عنوان: **ASP.NET MVC #20**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۰۳ ۱۸:۰۹:۰۰
آدرس: www.dotnettips.info
گروه‌ها: MVC

تهیه گزارشات تحت وب به کمک WebGrid

WebGrid از ASP.NET MVC 3.0 به صورت توکار به شکل یک Html Helper در دسترس می‌باشد و هدف از آن ساده‌تر سازی تهیه گزارشات تحت وب است. البته این گرید، تنها گرید مهیای مخصوص ASP.NET MVC نیست و پروژه MVC Contrib یا شرکت Telerik نیز نمونه‌های دیگری را ارائه داده‌اند؛ اما از این جهت که این Html Helper، بدون نیاز به کتابخانه‌های جانبی در دسترس است، بررسی آن ضروری می‌باشد.

صورت مساله

لیستی از کارمندان به همراه حقوق ماهیانه آن‌ها در دست است. اکنون نیاز به گزارشی تحت وب، با مشخصات زیر می‌باشد:
-1 گزارش باید دارای صفحه بندی بوده و هر صفحه تنها 10 ردیف را نمایش دهد.
-2 سطرها باید یک در میان دارای رنگی متفاوت باشند.
-3 ستون حقوق کارمندان در پایین هر صفحه، باید دارای جمع باشد.
-4 بتوان با کلیک بر روی عنوان هر ستون، اطلاعات را بر اساس ستون انتخابی، مرتب ساخت.
-5 لینک‌های حذف یا ویرایش یک ردیف نیز در این گزارش مهیا باشد.
-6 لیست تهیه شده، دارای ستونی به نام «ردیف» نیست. این ستون را نیز به صورت خودکار اضافه کنید.
-7 لیست نهایی اطلاعات، دارای ستونی به نام مالیات نیست. فقط حقوق کارمندان ذکر شده است. ستون محاسبه شده مالیات نیز باید به صورت خودکار در این گزارش نمایش داده شود. این ستون نیز باید دارای جمع پایین هر صفحه باشد.
-8 تمام اعداد این گزارش در حین نمایش باید دارای جدا کننده سه رقمی باشند.
-9 تاریخ‌های موجود در لیست، میلادی هستند. نیاز است این تاریخ‌ها در حین نمایش شمسی شوند.
-10 انتهای هر صفحه گزارش باید بتوان برچسب «صفحه y/n» را مشاهده کرد. n در اینجا منظور تعداد کل صفحات است و y شماره صفحه جاری می‌باشد.
-11 انتهای هر صفحه گزارش باید بتوان برچسب «رکوردهای y تا x از n» را مشاهده کرد. n در اینجا منظور تعداد کل رکوردها است.
-12 نام کوچک هر کارمند، ضخیم نمایش داده شود.
-13 به ازای هر شماره کارمندی، یک تصویر در پوشه images سایت وجود دارد. برای مثال images/id.jpg. ستونی برای نمایش تصویر متناظر با هر کارمند نیز باید اضافه شود.
-14 به ازای هر کارمند، تعدادی پروژه هم وجود دارد. پروژه‌های متناظر را توسط یک گرید تو در تو نمایش دهید.

راه حل به کمک استفاده از WebGrid

ابتدا یک پروژه خالی ASP.NET MVC را آغاز کنید. سپس مدل‌های زیر را به آن اضافه نمائید (یک کارمند که می‌تواند تعداد پروژه منتسب داشته باشد):

```
using System;
using System.Collections.Generic;

namespace MvcApplication17.Models
{
    public class Employee
    {
        public int Id { set; get; }
```

```
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public DateTime AddDate { get; set; }
        public double Salary { get; set; }
        public IList<Project> Projects { get; set; }
    }
}
```

```
namespace MvcApplication17.Models
{
    public class Project
    {
        public int Id { set; get; }
        public string Name { set; get; }
    }
}
```

سپس منبع داده نمونه زیر را به پروژه اضافه کنید. به عمد از ORM خاصی استفاده نشده تا بتوانید پروژه جاری را به سادگی در یک پروژه آزمایشی جدید، تکرار کنید.

```
using System;
using System.Collections.Generic;

namespace MvcApplication17.Models
{
    public static class EmployeeDataSource
    {
        public static IList<Employee> CreateEmployees()
        {
            var list = new List<Employee>();
            var rnd = new Random();
            for (int i = 1; i <= 1000; i++)
            {
                list.Add(new Employee
                    {
                        Id = i + 1000,
                        FirstName = "fName " + i,
                        LastName = "lName " + i,
                        AddDate = DateTime.Now.AddYears(-rnd.Next(1, 10)),
                        Salary = rnd.Next(400, 3000),
                        Projects = CreateRandomProjects()
                    });
            }
            return list;
        }

        private static IList<Project> CreateRandomProjects()
        {
            var list = new List<Project>();
            var rnd = new Random();
            for (int i = 0; i < rnd.Next(1, 7); i++)
            {
                list.Add(new Project
                {
                    Id = i,
                    Name = "Project " + i
                });
            }
            return list;
        }
    }
}
```

در ادامه یک کنترلر جدید را با محتوای زیر اضافه نمائید:

```
using System.Web.Mvc;
```

```
using MvcApplication17.Models;

namespace MvcApplication17.Controllers
{
    public class HomeController : Controller
    {
        [HttpPost]
        public ActionResult Delete(int? id)
        {
            return RedirectToAction("Index");
        }

        [HttpGet]
        public ActionResult Edit(int? id)
        {
            return View();
        }

        [HttpGet]
        public ActionResult Index(string sort, string sortdir, int? page = 1)
        {
            var list = EmployeeDataSource.CreateEmployees();
            return View(list);
        }
    }
}
```

علت تعریف متد index با پارامترهای sort و غیره به URLهای خودکاری از نوع زیر بر می‌گردد:

```
http://localhost:3034/?sort=LastName&sortdir=ASC&page=3
```

همانطور که ملاحظه می‌کنید، گرید رندر شده، از یک سری کوئری استرینگ برای مشخص سازی صفحه جاری، یا جهت مرتب سازی (صعودی و نزولی بودن آن) یا فیلد پیش فرض مرتب سازی، کمک می‌گیرد.

