عنوان: **یک نکته از ASP.NET 4.5 GridView**
نویسنده: ا.ح.م.د
تاریخ: ۱۳۹۱/۰۶/۲۷ ۲۳:۹
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, GridView, Paging

تا قبل از ASP.NET 4.5 ، هنگام کار با GridView رسم بر این بوده که به خاصیت DataSource ، یک منبع داده (مانند SqlDataSource و ...) را Bind کرده و متد DataBind را صدا نموده و نتیجه نمایش داده می‌شد.
اما با استفاده از ویژگی‌های جدید اضافه شده(هر چند با تأخیر نسبت به Gridهای پیشرفته دیگر) کار با این کنترل راحت‌تر و خواناتر شده است. یکی از این ویژگی‌ها را با هم بررسی می‌کنیم:
با استفاده از ویژگی SelectMethod میتوان متدی را به GridView معرفی کرد که وظیفه منبع داده را انجام داده و هنگام Bind فراخوانی شده و گرید را پر کند:
مثال:

```
<asp:GridView ID="gvCities"
                runat="server"
                AutoGenerateColumns="False"
                ItemType="WebApplication3.City"
                SelectMethod="GetAllCities">
                <Columns>
                    <asp:TemplateField HeaderText="نام">
                        <ItemTemplate><%#: Item.Name %></ItemTemplate>
                    </asp:TemplateField>
                </Columns>
     </asp:GridView>
```

نکته مهم در این کد ItemType است. با استفاده از این خاصیت به جای اینکه مانند قبل نام فیلدهایی که قرار است در گرید نمایش داده شود را بصورت string معرفی کنیم (مثلا در اینجا Eval("Name") ، اگر نام فیلد را غلط بنویسیم هنگام کامپایل خطایی صادر نمی‌شود)، آنرا بصورت Strongly Type نوشته و از اشتباه جلوگیری می‌کنیم.(+)
کد متد:

```
public IQueryable<City> GetAllCities()
        {
            var context = new EFContext();
            var q = from c in context.City
                    orderby c.Name
                    select c;
            return q;
        }
```

و سپس دستور زیر را فراخوانی می‌کنیم:

```
gvCities.DataBind();
```

اگر بخواهیم در گرید Paging داشته باشیم بصورت زیر عمل می‌کنیم:

```
<asp:GridView ID="gvCities"
                runat="server"
                AutoGenerateColumns="False"
                AllowPaging="True"
                PageSize="10"
                ItemType="WebApplication3.City"
                SelectMethod="GetAllCities">
                 <Columns>
                    <asp:TemplateField HeaderText="نام">
                        <ItemTemplate><%#: Item.Name %></ItemTemplate>
                    </asp:TemplateField>
                </Columns>
  </asp:GridView>
```

که در اینجا دو خصوصیت AllowPaging و PageSize را مقدار دهی کرده ایم. این خصوصیت‌ها اجازه صفحه بندی را به گرید

می‌دهند.حال برای اینکه متد نیز برای صفحه بندی آماده شود باید سه آرگومان به آن اضافه کنیم:(نام پارامترها باید دقیقا موارد زیر باشد)

1- startRowIndex: نقطه شروع صفحه بندی را مشخص می‌کند.

2- maximumRows: تعداد سطرهایی که گرید باید نمایش دهد را مشخص می‌کند.

3- totalRowCount: این پارامتر باید در تابع مقدار دهی شود (مانند مثال) تا مشخص شود نتیجه Query چند رکورد است و در نهایت گرید تعداد صفحات را بر این اساس نمایش می‌دهد.

و برای اینکه صفحه بندی را در Query هم لحاظ کنیم از دو تا بع Skip و Take استفاده شده است.

