عنوان: **پیاده سازی دکمه «بیشتر» یا «اسکرول نامحدود» به کمک jQuery در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۹:۳۶
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, MVC, JavaScript

مدتی است در اکثر سایت‌ها و طراحی‌های جدید، به جای استفاده از روش متداول نمایش انتخاب صفحه 1، 2 ... 100، برای صفحه بندی اطلاعات، از روش اسکرول نامحدود یا infinite scroll استفاده می‌کنند. نمونه‌ای از آن‌را هم در سایت جاری با دکمه «بیشتر» در ذیل اکثر صفحات و مطالب سایت مشاهده می‌کنید.

در ادامه قصد داریم نحوه پیاده سازی آن‌را در ASP.NET MVC به کمک امکانات jQuery بررسی کنیم.

مدل برنامه

```
namespace jQueryMvcSample02.Models
{
    public class BlogPost
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }
    }
}
```

در این برنامه و مثال، قصد داریم لیستی از مطالب را توسط اسکرول نامحدود، نمایش دهیم. هر آیتم نمایش داده شده، ساختاری همانند کلاس BlogPost دارد.

منبع داده فرضی برنامه

```
using System.Collections.Generic;
using System.Linq;
using jQueryMvcSample02.Models;

namespace jQueryMvcSample02.DataSource
{
    public static class BlogPostDataSource
    {
        private static IList<BlogPost> _cachedItems;
        /// <summary>
        /// با توجه به استاتیک بودن سازنده کلاس، تهیه کش، پیش از سایر فراخوانی‌ها صورت خواهد گرفت
        /// باید دقت داشت که این فقط یک مثال است و چنین کشی به معنای
        /// تهیه یک لیست برای تمام کاربران سایت است
        /// </summary>
        static BlogPostDataSource()
        {
            _cachedItems = createBlogPostsInMemoryDataSource();
        }

        /// <summary>
        /// هدف صرفا تهیه یک منبع داده آزمایشی ساده تشکیل شده در حافظه است
        /// </summary>
        private static IList<BlogPost> createBlogPostsInMemoryDataSource()
        {
```

```
            var results = new List<BlogPost>();
            for (int i = 1; i < 30; i++)
            {
                results.Add(new BlogPost { Id = i, Title = "عنوان " + i, Body = "متن ... متن ... متن "
+ i });
            }
            return results;
        }

        /// <summary>
        /// پارامترهای شماره صفحه و تعداد رکورد به ازای یک صفحه برای صفحه بندی نیاز هستند
        /// شماره صفحه از یک شروع می‌شود
        /// </summary>
        public static IList<BlogPost> GetLatestBlogPosts(int pageNumber, int recordsPerPage = 4)
        {
            var skipRecords = pageNumber * recordsPerPage;
            return _cachedItems
                        .OrderByDescending(x => x.Id)
                        .Skip(skipRecords)
                        .Take(recordsPerPage)
                        .ToList();
        }
    }
}
```

برای اینکه برنامه نهایی را به سادگی بتوانید اجرا کنید، به عمد از بانک اطلاعاتی خاصی استفاده نشده و صرفا یک منبع داده فرضی تشکیل شده در حافظه، در اینجا مورد استفاده قرار گرفته است. بدیهی است قسمت cachedItems را به سادگی می‌توانید با یک ORM جایگزین کنید.

تنها نکته مهم آن، نحوه تعریف متد GetLatestBlogPosts می‌باشد که برای صفحه بندی اطلاعات بهینه سازی شده است. در اینجا توسط متدهای Skip و Take، تنها بازه‌ای از اطلاعات که قرار است نمایش داده شوند، دریافت می‌گردد. خوشبختانه این متدها معادل‌های مناسبی را در اکثر بانک‌های اطلاعاتی داشته و استفاده از آن‌ها بر روی یک بانک اطلاعاتی واقعی نیز بدون مشکل کار می‌کند و تنها بازه محدودی از اطلاعات را واکشی خواهد کرد که از لحاظ مصرف حافظه و سرعت کار بسیار مقرون به صرفه و سریع است.

