

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 Grid های پیشرفته دیگر) کار با این کنترل راحت تر و خوانا تر 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 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 باید تعداد صفحات را بر این اساس نمایش می‌دهد.

و برای اینکه صفحه بندی را در Query هم لحاظ کنیم از دو تابع 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 واکنشی داده‌ها بصورت صفحه به صفحه می‌شود. دستورات SQL تولید شده در پروفایلر:

Object context #3 [/WebForm1.aspx]

Statements Object context Usage

Short SQL

```
SELECT ... FROM (Select COUNT(1) AS [A1] From [dbo].[City] AS [...
SELECT ... FROM (Select [Extent1].[Id] AS [Id], [Extent1].[Name... WHERE [Extent1].[row_number] > 10
```

Details Stack Trace

```
1 SELECT TOP (10) [Extent1].[Id] AS [Id],
2 [Extent1].[Name] AS [Name]
3 FROM (SELECT [Extent1].[Id] AS [Id],
4 [Extent1].[Name] AS [Name],
5 row_number() OVER (ORDER BY [Extent1].[Name] ASC) AS [row_numbe
6 FROM [dbo].[City] AS [Extent1]) AS [Extent1]
7 WHERE [Extent1].[row_number] > 10
8 ORDER BY [Extent1].[Name] ASC
```

همانطور که مشاهده می‌کنید دو دستور SQL تولید شده ، یکی برای بازگرداندن تعداد رکوردها و یکی هم برای واکنشی 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=wAsB0IZFXV4C&pg=PA338&lpg=PA338&dq=SelectMethod+gridview+asp.net+startRowIndex+maximumRows+totalRowCount&source=bl&ots=sA1GqNuQr9&sig=vz0aA6EQ6h1w3i0TQ45dppyMwtw&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 ۸:۰۶

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.

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 معمولاً 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"><c>true</c> اگر برابر با</param>
/// <param name="rowCount">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 rowCount)
{
    if (pageSize <= 0) pageSize = 20;

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

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

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

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

    // رد شدن 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: ۱۱:۰ ۱۳۹۳/۰۱/۲۶

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

```
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: میثم 99

تاریخ: ۱۱:۲۳ ۱۳۹۳/۰۱/۲۶

باسلام
مطلب 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: ۱۱:۴۷ ۱۳۹۳/۰۱/۲۶

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 می کنی برای دفعه بعد اگر اطلاعات در دیتابیس تغییری نکرده باشد دیگر به سمت دیتابیس مراجعه نمی کند و اطلاعات از cache خوانده می شود
2 امکان paging سمت سرور رو به شما میدهد .
3 برای پروژهای با دیتای بزرگ تست شده و جواب داده

نویسنده: میثم 99

تاریخ: ۱۲:۰۹ ۱۳۹۳/۰۱/۲۶

در mvc هم یک هیلپر برای اینکار ساخته شده است که خیلی خوب کار می کند.
ولی منظور من یک مازول ساده دست ساز بود که با توجه به سایت بتوان به هر شکل دلخواهی آنرا تغییر داد. در بعضی از پروژه ها واقعا یک همچین چیزی بدرد می خورد

نویسنده: وحید نصیری

تاریخ: ۱۲:۳۳ ۱۳۹۳/۰۱/۲۶

برای طراحی Pager سازگار با بوت استرپ این مطلب مفید است:

[A simple Bootstrap Pager Html Helper](#)

نویسنده: ایزدی

تاریخ: ۱۴:۱۸ ۱۳۹۳/۰۱/۲۶

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

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

[ObjectContainerDataSource Control](#)

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

کسانی که Table بی دردرس و Ajax ی میخوان از این پلاگین استفاده کنند :
[/http://www.jtable.org](http://www.jtable.org)

با MVC هم کاملاً 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: ۱۲:۲۲ ۱۳۹۳/۰۱/۲۷

کنترل [ObjectContainerDataSource Control](#) مربوط به فزیم ورک WCSF می باشد می توانید از سایت میکروسافت دانلود نمایید.

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

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 rowCount,
Expression<Func<T, bool>> whereClause =
null)
{
    if (pageSize <= 0) pageSize = 20;
    //مجموع ردیف های به دست آمده
    rowCount = query.Count();

    // اگر شماره صفحه کوچکتر از 0 بود صفحه اول نشان داده شود
    if (rowCount <= 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 می دهد
    query = whereClause == null ? query : query.Where(whereClause);

    // رد شدن از ردیف های اضافی و دریافت ردیف های مورد نظر برای صفحه مربوطه
    return query.Skip(excludedRows).Take(pageSize);
}
```

و برای فراخوانی هم اینطور استفاده کردم

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