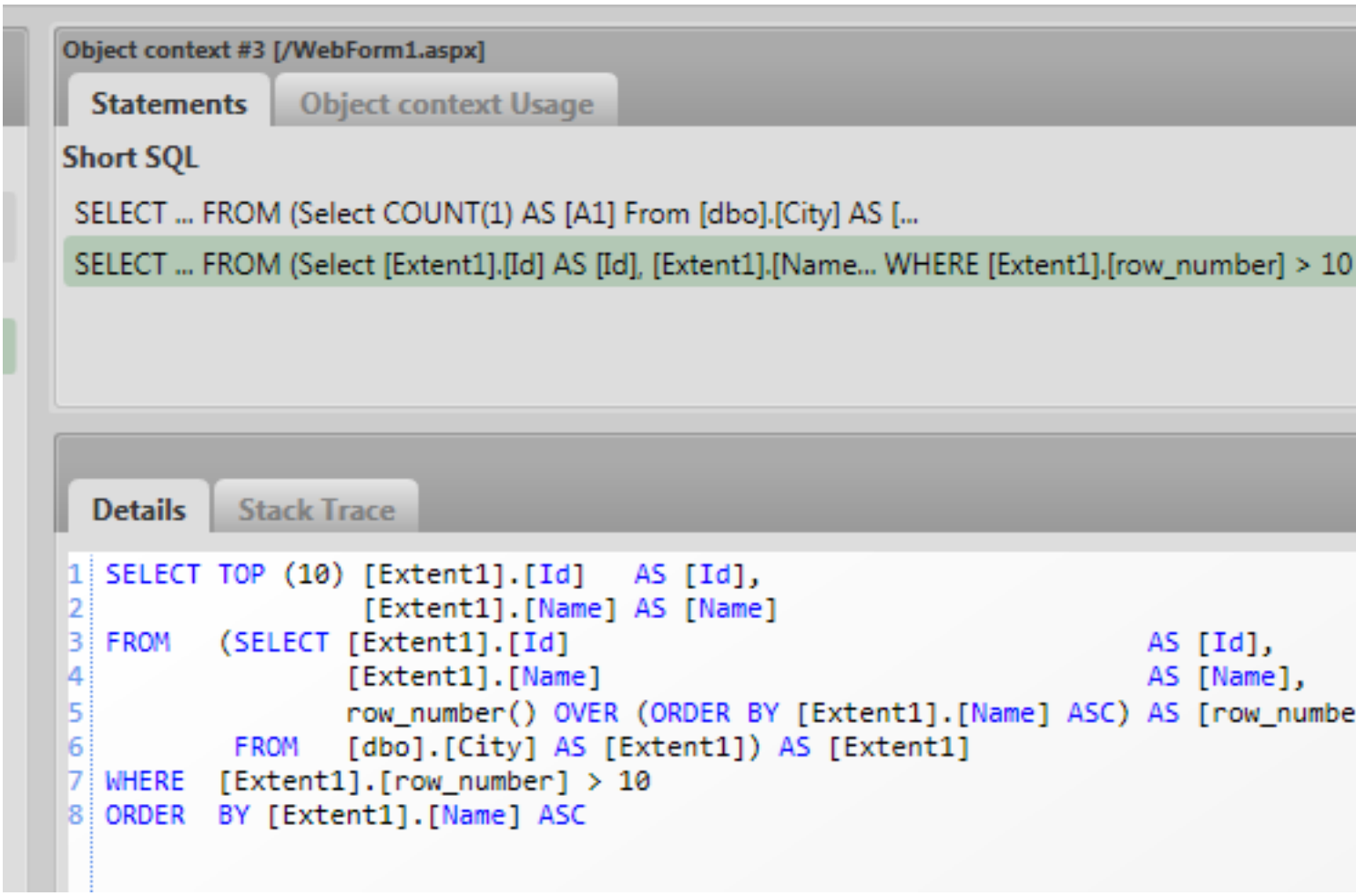
```
public IQueryable<City> GetAllCities(int startRowIndex, int maximumRows, out int totalRowCount)
 {
        var context = new EFContext();
        var q = from c in context.City
                    select c;

        totalRowCount = q.Count();

        return q.OrderBy(x=>x.Name).Skip(startRowIndex).Take(maximumRows);
 }
```

نکته مهم در این متد IQueryable بودن آن است که باعث واکشی داده‌ها بصورت صفحه به صفحه میشود.

دستورات SQL تولید شده در پروفایلر:



همانطور که مشاهده می‌کنید دو دستور SQL تولید شده ، یکی برای بازگرداندن تعداد رکوردها و یکی هم برای واکشی داده‌ها به

اندازه تعداد رکوردهای مجاز در هر صفحه.