کنترلر برنامه

```
using System.Linq;
using System.Web.Mvc;
using System.Web.UI;
using jQueryMvcSample02.DataSource;
using jQueryMvcSample02.Security;

namespace jQueryMvcSample02.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index()
        {
            آغاز کار با صفحه صفر است//
            var list = BlogPostDataSource.GetLatestBlogPosts(pageNumber: 0);
            return View(list); نمایش ابتدایی صفحه//
        }

        [HttpPost]
        [AjaxOnly]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public virtual ActionResult PagedIndex(int? page)
        {
            var pageNumber = page ?? 0;
            var list = BlogPostDataSource.GetLatestBlogPosts(pageNumber);
            if (list == null || !list.Any())
                return Content("no-more-info"); //این شرط ما است برای نمایش عدم یافتن رکوردها

            return PartialView("_ItemsList", list);
        }

        [HttpGet]
        public ActionResult Post(int? id)
```

```
        {
            if (id == null)
                return Redirect("/");

            //todo: show the content here
            return Content("Post " + id.Value);
        }
    }
}
```

کنترلر برنامه را در اینجا ملاحظه می‌کنید. برای کار با اسکرول نامحدود، به ازای هر صفحه، نیاز به دو متد است:
الف) یک متد که بر اساس HttpGet کار می‌کند. این متد در اولین بار نمایش صفحه فراخوانی می‌گردد و اطلاعات صفحه آغازین را نمایش می‌دهد.
ب) متد دومی که بر اساس HttpPost کار کرده و محدود است به درخواستی‌های AjaxOnly همانند متد PagedIndex.
از این متد دوم برای پردازش کلیک‌های کاربر بر روی دکمه «بیشتر» استفاده می‌گردد. بنابراین تنها کاری که افزونه جی‌کوئری تدارک دیده شده ما باید انجام دهد، ارسال شماره صفحه است. سپس با استفاده از این شماره، بازه مشخصی از اطلاعات دریافت و نهایتا یک PartialView رندر شده برای افزوده شدن به صفحه بازگشت داده می‌شود.

دو View برنامه

همانطور که برای بازگشت اطلاعات نیاز به دو اکشن متد است، برای رندر اطلاعات نیز به دو View نیاز داریم:
الف) یک PartialView که صرفا لیستی از اطلاعات را مطابق سلیقه ما رندر می‌کند. از این PartialView در متد PagedIndex استفاده خواهد شد:

```
@model IList<jQueryMvcSample02.Models.BlogPost>
<ul>
    @foreach (var item in Model)
    {
        <li>
            <h5>
                @Html.ActionLink(linkText: item.Title,
                                 actionName: "Post",
                                 controllerName: "Home",
                                 routeValues: new  { id = item.Id },
                                 htmlAttributes: null)
            </h5>
            @item.Body
        </li>
    }
</ul>
```

ب) یک View کامل که در بار اول نمایش صفحه، مورد استفاده قرار می‌گیرد:

```
@model IList<jQueryMvcSample02.Models.BlogPost>
@{
    ViewBag.Title = "Index";
    var loadInfoUrl = Url.Action(actionName: "PagedIndex", controllerName: "Home");
}
<h2>
    اسکرول نامحدود</h2>
@{ Html.RenderPartial("_ItemsList", Model); }
<div id="MoreInfoDiv">
</div>
<div align="center" style="margin-bottom: 9px;">
    <span id="moreInfoButton" style="width: 90%;" class="btn btn-info">بیشتر</span>
</div>
<div id="ProgressDiv" align="center" style="display: none">
    <br />
    <img src="@Url.Content("~/Content/images/loadingAnimation.gif")" alt="loading..."  />
</div>
@section JavaScript
{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#moreInfoButton").InfiniteScroll({
                moreInfoDiv: '#MoreInfoDiv',
```

```
                progressDiv: '#ProgressDiv',
                loadInfoUrl: '@loadInfoUrl',
                loginUrl: '/login',
                errorHandler: function () {
                    alert('خطایی رخ داده است');
                },
                completeHandler: function () {
                    // اگر قرار است روی اطلاعات نمایش داده شده پردازش ثانوی صورت گیرد
                },
                noMoreInfoHandler: function () {
                    alert('اطلاعات بیشتری یافت نشد');
                }
            });
        });
    </script>
}
```

چند نکته در اینجا حائز اهمیت است:

1) مسیر دقیق اکشن متد PagedIndex توسط متد Url.Action تهیه شده است.

