class library جدا تهیه شده است. بنابراین تنها کافی است ارجاعی را به اسمبلی آن به پروژه خود اضافه کنید. البته این پروژه برای MVC4 کامپایل شده است؛ اما با MVC3 هم قابل کامپایل است. فقط باید ارجاع به اسمبلی System.Web.Mvc.dll را حذف و نمونه MVC3 آن‌را جایگزین کنید.
پس از اضافه کردن ارجاعی به اسمبلی آن، اکشن متد فید شما یک چنین امضایی را باید بازگشت دهد:

```
FeedResult(string feedTitle, IList<FeedItem> rssItems, string language = "fa-IR")
```

آیتم اول، نام فید است. مورد دوم، لیست عناوینی است که قرار است در فید ظاهر شوند. برای مثال، هر بار 20 آیتم آخر مطالب سایت را گزارش‌گیری کرده و به فرمت لیستی از FeedItemها باید ارائه دهید. FeedItem هم یک چنین ساختاری دارد و در اسمبلی MvcRssHelper قرار گرفته:

```
using System;

namespace MvcRssHelper
{
    public class FeedItem
    {
        public string Title { set; get; }
        public string AuthorName { set; get; }
        public string Content { set; get; }
        public string Url { set; get; }
        public DateTime LastUpdatedTime { set; get; }
    }
}
```

دو نکته در اینجا حائز اهمیت است:
الف) تاریخ استاندارد یک فید میلادی است نه شمسی. به همین جهت DateTime در اینجا ظاهر شده است.
ب) Url آدرسی است به مطلب متناظر در سایت و باید یک آدرس مطلق مثلا شروع شده با http باشد.

یک مثال از استفاده آن

فرض کنید مدل مطالب سایت ما به نحو زیر است:

```
using System;

namespace MvcRssApplication.Models
{
    public class Post
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string AuthorName { set; get; }
        public string Body { set; get; }
        public DateTime Date { set; get; }
    }
}
```

و یک منبع داده فرضی (کوئری از بانک اطلاعاتی یا استفاده از یک ORM یا ... موارد دیگر) نهایتا تعدادی رکورد را در اختیار ما خواهد گذاشت:

```
using System;
using System.Collections.Generic;
using MvcRssApplication.Models;

namespace MvcRssApplication.DataSource
{
    public static class BlogItems
    {
        public static IList<Post> GetLastBlogPostsList()
        {
            var results = new List<Post>();
            for (int i = 1; i < 21; i++)
            {
                results.Add(new Post
                {
                     AuthorName = "شخص " + i,
                     Body = "مطلب " + i,
                     Date = DateTime.Now.AddDays(-i),
                     Id = i,
                     Title = "عنوان "+i
                });
            }
            return results;
        }
    }
}
```

اکنون برای نمایش این اطلاعات به صورت یک فید، تنها کافی است به صورت زیر عمل کنیم:

```
using System.Collections.Generic;
using System.Web.Mvc;
using MvcRssApplication.DataSource;
using MvcRssApplication.Models;
using MvcRssHelper;

namespace MvcRssApplication.Controllers
{
    public class HomeController : Controller
    {
        const int Min15 = 900;

        [OutputCache(Duration = Min15)]
        public ActionResult Index()
        {
            var list = BlogItems.GetLastBlogPostsList();
            var feedItemsList = mapPostsToFeedItems(list);
            return new FeedResult("فید مطالب سایت ما", feedItemsList);
        }

        private List<FeedItem> mapPostsToFeedItems(IList<Post> list)
        {
            var feedItemsList = new List<FeedItem>();
            foreach (var item in list)
```

```
            {
                feedItemsList.Add(new FeedItem
                {
                    AuthorName = item.AuthorName,
                    Content = item.Body,
                    LastUpdatedTime = item.Date,
                    Title = item.Title,
                    //این آدرس باید مطلق باشد به نحو زیر
                    Url = this.Url.Action(actionName: "Details", controllerName: "Post", routeValues:
new { id = item.Id }, protocol: "http")
                });
            }
            return feedItemsList;
        }
    }
}
```

توضیحات

BlogItems.GetLastBlogPostsList منبع داده فرضی ما است. در ادامه باید این اطلاعات را به صورت لیستی از FeedItemها در آوریم. می‌توانید از AutoMapper استفاده کنید یا در این مثال ساده، نحوه انجام کار را در متد mapPostsToFeedItems ملاحظه می‌کنید.