نظرات خوانندگان

نویسنده: saeid
تاریخ: ۱۳۹۱/۰۶/۳۰ ۱:۲۲

با سلام خدمت دوست عزیزم و تشکر از زحمت شما. سوالم این بود که واقعا این مورد در ورژن‌های قبلی امکان پذیر نبوده؟!
http://books.google.com/books?id=wAsBOIZFXV4C&pg=PA338&lpg=PA338&dq=SelectMethod+gridview+asp.net+startRowIndex+maximumRows+totalRowCount&source=bl&ots=sA1GqNuQr9&sig=vzOaA6EQ6h1w3iOTQ45dppyMwtw&hl=en&sa=X&ei=yztaUKTFJYSJhQf7o4CoDQ&ved=0CFAQ6AEwBQ#v=onepage&q=SelectMethod%20gridview%20asp.net%20startRowIndex%20maximumRows%20totalRowCount&f=false

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۳۰ ۸:۶

نه به صورت Strongly Type Binding که توضیح داده شد.

نویسنده: saeid
تاریخ: ۱۳۹۱/۰۶/۳۱ ۱۳:۲۴

بله درسته. ممنون از لینکی که به اشتراک گذاشتید. راستش مطلب رو بطور کامل نخونده بودم.

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();

            }
        }
```

عنوان: **پیاده سازی عملیات صفحه بندی (paging) در sql server**
نویسنده: محمد حسین عزتی
تاریخ: ۱۳۹۳/۰۷/۲۲ ۱۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, Paging, Performance

در خیلی مواقع ملاحظه میشود که برای نمایش تعدادی از رکوردهای یک جدول در پایگاه داده، کل مقادیر موجود درآن توسط یک دستور select به دست می‌آید و صفحه‌بندی خروجی، به کنترلهای موجود سپرده میشود. اگر پایگاه داده ما دارای تعداد زیادی رکورد باشد، آن موقع است که دچار مشکل می‌شویم. فرض کنید به طور همزمان ۵ نفر (که تعداد زیادی نیستند) از برنامه ما که شامل ۱۰۰۰۰۰ سطر داده میباشد استفاده کنند و در هر صفحه، ۱۰ رکورد نمایش داده شود و صفحه‌بندی ما از نوع معقولی نباشد. در این صورت به جای اینکه با ۵×۱۰ رکورد داده را بارگزاری کنیم، ۵×۱۰۰۰۰۰ رکورد یعنی ۵۰۰۰۰۰ رکورد را برای به دست آوردن ۵۰ رکورد بارگزاری میکنیم. در زیر روشی شرح داده میشود که توسط آن، این سربار اضافه از روی برنامه و سرورهای مربوطه حذف شود. به stored procedure و توضیحات مربوط به آن توجه فرمایید :

```
CREATE PROCEDURE sp_PagedItems
(
 @Page int,
 @RecsPerPage int
)
AS

-- We don't want to return the # of rows inserted
-- into our temporary table, so turn NOCOUNT ON
SET NOCOUNT ON


--Create a temporary table
CREATE TABLE #TempItems
(
ID int IDENTITY,
Name varchar(50),
Price currency
)


-- Insert the rows from tblItems into the temp. table
INSERT INTO #TempItems (Name, Price)
SELECT Name,Price FROM tblItem ORDER BY Price

-- Find out the first and last record we want
DECLARE @FirstRec int, @LastRec int
SELECT @FirstRec = (@Page - 1) * @RecsPerPage
SELECT @LastRec = (@Page * @RecsPerPage + 1)

-- Now, return the set of paged records, plus, an indiciation of we
-- have more records or not!
SELECT *,
MoreRecords =
(
 SELECT COUNT(*)
 FROM #TempItems TI
 WHERE TI.ID >= @LastRec
)
FROM #TempItems
WHERE ID > @FirstRec AND ID < @LastRec