2) در ابتدای نمایش صفحه، متد Html.RenderPartial کار نمایش اولیه اطلاعات را انجام خواهد داد.

3) از div خالی MoreInfoDiv، به عنوان محل افزوده شدن اطلاعات Ajax ایی دریافتی استفاده می‌کنیم.

4) دکمه بیشتر در اینجا تنها یک span ساده است که توسط css به شکل یک دکمه نمایش داده خواهد شد (فایل‌های آن در پروژه پیوست موجود است).

5) ProgressDiv در ابتدای نمایش صفحه مخفی است. زمانیکه کاربر بر روی دکمه بیشتر کلیک می‌کند، توسط افزونه جی‌کوئری ما نمایان شده و در پایان کار مجددا مخفی می‌گردد.

6) section JavaScript کار استفاده از افزونه InfiniteScroll را انجام می‌دهد.

و کدهای افزونه اسکرول نامحدود

```
// <![CDATA[
(function ($) {
    $.fn.InfiniteScroll = function (options) {
        var defaults = {
            moreInfoDiv: '#MoreInfoDiv',
            progressDiv: '#Progress',
            loadInfoUrl: '/',
            loginUrl: '/login',
            errorHandler: null,
            completeHandler: null,
            noMoreInfoHandler: null
        };
        var options = $.extend(defaults, options);

        var showProgress = function () {
            $(options.progressDiv).css("display", "block");
        }

        var hideProgress = function () {
            $(options.progressDiv).css("display", "none");
        }

        return this.each(function () {
            var moreInfoButton = $(this);
            var page = 1;
            $(moreInfoButton).click(function (event) {
                showProgress();
                $.ajax({
                    type: "POST",
                    url: options.loadInfoUrl,
                    data: JSON.stringify({ page: page }),
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    complete: function (xhr, status) {
                        var data = xhr.responseText;
                        if (xhr.status == 403) {
                            window.location = options.loginUrl;
                        }
                        else if (status === 'error' || !data) {
                            if (options.errorHandler)
```

```
                                options.errorHandler(this);
                        }
                        else {
                            if (data == "no-more-info") {
                                if (options.noMoreInfoHandler)
                                    options.noMoreInfoHandler(this);
                            }
                            else {
                                var $boxes = $(data);
                                $(options.moreInfoDiv).append($boxes);
                            }
                            page++;
                        }
                        hideProgress();
                        if (options.completeHandler)
                            options.completeHandler(this);
                    }
                });
            });
        });
    };
})(jQuery);
// ]]>
```

ساختار افزونه اسکرول نامحدود به این شرح است:
هربار که کاربر بر روی دکمه بیشتر کلیک می‌کند، progress div ظاهر می‌گردد. سپس توسط امکانات jQuery Ajax، شماره صفحه (بازه انتخابی) به اکشن متد صفحه بندی اطلاعات ارسال می‌گردد. در نهایت اطلاعات را از کنترلر دریافت و به moreInfoDiv اضافه می‌کند. در آخر هم شماره صفحه را یکی افزایش داده و سپس progress div را مخفی می‌کند.

دریافت مثال و پروژه کامل این قسمت
jQueryMvcSample02.zip

نظرات خوانندگان

نویسنده: Information
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۴:۴۲

به نظر شما این دکمه برای سئو سایت مضر نیست؟

نویسنده: ناصر فرجی
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۴:۴۵

آقای نصیری همون طور که میدونید این روش با seo مشکل داره. برای رفع این مشکل به نظر شما چه کاری میشه انجام داد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۵:۷

کاری که من انجام می‌دم قرار دادن دو لینک در بالای هر مطلب است (لینک به مطلب بعدی و مطلب قبلی):

به این ترتیب موتور جستجو از یک مطلب شروع می‌کند و به راحتی تا آخرین مطلب سایت را می‌تواند پیمایش کند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۵:۱۴

الان اگر به سایت رسمی wordpress مراجعه کنید، اکثر وبلاگ‌های آن به این ترتیب طراحی و ارائه می‌شن. ولی برای حفظ SEO، از تعبیه لینک به مطلب بعدی و قبلی استفاده می‌کنند؛ تا موتور جستجو به سادگی راه خودش را پیدا کند.

