عنوان: **صفحه بندی پویا در Entity Framework**
نویسنده: محمد رضا صفری
تاریخ: ۱۳۹۳/۰۱/۲۶ ۹:۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Entity framework, Paging

در اکثر برنامه‌ها ما نیازمند این موضوع هستیم که بتوانیم اطلاعاتی را به کاربر نشان دهیم. در بعضی از موارد این اطلاعات بسیار زیاد هستند و نیاز است در این حالت از صفحه بندی اطلاعات یا Data Paging استفاده کنیم. در ASP.NET برای ارائه اطلاعات به کاربر معمولا از کنترلهای ListView ، Gridview و امثالهم استفاده می‌شود. مشکل اساسی این کنترل‌ها این است که آنها اطلاعات را به صورت کامل از سرور دریافت کرده، سپس اقدام به نمایش صفحه بندی شده آن می‌نمایند که این موضوع باعث استفاده بی مورد از حافظه سرور شده و هزینه زیادی برای برنامه ما خواهد داشت.
صفحه بندی در سطح پایگاه داده بهترین روش برای استفاده بهینه از منابع است. برای رسیدن به این مقصود ما نیاز به یک کوئری خواهیم داشت که فقط همان صفحه مورد نیاز را به کنترلر تحویل دهد.
با استفاده از متد توسعه یافته زیر می‌توان به این مقصود دست یافت:

```
/// <summary>
/// صفحه بندی کوئری
/// </summary>
/// <param name="query">کوئری مورد نظر شما</param>
/// <param name="pageNum">شماره صفحه</param>
/// <param name="pageSize">سایز صفحه</param>
/// <param name="orderByProperty">ترتیب خواص</param>
/// <param name="isAscendingOrder">اگر برابر با <c>true</c> است صعودی باشد</param>
/// <param name="rowsCount">تعداد کل ردیف ها</param>
/// <returns></returns>
private static IQueryable<T> PagedResult<T, TResult>(IQueryable<T> query, int pageNum, int pageSize,
                Expression<Func<T, TResult>> orderByProperty, bool isAscendingOrder, out int rowsCount)
{
    if (pageSize <= 0) pageSize = 20;

    //مجموع ردیف‌های به دست آمده
    rowsCount = query.Count();

// اگر شماره صفحه کوچکتر از 0 بود صفحه اول نشان داده شود
    if (rowsCount <= pageSize || pageNum <= 0) pageNum = 1;

// محاسبه ردیف هایی که نسبت به سایز صفحه باید از آنها گذشت
    int excludedRows = (pageNum - 1) * pageSize;

    query = isAscendingOrder ? query.OrderBy(orderByProperty) :
query.OrderByDescending(orderByProperty);

// ردشدن از ردیف‌های اضافی و  دریافت ردیف‌های مورد نظر برای صفحه مربوطه
    return query.Skip(excludedRows).Take(pageSize);
}
```

نحوه استفاده :
فرض کنید که کوئری مورد نظر قرار است تا یکسری از مطالب را از جدول Articles نمایش دهد. برای دریافت 20 ردیف اول جهت استفاده در صفحه اول، از کد زیر استفاده می‌کنیم :

```
var articles = (from article in Articles
                where article.Author == "Abc"
                select article);

int totalArticles;

var firstPageData =  PagedResult(articles, 1, 20, article => article.PublishedDate, false, out
totalArticles);
```

یا به صورت ساده‌تر و قابل اجرا به صورت کلی‌تر :

```
var context = new AtricleEntityModel();
var query = context.ArticlesPagedResult(articles, <pageNumber>, 20, article => article.PublishedDate,
false, out totalArticles);
```

نظرات خوانندگان

نویسنده: محمد رضا ایزدی
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۱:۰

یه سوالی خط آخر چطوری اجرایی شده
شما تو کانتکس اون متد رو اوردین ؟
+

```
Helper.PagedResult(t, 1, 10, o=>true, false, out i);
```

یه اور لود هم اینطوری میشه براش نوشت اینطوری نیاز نیست حتما order در نظر گرفته شود

نویسنده: میثم99
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۱:۲۳

باسلام
مطلبت بسیار مفیدی بود. فقط اگر بتوانی خود صفحه بندی را هم قرار دهی بسیار عالی میشود.
منظورم چند تا لینک که صفحه اول و آخر و صفحه جاری و تعداد دارد.
ممنون