سپس یک View خالی را نیز برای متد Index ایجاد کنید. تا اینجا تنظیمات اولیه پروژه انجام شد.
کدهای کامل View را در ادامه ملاحظه می‌کنید:

```
@using System.Globalization
@model IList<MvcApplication17.Models.Employee>

@{
    ViewBag.Title = "Index";
}

@helper WebGridPageFirstItem(WebGrid grid)
{
    @(((grid.PageIndex + 1) * grid.RowsPerPage) - (grid.RowsPerPage - 1));
}

@helper WebGridPageLastItem(WebGrid grid)
{
    if (grid.TotalRowCount < (grid.PageIndex + 1 * grid.RowsPerPage))
    {
      @grid.TotalRowCount;
    }
    else
    {
      @((grid.PageIndex + 1) * grid.RowsPerPage);
    }
}

<h2>Employees List</h2>

@{
    var grid = new WebGrid(
```

```
        source: Model,
        canPage: true,
        rowsPerPage: 10,
        canSort: true,
        defaultSort: "FirstName"
    );
    var salaryPageSum = 0;
    var taxPageSum = 0;
    var rowIndex = ((grid.PageIndex + 1) * grid.RowsPerPage) - (grid.RowsPerPage - 1);
}

<div id="container">
    @grid.GetHtml(
            tableStyle: "webgrid",
            headerStyle: "webgrid-header",
            footerStyle: "webgrid-footer",
            alternatingRowStyle: "webgrid-alternating-row",
            selectedRowStyle: "webgrid-selected-row",
            rowStyle: "webgrid-row-style",
            htmlAttributes: new { id = "MyGrid" },
            mode: WebGridPagerModes.All,
            columns: grid.Columns(
                 grid.Column(header: "#",
                             style: "text-align-center-col",
                             format: @<text>@(rowIndex++)</text>),
                 grid.Column(columnName: "FirstName", header: "First Name",
                             format: @<span style='font-weight: bold'>@item.FirstName</span>,
                             style: "text-align-center-col"),
                 grid.Column(columnName: "LastName", header: "Last Name"),
                 grid.Column(header: "Image",
                             style: "text-align-center-col",
                             format: @<text><img alt="@item.Id" src="@Url.Content("~/images/" +
@item.Id + ".jpg")" /></text>),
                 grid.Column(columnName: "AddDate", header: "Start",
                             style: "text-align-center-col",
                             format: item =>
                             {
                                 int ym = item.AddDate.Year;
                                 int mm = item.AddDate.Month;
                                 int dm = item.AddDate.Day;
                                 var persianCalendar = new PersianCalendar();
                                 int ys = persianCalendar.GetYear(new DateTime(ym, mm, dm, new
GregorianCalendar()));
                                 int ms = persianCalendar.GetMonth(new DateTime(ym, mm, dm, new
GregorianCalendar()));
                                 int ds = persianCalendar.GetDayOfMonth(new DateTime(ym, mm, dm, new
GregorianCalendar()));
                                 return ys + "/" + ms.ToString("00") + "/" + ds.ToString("00");

                             }),
                 grid.Column(columnName: "Salary", header: "Salary",
                             format: item =>
                             {
                                 salaryPageSum += item.Salary;
                                 return string.Format("${0:n0}", item.Salary);
                             },
                             style: "text-align-center-col"),
                 grid.Column(header: "Tax", canSort: true,
                             format: item =>
                             {
                                 var tax = item.Salary * 0.2;
                                 taxPageSum += tax;
                                 return string.Format("${0:n0}", tax);
                             }),
                 grid.Column(header: "Projects", columnName: "Projects",
                             style: "text-align-center-col",
                             format: item =>
                             {
                                 var subGrid = new WebGrid(
                                                         source: item.Projects,
                                                         canPage: false,
                                                         canSort: false
                                                         );
                                 return subGrid.GetHtml(
                                                         htmlAttributes: new { id = "MySubGrid" },
                                                         tableStyle: "webgrid",
                                                         headerStyle: "webgrid-header",
                                                         footerStyle: "webgrid-footer",
                                                         alternatingRowStyle: "webgrid-alternating-row",
                                                         selectedRowStyle: "webgrid-selected-row",
                                                         rowStyle: "webgrid-row-style"
```

```
                                    );
                                }),
                grid.Column(header: "",
                            style: "text-align-center-col",
                            format: item => @Html.ActionLink(linkText: "Edit", actionName: "Edit",
                                                             controllerName: "Home", routeValues: new
{ id = item.Id },
                                                             htmlAttributes: null)),
                grid.Column(header: "",
                            format: @<form action="/Home/Delete/@item.Id" method="post"><input
type="submit"
                                          onclick="return confirm('Do you want to delete this
record?');"
                                          value="Delete"/></form>),
                grid.Column(header: "", format: item => item.GetSelectLink("Select"))
                     )
                 )

    <strong>Page:</strong> @(grid.PageIndex + 1) / @grid.PageCount,
    <strong>Records:</strong> @WebGridPageFirstItem(@grid) - @WebGridPageLastItem(@grid) of
@grid.TotalRowCount

    @*
    @if (@grid.HasSelection)
    {
        @RenderPage("~/views/path/_partial_view.cshtml", new { Employee = grid.SelectedRow })
    }
    *@
</div>

@section script{
<script type="text/javascript">
    $(function () {
        $('#MyGrid tbody:first').append(
        '<tr class="total-row"><td></td>\
         <td></td><td></td><td></td>\
         <td><strong>Total:</strong></td>\
         <td>@string.Format("${0:n0}", @salaryPageSum)</td>\
         <td>@string.Format("${0:n0}", @taxPageSum)</td>\
         <td></td><td></td><td></td></tr>');
    });
</script>
}
```

