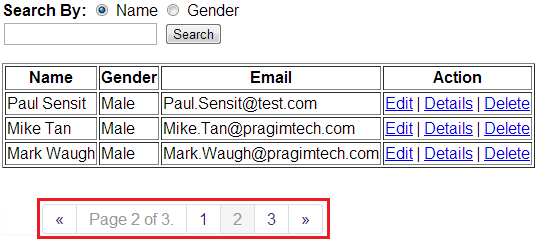
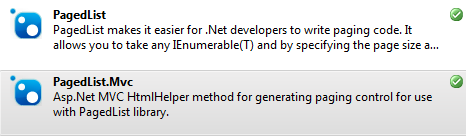
In this video we will discuss, **implementing pagination in an asp.net mvc application**. Please watch [Part 62](http://csharp-video-tutorials.blogspot.com/2013/07/part-62-implementing-search.html), before proceeding.

By the end of this video, the index page should support both **search functionality** and **pagination** as shown below.   
**Step 1:** Install **PagedList.Mvc** using NuGet package manager. PagedList.Mvc is dependent on PagedList. Installing PagedList.Mvc will automatically install PagedList package as well.   
   
  
**Step 2:** Include the following using statements in HomeController.cs file  
using PagedList.Mvc;  
using PagedList;  
  
Modify the **Index**() action method as shown below. Notice that we are passing **page**parameter to this function. This parameter is used for specifying the page number. This parameter can be null, and that's the reason we have chosen a **nullable integer**. We convert the list, to a paged list, using ToPagedList(). Also, notice that, we are using null-coalescing operator. If  the "page" parameter is null, then 1 is passed as the page number, else, the value contained in the "page" parameter is used as the page number.  
public ActionResult Index(string searchBy, string search, int? page)  
{  
    if (searchBy == "Gender")  
    {  
        return View(db.Employees.Where(x => x.Gender == search || search == null).ToList().ToPagedList(page ?? 1, 3));  
    }  
    else  
    {  
        return View(db.Employees.Where(x => x.Name.StartsWith(search) || search == null).ToList().ToPagedList(page ?? 1, 3));  
    }  
}  
  
**Step 3:** Make the following modifications to Index.cshtml view  
**a)** Include the following 2 using statements on the view.  
@using PagedList.Mvc;  
@using PagedList;  
  
**b)** The model for the view should be IPagedList<Employee>.  
@model IPagedList<MVCDemo.Models.Employee>  
  
**c)** Since, we have changed the model of the view, from IEnumerable<MVCDemo.Models.Employee> to IPagedList<MVCDemo.Models.Employee>, change the section that displays table headings as shown below.  
<tr>  
    <th>  
        @Html.DisplayNameFor(model => model.First().Name)  
    </th>  
    <th>  
        @Html.DisplayNameFor(model => model.First().Gender)  
    </th>  
    <th>  
        @Html.DisplayNameFor(model => model.First().Email)  
    </th>  
    <th>Action</th>  
</tr>  
  
**d)** **Finally to display page numbers for paging**  
@Html.PagedListPager(Model, page => Url.Action("Index", new { page, searchBy = Request.QueryString["searchBy"], search = Request.QueryString["search"] }))  
  
**e)** **If you want to display the pager, only if there are more than 1 page**

@Html.PagedListPager(Model, page => Url.Action("Index", new { page, searchBy = Request.QueryString["searchBy"], search = Request.QueryString["search"] }), new PagedListRenderOptions() { Display = PagedListDisplayMode.IfNeeded })  
  
**f)** **If you want to display, the current active page and the total number of pages**  
@Html.PagedListPager(Model, page => Url.Action("Index", new { page, searchBy = Request.QueryString["searchBy"], search = Request.QueryString["search"] }), new PagedListRenderOptions() { Display = PagedListDisplayMode.IfNeeded, DisplayPageCountAndCurrentLocation = true })  
  
**g) If you want to display the number of rows displayed, of the total number of rows available.**  
@Html.PagedListPager(Model, page => Url.Action("Index", new { page, searchBy = Request.QueryString["searchBy"], search = Request.QueryString["search"] }), new PagedListRenderOptions() { Display = PagedListDisplayMode.IfNeeded, DisplayItemSliceAndTotal = true })