-- Turn NOCOUNT back OFF
SET NOCOUNT OFF
```

در این کد دو پارامتر از نوع integer تعریف میکنیم. اول پارامتر @Page که مربوط به شماره صفحه‌ای می‌باشد که قصد دارید آن‌را بارگزاری نمایید. دومین پارامتر با نام @RecsPerPage تعداد رکوردهایی است که هر بار میخواهید بارگزاری شوند. مثلا اگر میخواهید هر بار ۱۵ عدد از رکوردها را نمایش دهید، این مقدار را باید برابر ۱۵ قرار دهیم. در مرحله بعد یک جدول موقت با نام #TempItems ساخته شده است که به طور موقت مقادیری را در حافظه نگه میدارد. نکته کلیدی که جلوتر از آن استفاده شده، ستون با نام ID است که از نوع auto-increment بوده و روی جدول موقت تعریف شده است. این ستون شناسه هر سطر را در خود نگه میدارد که به صورت اتوماتیک بالا میرود و جزء لاینفکی از این نوع paging میباشد. پس از آن جدول موقت را توسط

رکوردهای جدول واقعی با نام tblItem توسط دستور select پر میکنیم.

در مرحله بعد شماره اولین و آخرین سطر مورد نظر را بر اساس پارامترهای ورودی محاسبه کرده و در متغیرهای @FirstRec و @LastRec می‌ریزیم.
برای استفاده از این کد فقط کافیست که پارامترهای ورودی را مقداردهی نمایید. مثلا اگر میخواهید در یک کنترل Grid از آن استفاده کنید باید ابتدا یک کوئری داشته باشید که تعداد کل سطرها را به شما بدهد و بر اساس این مقدار تعداد صفحات مورد نظر را به دست آورید. پس از آن با کلیک روی هر کدام از شماره صفحات آن را به عنوان مقدار به پارامتر مورد نظر بفرستید و از آن لذت ببرید.

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۷/۲۲ ۱۹:۵۹

ضمن تشکر از شما. یک اصلاح کوچک: جدول موقتی ایجاد شده در پایان کار رویه ذخیره شده باید drop بشه.

نویسنده: محمد حسین عزتی
تاریخ: ۱۳۹۳/۰۷/۲۲ ۲۰:۱۸

از دقت شما به این نکته ظریف ممنونم
این موضوع در راستای آموزش عنوان مطلبش بود اما نکته شما جهت بالا بردن کیفیت و بهینه کردن کد مورد استفاده قرار میگیرد و عدم drop مشکلی در رسیدن به هدف مورد نظر ایجاد نمی‌کند
متشکرم

نویسنده: امید
تاریخ: ۱۳۹۳/۰۷/۲۲ ۲۱:۲۰

در sql 2012 به بعد جهت صفحه بندی دستورات offset و fetch اضافه شده که از لحاظ Performance بهینه‌تر از باقی روش‌های میباشد . مقایسه صفحه بندی‌های مختلف

نویسنده: محمد حسین عزتی
تاریخ: ۱۳۹۳/۰۷/۲۲ ۲۱:۵۴

ممنونم بخاطر لینک مفیدی که در قسمت نظر ارسال نمودید
تنها نقطه ضعف این مقاله همینطور که خود شما هم متذکر شده اید این است که برای ورژن‌های بانک اطلاعاتی بعد از 2012 قابل استفاده است. هنوز بسیاری از نرم افزارها و سازمانهای ما هنوز با ورژن‌های قدیمی‌تر کار می‌کنند.

متشکرم

نویسنده: حمیدرضا کاظم نادی
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۱:۲۹

ممنون از مطلب خوبتون
به نظرم اگه جوری Sp را مینوشتید که یک ورودی متنی Query یا یک جدول موقت میگرفت و عمل Paging را روی اون انجام میداد مطلبتون بسیار کامل‌تر بود. مثلا ورودی Sp به این صورت بود که ('select * from Tbl_1',1,10) بازهم ممنون

نویسنده: محمد حسین عزتی
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۲:۵۴

سلام
ممنونم از نظرتون
دوتا پارامتر داره از ورودی دریافت می‌کنه و هدف نحوه انجام صفحه بندی بوده

متشکرم

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۳:۳۶

روشی بسیار قدیمی‌است و این روزها آنچنان کاربرد ندارد.

برای صفحه‌بندی :
- در SQL Server 2008 از ROW_NUMBER استفاده می‌کنند.
- در SQL Server 2012 به بعد از OFFSET FETCH استفاده می‌کنند .

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۴:۱۰

تاریخچه‌ای از روش‌های مختلف صفحه بندی اطلاعات در SQL Server در این مقاله بحث شده به همراه بررسی کارآیی آن‌ها:

Comparing performance for different SQL Server paging methods