توضیحات ریز جزئیات View فوق

تعریف ابتدایی شیء WebGrid و مقدار دهی آن

در ابتدا نیاز است یک وهله از شیء WebGrid را ایجاد کنیم. در اینجا می‌توان تنظیم کرد که آیا نیاز است اطلاعات نمایش داده شده دارای صفحه بندی (canPage) خودکار باشند؟ منبع داده (source) کدام است. در صورت فعال سازی صفحه بندی خودکار، چه تعداد ردیف (rowsPerPage) در هر صفحه نمایش داده شود. آیا نیاز است بتوان با کلیک بر روی سر ستون‌ها، اطلاعات را بر اساس فیلد متناظر با آن مرتب (canSort) ساخت؟ همچنین در صورت نیاز به مرتب سازی، اولین باری که گرید نمایش داده می‌شود، بر اساس چه فیلدی (defaultSort) باید مرتب شده نمایش داده شود:

```
@{
    var grid = new WebGrid(
        source: Model,
        canPage: true,
        rowsPerPage: 10,
        canSort: true,
        defaultSort: "FirstName"
    );
    var salaryPageSum = 0;
    var taxPageSum = 0;
    var rowIndex = ((grid.PageIndex + 1) * grid.RowsPerPage) - (grid.RowsPerPage - 1);
}
```

در اینجا همچنین سه متغیر کمکی هم تعریف شده که از این‌ها برای تهیه جمع ستون‌های حقوق و مالیات و همچنین نمایش شماره ردیف جاری استفاده می‌شود. فرمول نحوه محاسبه اولین ردیف هر صفحه را هم ملاحظه می‌کنید. شماره ردیف‌های بعدی، rowIndex++ خواهند بود.

تعریف رنگ و لعاب گرید نمایش داده شده

در ادامه به کمک متد grid.GetHtml، رشته‌ای معادل اطلاعات HTML صفحه جاری، بازگشت داده می‌شود. در اینجا می‌توان یک سری خواص تکمیلی را تنظیم نمود. برای مثال:

```
tableStyle: "webgrid",
headerStyle: "webgrid-header",
footerStyle: "webgrid-footer",
alternatingRowStyle: "webgrid-alternating-row",
selectedRowStyle: "webgrid-selected-row",
rowStyle: "webgrid-row-style",
htmlAttributes: new { id = "MyGrid" },
```

هر کدام از این رشته‌ها در حین رندر نهایی گرید، تبدیل به یک class خواهند شد. برای نمونه:

```
<div id="container">
      <table class="webgrid" id="MyGrid">
        <thead>
          <tr class="webgrid-header">
```

به این ترتیب با اندکی ویرایش css سایت، می‌توان انواع و اقسام رنگ‌ها را به سطرها و ستون‌های گرید نهایی اعمال کرد. برای مثال اطلاعات زیر را به فایل css سایت اضافه نمائید:

```
/* Styles for WebGrid
---------------------------------------------------------*/
.webgrid
{
 width: 100%;
 margin: 0px;
 padding: 0px;
 border: 0px;
 border-collapse: collapse;
 font-family: Tahoma;
 font-size: 9pt;
}

.webgrid a
{
 color: #000;
}

.webgrid-header
{
 padding: 0px 5px;
 text-align: center;
 border-bottom: 2px solid #739ace;
 height: 20px;
 border-top: 2px solid #D6E8FF;
 border-left: 2px solid #D6E8FF;
 border-right: 2px solid #D6E8FF;
}

.webgrid-header th
{
 background-color: #eaf0ff;
 border-right: 1px solid #ddd;
}
```

```
.webgrid-footer
{
 padding: 6px 5px;
 text-align: center;
 background-color: #e8eef4;
 border-top: 2px solid #3966A2;
 height: 25px;
 border-bottom: 2px solid #D6E8FF;
 border-left: 2px solid #D6E8FF;
 border-right: 2px solid #D6E8FF;
}

.webgrid-alternating-row
{
 height: 22px;
 background-color: #f2f2f2;
 border-bottom: 1px solid #d2d2d2;
 border-left: 2px solid #D6E8FF;
 border-right: 2px solid #D6E8FF;
}

.webgrid-row-style
{
 height: 22px;
 border-bottom: 1px solid #d2d2d2;
 border-left: 2px solid #D6E8FF;
 border-right: 2px solid #D6E8FF;
}

.webgrid-selected-row
{
 font-weight: bold;
}

.text-align-center-col
{
 text-align: center;
}

.total-row
{
 background-color:#f9eef4;
}
```

همانطور که ملاحظه می‌کنید، رنگ‌های ردیف‌ها، هدر و فوتر گرید و غیره در اینجا تنظیم می‌شوند.
به علاوه اگر دقت کرده باشید در تعاریف گرید، htmlAttributes هم مقدار دهی شده است. در اینجا به کمک یک anonymously typed object، مقدار id گرید مشخص شده است. از این id در حین کار با jQuery استفاده خواهیم کرد.

