CODER FOUNDRY

MVC

Adding Paging to a List View

It's a good idea, when displaying lists of objects on a page, to control the number of objects from that list that are displayed at any given time. When shopping online, for example, most shopping pages provide the ability to restrict the number of items matching a given criteria to 10, 20, or 50 items per page. We want to do the same thing with your Blog so as to avoid having too much information displayed on your list page at any given time. To do this, we use a Nuget package called PagedList. Follow the steps below to add this functionality to your Blog.

1) Add the PagedList.Mvc NuGet package.
2) Add two using statements – PagedList and PagedList.Mvc – to the BlogPosts controller.
3) Add a nullable int parameter, page, to the Index action. Add the following code before returning the view:

```
int pageSize = 3; // display three blog posts at a time on this page
int pageNumber = (page ?? 1);
```

4) Instead of returning a list of blog posts objects (i.e. posts.ToList()), return a PagedList (posts.ToPagedList()). Be sure to pass in pageNumber and pageSize to the ToPagedList() method.
5) Open the Index view (Index.cshtml) and add a using statement for PagedList and PagedList.Mvc (i.e. `@using PagedList;`) to the top, above the model reference.
6) Change the model referenced to type IPagedList<T> where T is the type of enumerable returned by the action.
7) Add a link to the stylesheet for PagedList:

```
<link href="~/Content/PagedList.css" rel="stylesheet" type="text/css">
```

8) Add the following to the bottom of the page (wherever you want your paging controls to appear) – this will insert controls for navigating the pages of your list:

```
Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCount
@Html.PagedListPager(Model, page => Url.Action("Index", new { page }))
```

9) Test your index view to make sure your paging works as designed.