نویسنده: محسن عباس آبادعربی
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۱:۴۷

ضمن تشکر از مطلب فوق
اگر شما در ASP.net استفاد میکنید میتوانی از کنترل ObjectContainerDataSource استفاده کنی که چند مزیت دارد
1 سرعت بالایی دارد
2 امکان sql cache dependency رو فعال میکنه یعنی فقط در هنگامی که شما اطلاعات رو در داخل گرید لود می‌کنید برای دفعه بعد اگر اطلاعات در دیتابیس تغییری نکرده باشد دیگر به سمت دیتابیس مراجعه نمی‌کند و اطلاعات از cache خوانده می‌شود
2 امکان paging سمت سرور رو به شما میدهد .
3 برای پروژهای با دیتای بزرگ تست شده و جواب داده

نویسنده: میثم99
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۲:۹

در mvc هم یک هلپر برای اینکار ساخته شده است که خیلی خوب کار می‌کند.
ولی منظور من یک ماژول ساده دست ساز بود که با توجه به سایت بتوان به هر شکل دلخواهی آنرا تغییر داد. در بعضی از پروژه‌ها واقعا یک همچین چیزی بدرد می‌خورد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۲:۳۳

برای طراحی Pager سازگار با بوت استرپ این مطلب مفید است:
A simple Bootstrap Pager Html Helper

نویسنده: ایزدی
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۴:۱۸

ObjectContainerDataSource میشه یه توضیحی در موردش بدین یا یه مقاله معرفی کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۴:۲۷

ObjectContainerDataSource Control

نویسنده: محمد رضا صفری
تاریخ: ۱۳۹۳/۰۱/۲۶ ۲۰:۳۲

کسانی که Table بی دردسر و Ajax ی میخوان از این پلاگین استفاده کنند :

/http://www.jtable.org

با MVC هم کاملا سازگار هست و نمونه هم داره .

آموزش کامل : http://www.codeproject.com/Articles/277576/AJAX-based-CRUD-tables-using-ASP-NET-MVC-and-jTa

Datatable هم هست .

نویسنده: محسن عباس آبادعربی
تاریخ: ۱۳۹۳/۰۱/۲۷ ۱۲:۲۲

کنترل ObjectContainerDataSource Control مربوط به فزیم ورک WCSF می‌باشد می‌توانید از سایت مایکروسافت دانلود نمایید.

نویسنده: ایزدی
تاریخ: ۱۳۹۳/۰۶/۰۸ ۱۶:۵۶

سلام
من یک برنامه تولید لایه business نوشتم از کد شما هم استفاده کردم با اجازتون یه تغییر کوچیک دادم توش خواستم اینجا هم بذارم که اگر کسی خواست استفاده کنه

```
private static IQueryable<T> PagedResult<T, TResult>(IQueryable<T> query, int pageNum, int pageSize,
                                                     Expression<Func<T, TResult>>
orderByProperty,
                                                     bool isAscendingOrder, out int rowsCount,
                                                     Expression<Func<T, bool>> whereClause =
null)
        {
            if (pageSize <= 0) pageSize = 20;

            //مجموع ردیف‌های به دست آمده
            rowsCount = query.Count();

            // اگر شماره صفحه کوچکتر از 0 بود صفحه اول نشان داده شود
            if (rowsCount <= pageSize || pageNum <= 0) pageNum = 1;

            // محاسبه ردیف هایی که نسبت به سایز صفحه باید از آنها گذشت
            int excludedRows = (pageNum - 1) * pageSize;

            query = isAscendingOrder ? query.OrderBy(orderByProperty) :
query.OrderByDescending(orderByProperty);

            //جستجو را در صورت لزوم انجام می‌دهد

            query = whereClause == null ? query : query.Where(whereClause);


            // ردشدن از ردیف‌های اضافی و دریافت ردیف‌های مورد نظر برای صفحه مربوطه
            return query.Skip(excludedRows).Take(pageSize);
        }
```

و برای فراخوانی هم اینطور استفاده کردم

```
public static List<t_Prodcts> GetPaging(int currentPage, int pageSize, out int count,
                                                Expression<Func<t_Prodcts, bool>> search = null)
        {
            using (var db = new asusIranDBConnection())
            {
                return PagedResult(db.t_Prodcts, currentPage, pageSize, o => true, false, out count,
search).ToList();

            }
        }
```