تعیین نوع Pager

پارامتر دیگری که در متد grid.GetHtml تنظیم شده است، mode: WebGridPagerModes.All می‌باشد. WebGridPagerModes یک enum با محتوای زیر است و توسط آن می‌توان نوع Pager گرید را تعیین کرد:

```
[Flags]
public enum WebGridPagerModes
{
        Numeric = 1,
        //
        NextPrevious = 2,
        //
        FirstLast = 4,
        //
        All = 7,
}
```

نحوه تعریف ستون‌های گرید

اکنون به مهم‌ترین قسمت تهیه گزارش رسیده‌ایم. در اینجا با مقدار دهی پارامتر columns، نحوه نمایش اطلاعات ستون‌های مختلف مشخص می‌گردد. مقداری که باید در اینجا تنظیم شود، آرایه‌ای از نوع WebGridColumn می‌باشد و مرسوم است به کمک متد کمکی grid.Columns، اینکار را انجام داد.
متد کمکی grid.Column، یک وهله از شیء WebGridColumn را بر می‌گرداند و از آن برای تعریف هر ستون استفاده خواهیم کرد. توسط پارامتر columnName آن، نام فیلدی که باید اطلاعات ستون جاری از آن اخذ شود مشخص می‌شود. به کمک پارامتر header، عبارت سرستون متناظر تنظیم می‌گردد. پارامتر format، مهم‌ترین و توانمندترین پارامتر متد grid.Column است:

```
grid.Column(columnName: "FirstName", header: "First Name",
                              format: @<span style='font-weight: bold'>@item.FirstName</span>,
                              style: "text-align-center-col"),
grid.Column(columnName: "LastName", header: "Last Name"),
```

پارامتر format، به نحو زیر تعریف شده است:

```
Func<dynamic, object> format
```

به این معنا که هر بار پیش از رندر سطر جاری، زمانیکه قرار است سلولی رندر شود، یک شیء dynamic در اختیار شما قرار می‌گیرد. این شیء dynamic یک رکورد از اطلاعات Model جاری است. به این ترتیب به اطلاعات تمام سلول‌های ردیف جاری دسترسی خواهیم داشت. بر این اساس هر نوع پردازشی را که لازم بود، انجام دهید (شبیه به فرمول نویسی در ابزارهای گزارش سازی، اما اینبار با کدهای سی شارپ) و مقدار فرمت شده نهایی را به صورت یک رشته بر گردانید. این رشته نهایتا در سلول جاری درج خواهد شد.
اگر از پارامتر فرمت استفاده نشود، همان مقدار فیلد جاری بدون تغییری رندر می‌گردد.
حداقل به دو نحو می‌توان پارامتر فرمت را مقدار دهی کرد:

```
format: @<span style='font-weight: bold'>@item.FirstName</span>
or
format: item =>
                     {
                          salaryPageSum += item.Salary;
                           return string.Format("${0:n0}", item.Salary);
                      }
```

مستقیما از توانمندی‌های Razor استفاده کنید. مثلا یک تگ کامل را بدون نیاز به محصور سازی آن بین "" شروع کنید. سپس @item به وهله‌ای از رکورد در دسترس اشاره می‌کند که در اینجا وهله‌ای از شیء کارمند است.
و یا همانند روشی که برای محاسبه جمع حقوق هر صفحه مشاهده می‌کنید، مستقیما از lambda expressions برای تعریف یک anonymous delegate استفاده کنید.

نحوه اضافه کردن ستون ردیف

ستون ردیف، یک ستون محاسبه شده (calculated field) است:

```
grid.Column(header: "#",
                 style: "text-align-center-col",
                 format: @<text>@(rowIndex++)</text>),
```

نیازی نیست حتما یک grid.Column، به فیلدی در کلاس کارمند اشاره کند. مقدار سفارشی آن را به کمک پارامتر format تعیین خواهیم کرد. هر بار که قرار است یک ردیف رندر شود، یکبار این پارامتر فراخوانی خواهد شد. فرمول محاسبه rowIndex ابتدای صفحه را نیز پیشتر ملاحظه نمودید.

نحوه اضافه کردن ستون سفارشی تصاویر کارمندها

ستون تصویر کارمندها نیز مستقیما در کلاس کارمند تعریف نشده است. بنابراین می‌توان آن‌را با مقدار دهی صحیح پارامتر format ایجاد کرد:

```
grid.Column(header: "Image",
                  style: "text-align-center-col",
                  format: @<text><img alt="@item.Id" src="@Url.Content("~/images/" + @item.Id +
".jpg")" /></text>),
```

در این مثال، تصاویر کارمندها در پوشه images واقع در ریشه سایت، قرار دارند. به همین جهت از متد Url.Content برای مقدار دهی صحیح آن استفاده کردیم. به علاوه در اینجا @item.Id به Id رکورد در حال رندر اشاره می‌کند.

