Paging-ListView 使用说明

➢概述:

仅需经过简单的配置即可为你的 asp.net mvc 项目实现带有分页功能的通用列表视图!

➢制作背景:

在做项目的过程中发现后台中使用了大量的列表视图，比如

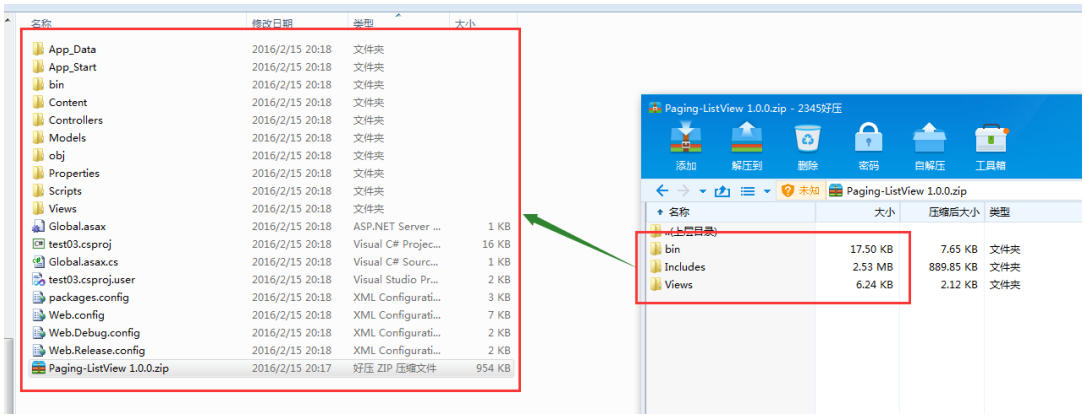
针对每个页面都需要写一个列表视图，一个项目做完后发现控制器和视图中存在大量重复代码，于是试图简化和规范化列表视图，经过一星期的归纳总结，最终实现了设想。不仅实现了通用列表视图，还在列表视图中实现了分页功能和对操作的配置，仅需经过简单的配置即可实现下图所示的效果.

✧ Tip:页面 css 采用的是目前最流行的前端框架 bootstrap3，如果不喜欢页面样式可修改/Views/Shared/ 目录下的 PagingListView.cshtml 文件

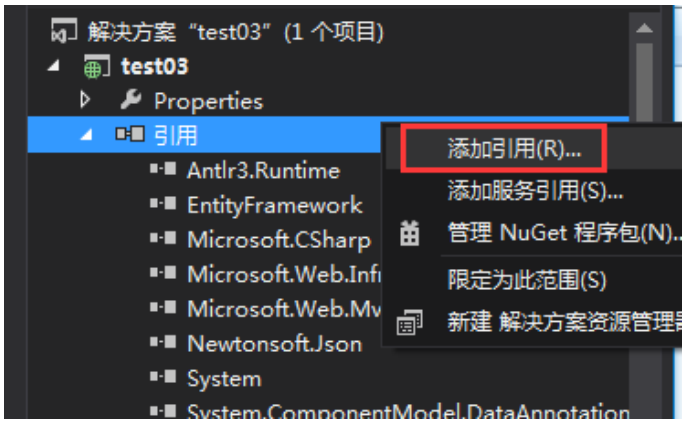
是不是心动了，赶紧跟我一起来开始配置吧~

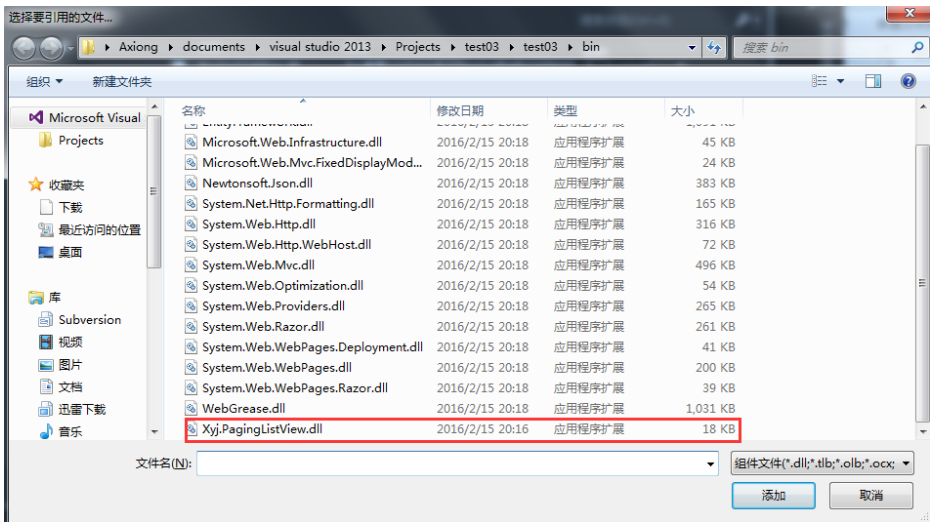
➢准备工作

1. 到 http://52xyj.cn/Paging-ListView 下载最新版本压缩包，将压缩包解压到项目所在目录（若提示覆盖，选择是）



2. 引用 bin 目录下的 Xyj.PagingListView.dll 文件





➢配置模型【Model】

✧ 为模型引入命名空间

```
using xyj.Plugs;
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
using xyj.Plugs;//必须引用
```