نکته مهم بکارگرفته شده در متد mapPostsToFeedItems، استفاده از Url.Action برای تولید آدرس‌هایی مطلق متناظر با کنترلر نمایش مطالب سایت است.

پس از اینکه feedItemsList نهایی به صورت پویا تهیه شد، تنها کافی است return new FeedResult را به نحوی که ملاحظه می‌کنید فراخوانی کنیم تا خروجی حاصل به صورت یک فید RSS نمایش داده شود و قابل استفاده باشد. ضمنا جهت کاهش بار سرور می‌توان از OutputCache نیز به مدتی مشخص استفاده کرد.

نظرات خوانندگان

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۶/۲۰ ۱۹:۸

سلام. بسیار کاربردی و عالی. سپاس.

نویسنده: احمدعلی شفیعی
تاریخ: ۱۳۹۱/۰۶/۲۱ ۰:۳۳

یک‌سری از وبگاه‌ها عادت دارن از description به‌جای content استفاده کنند.آیا این کتابخانه این قابلیت رو داره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۱ ۰:۴۳

خروجی کتابخانه فوق، RSS استاندارد است و description دارد. مقادیر خواص کلاس FeedItem نهایتا به این خروجی نگاشت می‌شود.
برای نمونه این خروجی رو می‌تونید در سورس نهایی فید سایت مشاهده کنید.

نویسنده: محمد صافدل
تاریخ: ۱۳۹۱/۰۶/۲۱ ۱۰:۳۰

ممنون آقای نصیری. بسیار عالی بود و ابهامات زیادی در مورد Rss برای من برطرف شد.

نویسنده: محمد صافدل
تاریخ: ۱۳۹۱/۰۶/۲۳ ۱۵:۲۴

در صورت امکان در مورد نحوه استفاده از کتابخانه MvcRssHelper در پروژه‌های ASP.NET و تغییراتی که باید در این کتابخانه انجام شود، راهنمایی کنید. با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۳ ۱۶:۲۲

آنچنان تفاوتی نداره. فقط اینبار باید از فایل‌های ashx استفاده کنید. روال ProcessRequest آن معادل روال ExecuteResult موجود در فایل FeedResult.cs است.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۹:۹

سلام
ممنوم بابت مطلب خوبتون
ولی فایل که برای دانلود گذاشتید مشکل داره
پروژه ام وی سی توش نیست ... ممنونم می‌شم اگر اصلاح کنید

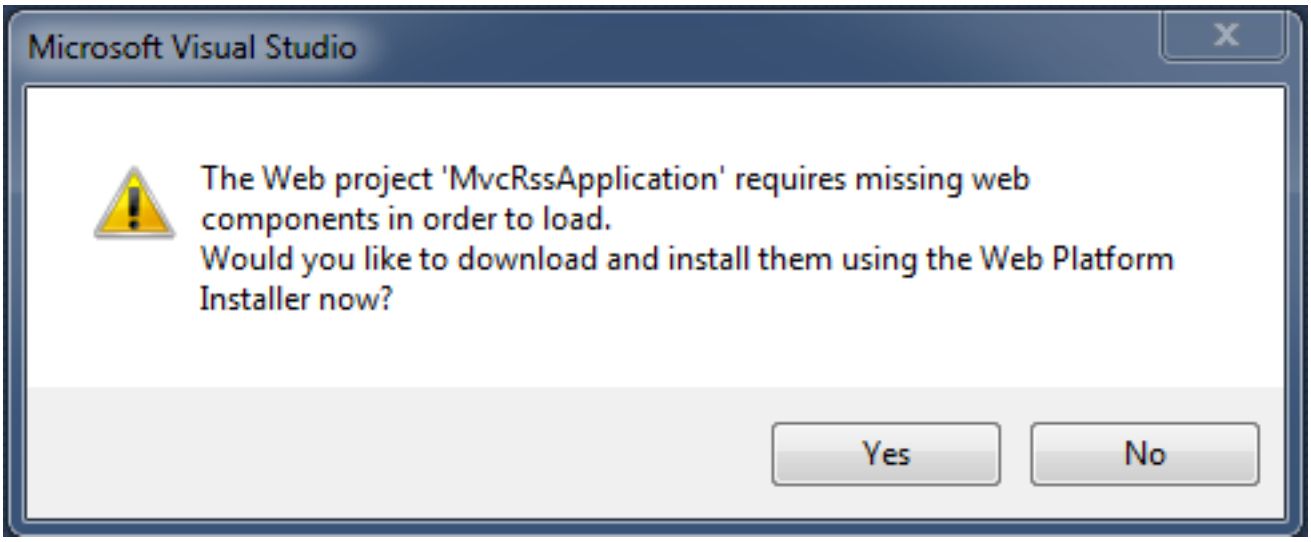
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۹:۴۱