نحوه تبدیل تاریخ‌ها به تاریخ شمسی

در ادامه بازهم به کمک پارامتر format، یک وهله از شیء dynamic اشاره کننده به رکورد در حال رندر را دریافت می‌کنیم. سپس فرصت خواهیم داشت تا بر این اساس، فرمول نویسی کنیم. دست آخر هم رشته مورد نظر نهایی را بازگشت می‌دهیم:

```
grid.Column(columnName: "AddDate", header: "Start",
                  style: "text-align-center-col",
                  format: item =>
                  {
                       int ym = item.AddDate.Year;
                       int mm = item.AddDate.Month;
                       int dm = item.AddDate.Day;
                       var persianCalendar = new PersianCalendar();
                       int ys = persianCalendar.GetYear(new DateTime(ym, mm, dm, new
GregorianCalendar()));
                       int ms = persianCalendar.GetMonth(new DateTime(ym, mm, dm, new
GregorianCalendar()));
                       int ds = persianCalendar.GetDayOfMonth(new DateTime(ym, mm, dm, new
GregorianCalendar()));
                       return ys + "/" + ms.ToString("00") + "/" + ds.ToString("00");
                    }),
```

اضافه کردن ستون سفارشی مالیات

در کلاس کارمند، خاصیت حقوق وجود دارد اما مالیات خیر. با توجه به آن می‌توانیم به کمک پارامتر format، به اطلاعات شیء dynamic در حال رندر دسترسی داشته باشیم. بنابراین به اطلاعات حقوق دسترسی داریم و سپس با کمی فرمول نویسی، مقدار نهایی مورد نظر را بازگشت خواهیم داد. همچنین در اینجا می‌توان نحوه بازگشت مقدار حقوق را به صورت رشته‌ای حاوی جدا کننده‌های سه رقمی نیز مشاهده کرد:

```
grid.Column(columnName: "Salary", header: "Salary",
                format: item =>
                            {
                                salaryPageSum += item.Salary;
                                return string.Format("${0:n0}", item.Salary);
                            },
                            style: "text-align-center-col"),
grid.Column(header: "Tax", canSort: true,
```

```
                                format: item =>
                                {
                                    var tax = item.Salary * 0.2;
                                    taxPageSum += tax;
                                    return string.Format("${0:n0}", tax);
                                }),
```

اضافه کردن گردیدهای تو در تو

متد Grid.GetHtml، یک رشته را بر می‌گرداند. بنابراین در هر چند سطح که نیاز باشد می‌توان یک گرید را بر اساس اطلاعات دردسترس رندر کرد و سپس بازگشت داد:

```
grid.Column(header: "Projects", columnName: "Projects",
                  style: "text-align-center-col",
                  format: item =>
                  {
                                var subGrid = new WebGrid(
                                                        source: item.Projects,
                                                        canPage: false,
                                                        canSort: false
                                                        );
                                return subGrid.GetHtml(
                                                        htmlAttributes: new { id = "MySubGrid" },
                                                        tableStyle: "webgrid",
                                                        headerStyle: "webgrid-header",
                                                        footerStyle: "webgrid-footer",
                                                        alternatingRowStyle: "webgrid-alternating-row",
                                                        selectedRowStyle: "webgrid-selected-row",
                                                        rowStyle: "webgrid-row-style"
                                    );
                       }),
```

در اینجا کار اصلی از طریق پارامتر format شروع می‌شود. سپس به کمک item.Projects به لیست پروژه‌های هر کارمند دسترسی خواهیم داشت. بر این اساس یک گرید جدید را تولید کرد و سپس رشته معادل با آن را به کمک متد subGrid.GetHtml دریافت و بازگشت می‌دهیم. این رشته در سلول جاری درج خواهد شد. به نوعی یک گزارش master detail یا sub report را تولید کرده‌ایم.

اضافه کردن دکمه‌های ویرایش، حذف و انتخاب

هر سه دکمه ویرایش، حذف و انتخاب در ستون‌هایی سفارشی قرار خواهند گرفت. بنابراین مقدار دهی header و format متد grid.Column کفایت می‌کند:

```
grid.Column(header: "",
                  style: "text-align-center-col",
                   format: item => @Html.ActionLink(linkText: "Edit", actionName: "Edit",
                                                                controllerName: "Home", routeValues: new
{ id = item.Id },
                                                                htmlAttributes: null)),
grid.Column(header: "",
                  format: @<form action="/Home/Delete/@item.Id" method="post"><input type="submit"
                                        onclick="return confirm('Do you want to delete this
record?');"
                                        value="Delete"/></form>),
grid.Column(header: "", format: item => item.GetSelectLink("Select"))
```

نکته جدیدی که در اینجا وجود دارد متد item.GetSelectLink می‌باشد. این متد جزو متدهای توکار گرید است و کار آن بازگشت دادن شیء grid.SelectedRow می‌باشد. این شیء پویا، حاوی اطلاعات رکورد انتخاب شده است. برای مثال اگر نیاز باشد این

اطلاعات به صفحه‌ای ارسال شود، می‌توان از روش زیر استفاده کرد:

```
@if (@grid.HasSelection)
    {
        @RenderPage("~/views/path/_partial_view.cshtml", new { Employee = grid.SelectedRow })
    }
```

نمایش برچسب‌های صفحه x از n و رکوردهای x تا y از z

در یک گزارش خوب باید مشخص باشد که صفحه جاری، کدامین صفحه از چه تعداد صفحه کلی است. یا رکوردهای صفحه جاری چه بازه‌ای از تعداد رکوردهای کلی را تشکیل می‌دهند. برای این منظور چند متد کمکی به نام‌های WebGridPageFirstItem و WebGridPageLastItem تهیه شده‌اند که آن‌ها را در ابتدای View ارائه شده، مشاهده نمودید:

```
<strong>Page:</strong> @(grid.PageIndex + 1) / @grid.PageCount,
    <strong>Records:</strong> @WebGridPageFirstItem(@grid) - @WebGridPageLastItem(@grid) of
@grid.TotalRowCount
```