✧ 给模型添加注释标记

```
[ModelNoteAttrabute("注释内容")]
```

```
public partial class T_NoticeType
{
    [Key]
    [ModelNoteAttrabute("公告类型编号")]
    1 个引用
    public int NoticeTypeID { get; set; }
    [ModelNoteAttrabute("公告类型名称")]
    1 个引用
    public string NoticeTypeName { get; set; }
    [ModelNoteAttrabute("公告类型备注")]
    1 个引用
    public string NOTE { get; set; }
}
```

➢配置控制器【Controller】

✧ 为控制器引入命名空间

```
using xyj.Plugs;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using test01.Models;
using xyj.Plugs;//必须引入命名空间
```

✧ 让控制器实现接口

```
public class HomeController: Controller,ICutPage,IModelNote
```

```
0 个引用
public class HomeController : Controller, ICutPage, IModelNote//必须实现ICutPage,IModelNote接口
{
    //
    // GET: /Home/
    testDbContext tc = new testDbContext();
    0 个引用
    public ActionResult Index(int id=1)
    {
        if (id == 1)
            return RedirectToAction("NoticeTypeShow");
        if (id == 2)
            return RedirectToAction("NoticeShow");

        return null;
    }
```

✧ 为动作【Action】添加参数

```
(int perCount = 8, int pageIndex = 1)
```

```
public ActionResult NoticeTypeShow(int perCount = 8, int pageIndex = 1)
```

✧ 调用不带连接操作的函数

```
this.CutPage<
                /*数据源模型*/数据源模型>(
                /*数据源模型*/this.GetModelNote<数据源模型>()
                , this, perCount, pageIndex
                ,/*数据源*/数据源.ToList(),
                /*要显示的列索引*/new int[] { 0, 1, 2 },
                /*页面标题*/"页面标题",
                /*列表视图操作栏*/new Dictionary<string, string>() { { "编辑", "Edit" }, { "删除", "
Delete" } }
                );
```

```
0 个引用
public ActionResult NoticeTypeShow(int perCount = 8, int pageIndex = 1)
{
    this.CutPage<
        /*数据源模型*/T_NoticeType>(
        /*数据源模型*/this.GetModelNote<T_NoticeType>()
        , this, perCount, pageIndex
        ,/*数据源*/tc.T_NoticeType.ToList(),
        /*要显示的列索引*/new int[] { 0, 1, 2 },
        /*页面标题*/"公告列表",
        /*列表视图操作栏*/new Dictionary<string, string>() { { "编辑", "NoticeTypeEdit" }, { "删除", " NoticeTypeDelete" } }
        );
    return View("Index");
}
```

✧ 调用含有多表查询的函数

```
this.CutPage<
                /*数据源模型*/数据源模型
                ,/*视图数据源模型*/视图数据源模型>(
                /*视图数据源模型*/this.GetModelNote<视图数据源模型>()
                , this, perCount, pageIndex
                ,/*数据源*/数据源.ToList()
                ,/*视图数据源*/视图数据源.ToList()
                ,/*要显示的视图列索引*/new int[] { 0, 1, 2, 3, 4, 5 }
                ,/*页面标题*/"页面标题"
                ,/*列表视图操作栏*/ new Dictionary<string, string>() { { "编辑", "Edit" }, { "删除",
"Delete" } }
                );
```

```
public ActionResult NoticeShow(int perCount = 8, int pageIndex = 1)
{
    var data = tc.T_Notice.ToList();
    this.CutPage<
        /*数据源模型*/T_Notice
        ,/*视图数据源模型*/T_Notice_Linked>(
        /*视图数据源模型*/this.GetModelNote<T_Notice_Linked>()
        , this, perCount, pageIndex
        ,/*数据源*/data
        ,/*视图数据源*/data.Join(tc.T_NoticeType,
              a => a.NoticeTypeID,
              b => b.NoticeTypeID,
              (a, b) => new T_Notice_Linked { NoticeID = a.NoticeID.ToString(), NoticeTypeID = a.NoticeID.ToString(),
                                              Contents = a.Contents, NOTE = a.NOTE, NoticeTypeName = b.NoticeTypeName, NoticeTypeNOTE = b.NOTE,
                                              SubmitTime = a.SubmitTime.ToString(), Title = a.Title, UserID = a.UserID }).ToList()
        ,/*要显示的视图列索引*/new int[] { 0, 1, 2, 3, 4, 5 }
        ,/*页面标题*/"公告列表"
        ,/*列表视图操作栏*/ new Dictionary<string, string>() { { "编辑", "NoticeEdit" }, { "删除", "NoticeDelete" } }
        );

    return View("Index");
}
```

➢配置视图【View】

✧ 在视图中引入列表视图

```
@Html.Partial("_PagingListView")
```

```
1  <!--引入列表视图-->
2  @Html.Partial("_PagingListView")
```

➢联系作者

最新版本动态请关注网站：http://52xyj.cn/Paging-ListView

有任何意见或更好的建议请发送邮件至 xiongyingjie@sina.cn

捐助作者：xiongyingjie@sina.cn (支付宝)

Design by 小熊

2016/2/15