نویسنده: ناصر فرجی
تاریخ: ۱۳۹۲/۰۱/۰۴ ۲۰:۵۲

ممنون از پاسختون. چون سایت شما حالتی مثل یک وبلاگ داره میشه از این روش استفاده کرده. اما خیلی از سایت‌ها نمیشه این امکان رو گذاشت! خوشحال میشم راجع به حل معضلی به اسم seo در ajax مقالات بیشتری رو از شما یا سایر دوستان ببینیم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۲۱:۲۶

نهایتا یکی از اهدف مهم SEO این است که یک ربات بتواند سایت را راحت پیمایش کند. خیلی از سایت‌های دیگر هم برای این منظور از مفهومی به نام « site map » استفاده می‌کنند که به اکثر مداخل مهم سایت لینک دارد.

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۲/۰۱/۱۱ ۹:۳۵

با تشکر از مطالب مفید شما
تقریبا اکثر امکاناتی که در سایت خودتان استفاده کردید را در این دوره به اشتراک گذاشتید که بسیار ارزشمند و کاربردی هستند که این جای بسی تشکر و قدردانی را دارد
یک خواهش هم بنده داشتم آن هم اینکه در مورد نمایش و نحوه کاربردی کردن پیام همین سایت یک مطلب ارائه دهید (البته مطالب مشابهی در همین سایت ^ و ^ و ^ وجود دارند) اما به زیبایی کار شما نیستند؟

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۱ ۹:۵۴

من در این سایت بجای alert معمولی از افزونه jQuery noty استفاده کردم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۲۹ ۱۹:۱۳

نگارش جدید این افزونه و مثال را از اینجا می‌توانید دریافت کنید: jQueryMvcSample02_V2.zip
تغییرات:
- اضافه شدن history مشاهده صفحات جدید به دکمه back مرورگر. برای اینکار از افزونه Path.Js کمک گرفته شد. این مورد همچنین سبب می‌شود بتوان آدرس صفحه جاری را ذخیره و بعدا بازیابی کرد.
- اضافه شدن دو دراپ داون برای مرتب سازی بر اساس یک سری فیلد به صورت صعودی و یا نزولی
- محو دکمه بیشتر در زمان کلیک بر روی آن جهت جلوگیری از کلیک‌های بیش از حد با سرعت دریافت پایین اینترنت.

نویسنده: حامد شاه محمدی
تاریخ: ۱۳۹۳/۰۹/۱۴ ۱۲:۲۱

سلام از زحماتی که کشیدید ممنونم
طبق ورژن دوم پروژه ای که گذاشتید عمل کردم منتهی DataSource رو به بانک وصل کردم ولی وقتی اجرا میکنم Loading اجراست فقط هیچ دستکاری دیگه ای هم نشده میشه کمکم کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۴ ۱۲:۲۴

با فایرباگ بررسی کنید چه خطایی گرفتید .

نویسنده: حسین پاکدل
تاریخ: ۱۳۹۴/۰۱/۲۱ ۱۲:۵۰

سلام
چطور میشه بجای استفاده از دکمه جهت افزایش موارد نمایشی، صرفاً وقتی به پایین صفحه میرسیم بطور خودکار موارد جدید اضافه بشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۱ ۱۴:۵۵

رویداد اسکرول را باید تحت نظر قرار داد و هر زمانیکه برای مثال از یک المان خاص رد شد، کدهای رویداد کلیک را فراخوانی کند:

```
$(function(){
  $(window).scroll(function(){
    var aTop = $('.elem').height();
    if($(this).scrollTop()>=aTop){
        //todo: ....
    }
  });
});
```

نویسنده: صادق نجاتی
تاریخ: ۱۳۹۴/۰۱/۲۲ ۱۳:۵۰

```
$(function () {
            $(window).scroll(function () {
                var aTop = $(document).height() - $(window).scrollTop() - $(window).height();
                if (aTop == 0) {
                    alert("ok");
                }
            });
        });
```

با کد بالا به محض اینکه اسکرول به پایین صفحه برسه alert("ok"); اجرا میشه