نمایش جمع ستون‌های حقوق و مالیات در هر صفحه

گرید توکار همراه با ASP.NET MVC در این مورد راه حلی را ارائه نمی‌دهد. بنابراین باید اندکی دست به ابتکار زد. مثلا:

```
@section script{
<script type="text/javascript">
    $(function () {
        $('#MyGrid tbody:first').append(
        '<tr class="total-row"><td></td>\
         <td></td><td></td><td></td>\
         <td><strong>Total:</strong></td>\
         <td>@string.Format("${0:n0}", @salaryPageSum)</td>\
         <td>@string.Format("${0:n0}", @taxPageSum)</td>\
         <td></td><td></td><td></td></tr>');
    });
</script>
}
```

در این مثال به کمک jQuery با توجه به اینکه id گرید ما MyGrid است، یک ردیف سفارشی که همان جمع محاسبه شده است، به tbody جدول نهایی تولیدی اضافه می‌شود. از tbody:first هم در اینجا استفاده شده است تا ردیف اضافه شده به گریدهای تو در تو اعمال نشود.

سپس فایل Views\Shared_Layout.cshtml را گشوده و از section تعریف شده، برای مقدار دهی master page سایت، استفاده نمائید:

```
<head>
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
    <script src="@Url.Content("~/Scripts/jquery-1.5.1.min.js")" type="text/javascript"></script>
    @RenderSection("script", required: false)
</head>
```

نظرات خوانندگان

نویسنده: مجید شمخانی
تاریخ: ۱۳۹۱/۰۲/۰۳ ۱۹:۱۸:۴۵

آفرین بر همت عالی‌تان.

نویسنده: ilia
تاریخ: ۱۳۹۱/۰۲/۰۳ ۱۹:۳۷:۵۵

تشکر فراوان آقای نصیری . خسته نباشید. خواندن مقالات شما لذت بخش و خیلی کاربردی و مفیده .

نویسنده: Mohsen Bahrzadeh
تاریخ: ۱۳۹۱/۰۲/۰۳ ۲۲:۲۲:۵۵

خیلی جالب هست استاد منتظر ادامش هستیم.

نویسنده: Naser Tahery
تاریخ: ۱۳۹۱/۰۲/۰۳ ۲۳:۲۵:۰۰

واقعا نشد به لایک کردن تشکر دوستان اکتفا کرد و از نوشتن کامنت اضافی جلوگیری. هر قدر تشکر شود کافی نیست.آرزو دارم به تمام آرزوهاتون برسید.

نویسنده: Mojtaba
تاریخ: ۱۳۹۱/۰۲/۰۴ ۰۰:۲۲:۵۱

واقعا عالی کاربردی عالمانه بسیار فنی و .....
تقریبا از توان علمی اکثر این وریها دور
در حسرت دیدن چنین مهندسانی با این روحیات و این سطح از توانمندی
بسیار بسیار سپاسگزارم جدا لایک خالی فایده نداشت!

نویسنده: Mohammadi4net
تاریخ: ۱۳۹۱/۰۲/۰۴ ۰۹:۴۰:۴۲

نمی دونم در مورد این همه لطفی که به هموطنات داری چی باید بگم ، امیدوارم همیشه در مسیر پیشرفت و سربلندی قدم بردارید.

نویسنده: Na3er Faraji
تاریخ: ۱۳۹۱/۰۲/۰۴ ۲۳:۰۲:۵۴

ممنون.واقعا روم نشد تشکر نکنم.

نویسنده: mojtaba kaviani
تاریخ: ۱۳۹۱/۰۲/۰۵ ۰۰:۵۶:۱۵

ممنون از اینکه همیشه جدیدترین تکنولوژی ها رایگان آموزش میدی منم چند تا سایت با mvc ساختم واقعا عالیه از جمله http://efish.farsedu.ir که بیش از 120000 کاربر داره ولی افت سرعتی وجود نداره

نویسنده: Atoma
تاریخ: ۱۳۹۱/۰۲/۰۵ ۱۴:۱۶:۴۰

ضمن تشکر فراوان از این همت عالی و وقتی که میذارید جناب نصیری!

بنده یک سوال داشتم:
اگر شما در محل کارتون، قرار بود یک پروژه بزرگ تحت وب با دات نت را راه اندازی کنید، آیا انتخابتون MVC بود یا اینکه فعلا از WebForm ها استفاده می کردید تا MVC بالغ تر شود و بعدا سراغش می رفتید؟

بازم ممنونم ازتون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۰۵ ۱۶:۴۶:۴۷

برای توضیحات بیشتر لطفا مراجعه کنید به قسمت اول این سری که در مورد «چرا ASP.NET MVC» بحث شد.

نویسنده: امیرحسین
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۱۵

سلام
امکانی هست که بتونیم کاری کنیم که این پیجینگ‌ها هم به صورت ایجکسی نمایش داده بشه؟
یعنی برای تغییر پیج‌های گرید صفحه پست بک نشه؟

نویسنده: امیرحسین
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۲۰

ببخشید
امکانش هست در مورد سرت کردن کالمن‌ها توضیح بدید؟
چطوری میشه که وقتی روی هدرها کلیک می‌کنیم ، سرت فیلد عوض می‌شه؟ بدون اینکه سمت سرور اتفاقی بیافته؟
ممنونم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۲۵

- این مثال هم paging خودکار داره و هم sorting خودکار. به علاوه پارامترهای شماره صفحه و فیلد سورت شونده هم در اکشن متدها قابل دریافت است که ذکر شده.
- اگر می‌خواهید با ajax کار کنید باید ajaxUpdateContainerId آن‌را مقدار دهی کرده و سپس از Ajax.ActionLink استفاده کنید. یک مثال در اینجا: (^)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۲۶

- دیتاسورس این مثال از new **Random** استفاده کرده.
- در متد Index موجود در HomeController یک breakpoint قرار بدید تا بشود پارامترهای دریافتی را بهتر آنالیز کرد.