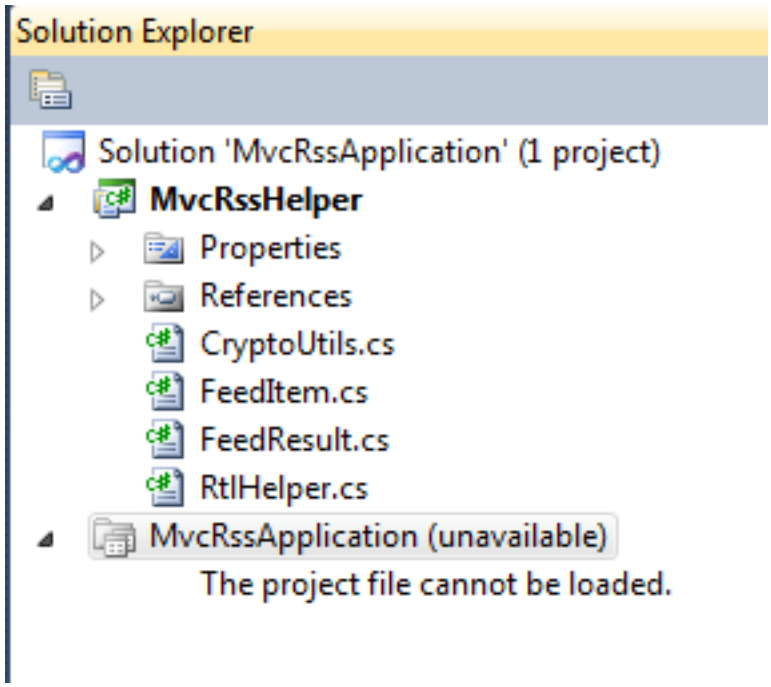
داخل فایل MvcRssApplication.zip فوق این موارد هست:
پوشه MvcRssApplication : یک پروژه مثال

پوشه MvcRssHelper : کتابخانه اصلی

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۲۳ ۱۵:۲۲

سلام، من هم توی لود پروژه مشکل دارم، البته همه نسخه‌های mvc رو نصب شده داشتم ولی:





نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۳ ۱۶:۲۶

- این یک پروژه MVC4 است. همچنین فرض هم بر این است که IIS Express بر روی سیستم شما نصب است (^).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۳۰ ۰:۱۱

نسخه به روز شده این پروژه:

MvcRssApplication.zip

تغییرات:

پیشتر فقط تاریخ به روز رسانی را داشت:

```
<a10:updated>2012-10-20T16:09:13+03:30</a10:updated>
```

الان تاریخ انتشار هم به آن اضافه شده:

```
<pubDate>Sat, 30 Jan 2010 02:26:32 -0800</pubDate>
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۲ ۲۳:۷

به روز رسانی سوم:

MvcRssApplication3.zip

تغییرات (بر اساس نظرات W3C Feed Validation Service):

- channel link به آن اضافه شد.
- فضای نام a10 ایی که تولید می‌شد با atom جایگزین شد.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۸/۱۳ ۱۷:۳

سلام

از این روش می‌شه برای ساخت sitemap هم استفاده کرد؟
اگر نه چطور؟