نویسنده: داریوش تصدیقی
تاریخ: ۱۳۹۱/۰۶/۱۲ ۱۱:۴۹

با سلام و احترام خدمت دوست و همکار گرامی جناب آقای وحید نصیری عزیز
از مطلب خوب شما بسیار استفاده کردم و بسیار مفید بود...
با تشکر و سپاس از شما
ارادتمند شما
داریوش تصدیقی

نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۱/۰۷/۱۶ ۱۰:۴۶

از لطفتان بسیار سپاس گذارم.
خیلی خیلی ممنون

نویسنده: رضا
تاریخ: ۱۳۹۱/۰۷/۲۳ ۱۳:۱۱

رکورد select شده با این دستور قابل ارجاع و بازیابی نیست و موقع اجرا، خطا میدهد:

```
@RenderPage("~/views/path/_partial_view.cshtml", new { Employee = grid.SelectedRow })
```

چگونه میشود رکورد انتخابی را به‌جا selectشدن به‌صورت لینکی درآورد که به صفحه جزئیات مشخصات کارمند برود. یعنی در قسمت routeValues باید چی‌رو پاس بدیم؟
ممنونم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۳ ۱۴:۴۶

الف) نحوه استفاده از grid.SelectedRow پس از اصلاح مسیر نمادین views/path و ساخت فایلی به نام _partial_view.cshtml در مسیر views/home :

```
@if (@grid.HasSelection)
{
   @RenderPage("~/views/home/_partial_view.cshtml", new { Employee = grid.SelectedRow })
}
```

محتویات فایل _partial_view.cshtml:

```
<br />
<strong>LastName: </strong> @Page.Employee.LastName
```

ب) به همان روشی که در مورد لینک edit در سورس بکار رفته عمل کنید.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۲/۰۸ ۱۶:۳۱

با سلام.

آیا امکان دارد در هنگام page بندی هر بار تنها رکوردهای موردنیاز را از منبع داده بازیابی کرد مثلا 20 تا سطر و در وب گرید نمایش دهیم؟ در این صورت وب گرید تعداد کل سطرها برای page بندی را از کجا میگیرد؟

با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۸ ۱۶:۴۱

- متد index یک چنین امضایی دارد

```
public ActionResult Index(string sort, string sortdir, int? page = 1)
```

زمانیکه این اطلاعات را دارید، برای مثال از LINQ استفاده کرده و با استفاده از متدهای Skip ، Take و OrderBy کار بازیابی قطعه‌ای از اطلاعات مورد نظر را انجام دهید.
- تعداد کل سطرها را هم کوئری بگیرید جداگانه و کش کنید. روش برای ارسال آن به یک View همانند کلیه روش‌های قابل استفاده در MVC است.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۲/۰۸ ۲۳:۲۰

متشکرم.

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۰۸/۱۹ ۱۲:۳۷

هنگام استفاده، من این پیغام رو دریافت کردم

```
The type or namespace name 'WebGrid' could not be found.
```

با اضافه کردن رفرنس زیر به فایل Web.config مشکل حل شد.

```
<compilation debug="true" targetFramework="4.0">
      <assemblies>
        <add assembly="System.Web.Helpers, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" />
      </assemblies>
    </compilation>
```

آیا هر بار باید این رفرنس رو خودمون به صورت دستی اضافه کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۹ ۱۲:۴۵

از چه نگارشی از MVC استفاده می‌کنید؟ System.Web.Helpers, Version= **1.0.0.0** مربوط به MVC 3 است. یک سری نکات هست که باید برای ارتقاء به MVC4 مدنظر داشت: « نحوه ارتقاء برنامه‌های موجود MVC3 به MVC4 »

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۰۸/۱۹ ۱۶:۲۰

نگارش 4.
این System.Web.Helpers, Version= **1.0.0.0** رو هم که به System.Web.Helpers, Version= **2.0.0.0** در لوکال تغییر میدم مشکلی وجود نداره، اما روی هاست اگر اضافه نکنم همون خطای قبلی رو میده و اگر هم اضافه کنم خطای زیر رو میده .

```
Could not load file or assembly 'System.Web.Helpers, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35' or one of its dependencies
```

فایل‌های پوشه bin لوکال رو هم به bin سرور منتقل کردم اما باز هم نشد!
لینکی که معرفی کردی رو هم بررسی کردم. mvc4 بود .

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۰۸/۱۹ ۱۶:۵۷

مشکل حل شد.
دستور Install-Package Microsoft.AspNet.WebPages رو در کنسول زدم و سپس System.Web.Helpers.dll رو از پوشه bin لوکال به پوشه bin سرور بردم.
علت رو اما نفهمیدم! چون این فایل از اول هم توی bin لوکال نبود و بعد از اجرای این دستور اضافه شد، پس چطور قبلش صفحه بدرستی نمایش داده میشد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۹ ۱۷:۱۶

محل بررسی وجود اسمبلی‌ها از GAC و سایر مسیرهای سیستمی شروع می‌شود. بنابراین اگر پیشتر اسمبلی مدنظر در GAC نصب شده باشد، کار به بررسی سایر مسیرها نخواهد رسید.

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۱۰/۰۷ ۱۰:۰

چطور میشه برای قسمت هدر گرید هم یک نوع فرمت ساخت؟
مثلا می‌خواهیم در هدر یک Checkbox قرار بدیم.
با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۷ ۱۰:۳۳

یک مثال در این مورد WebGrid Helper with Check All Checkboxes

نویسنده: mostafa
تاریخ: ۱۳۹۲/۱۰/۰۷ ۱۰:۴۵

سلام؛ تو view که داریم میخوایم چند تا کنترل مثه گرید ویو و repeater داشته باشیم، واسه این کار چه جوری باید کنترلررو بنویسیم؟ میشه چند متد به یک view بفرستیم؟ مشکلم سر فرستادن مقدار از کنترلر به ویوئه. چون کنترلر یه خروجی بیشتر نداره.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۷ ۱۱:۳۲

می‌تونید از ViewModel استفاده کنید.

نویسنده: طاهری
تاریخ: ۱۳۹۲/۱۱/۰۹ ۱۷:۱۰

سلام
اگه بخوام بازه گزارشگیری رو کاربر وارد کند و بر اساس اون اطلاعات تو webgrid نمایش داده شود. یه مشکلی وجود دارد:
مشکل اینجاست که وقتی برای webgrid از

```
@model IEnumerable<myproject.Models.mymodel1>
```

استفاده می‌کنم کد زیر دیگه کار نمیکنه

```
@using (Html.BeginForm(actionName: "myaction", controllerName: "mycontroller"))
        {
                @Html.EditorForModel()
            <input type="submit" value="نمایش" />
        }
```

یعنی هیچ textbox ای برای گرفتن بازه تو این صفحه نشون داده نمیشه چون دیگه مدل رو نمیشناسه.
برای حل این مشکل چیکار باید کرد؟ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۹ ۱۷:۳۸

در یک View اگر قسمت‌های مختلف صفحه نیاز به مدل‌های متفاوتی دارند باید (بهتر است) از ViewModel استفاده کنید. برای نمونه اگر ViewModel شما به صورت زیر تعریف شده‌است و Model1 و Model2 هم هر کدام یک کلاس مجزا هستند:

```
public class MyViewModel
{
    public Model1 Model1 { set; get; }
    public Model2 Model2 { set; get; }
}
```

با این اکشن متد:

```
public ActionResult Index()
{
    var model = new MyViewModel();
    return View(model);
}
```

در یک View به نحو ذیل قابل استفاده خواهد بود:

```
@model Models.MyViewModel

@Html.EditorFor(model=>model.Model1)
<br />
@Html.EditorFor(model=>model.Model2)
```

نویسنده: محمد جواد تواضعی
تاریخ: ۱۳۹۲/۱۱/۱۰ ۰:۱۸

با سلام؛ آیا امکان این است که ویرایش اطلاعات به صورت Inline درون خود جدول انجام شود ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۰ ۱:۶

بله. Inline Editing With The WebGrid

نویسنده: طاهری
تاریخ: ۱۳۹۲/۱۱/۱۲ ۱۰:۵۱

من یه viewmodel به صورت زیر درست کردم:

```
public class viewmodel1
    {
        public model1 M1 { get; set; }
        public IEnumerable<model1> IEM1 { get; set; }
    }
```

این viewmodel رو تو view استفاده می‌کنم. از M1 برای گرفتن بازه گزارشگیری از کاربر و از IEM1 برای وب گرید، ولی به محض اجرای برنامه، اون قسمت که دارم وب گرید رو با new، IEM1 میکنم خطای null میده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۲ ۱۱:۲۷

مثالی که نوشتم برای استفاده در Html.EditorFor بود. خواص این model قبل از return View برای استفاده در یک گزارش باید مقدار دهی شوند و گرنه مقدار پیش فرض خود یا همان نال را خواهند داشت. همچنین بعد از این تغییر، جایی که new WebGrid دارید، پارامتر source آن دیگر Model نیست و اینبار Model.IEM1 است.

نویسنده: ع طاهری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۲:۲۸

سلام.از راهنمایی که کردین ممنونم.مشکلم با viewmodel حل شد.
-همانطور که شما گفتید model در این حالت باید قبل از return view مقداردهی شود. حالا مسئله اینجاست که کاربر بازه‌های گزارشگیری رو وارد کرد که مثلا از تاریخ Aتا تاریخB گزارش میخواد.دو مقدار BوA که از کاربر گرفتم رو چطور به کنترلر پاس بدم که ازشون تو کوئری استفاده کنم؟ و نتیجه select رو برای همان ویو بفرستم تا در webgrid نمایش دهد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۲:۳۰

« آشنایی با روش‌های مختلف ارسال اطلاعات یک درخواست به کنترلر »

نویسنده: عثمان رحیمی
تاریخ: ۱۳۹۴/۰۷/۲۰ ۲۳:۴۴

گرید خوب و سبکی هستش البته با کمی تغییر در نمایش آن .دو سوال داشتم
آیا امکان تنظیم تعداد کل رکوردها به صورت ViewBag هست که بخواهیم متد واکشی رو به صورت ده رکورد (skip,take)بگیریم ؟ دقیقا کجا باید تعداد کل رکوردها ست شود ؟
-2 آیا میتوان به صفحه بندی دسترسی داشت ؟ به طور پیش فرض از تگ a استفاده میشه ، اگر بخواهیم بجای آن تگ li تولید شود چطور ؟
تشکر