**PagingSortingSearch**

В този пример ще покажем как да реализираме търсене , сортиране и в края и страниране.

За примера ще използваме базата от данни RestaurantsDB.

----------------

Създаваме празен проект File->New-> Asp.Net Web Application с името „PagingSortingSearch“ – Empty + MVC

Добавяме Entity Framework Data Model:

Десен бутон на проекта -> Add new item -> Data -> ADO.NET Entity Data Model с името “RestaurantsModel”. Избираме <EF Designer from database> - Next - <New Connection>

Server name: localhost\sqlexpress

Database name: RestaurantsDB

Next -> Entity Framework 6.x -> Next -> +Tables, + Pluralize … , + Include foreign key… -> Finish

От файла DatabaseModel.Context.cs виждаме че контекста е с името RestaurantsDBEntities

и с него ще достъпваме базата данни.

--------------

Добавяме HomeController и Index.cshtml

---------------

В Models добавяме един SearchViewModel клас. Тъй като към view-то ще визуализираме списък с намерените ресторанти трябва във този viewModel клас да имаме списък който да ги поддържа. Затова дефинираме и един помощен клас SearchItem, който ще съдържа информация за един ресторант, а във SearchViewModel ще имаме списък от такива.

public class SearchItem

{

public int RestaurantID { get; set; }

public string RestaurantName { get; set; }

public string RestaurantCategory { get; set; }

public string RestaurantCity { get; set; }

public SearchItem(Restaurant dbRestaurant)

{

RestaurantID = dbRestaurant.ID;

RestaurantName = dbRestaurant.Name;

RestaurantCategory = dbRestaurant.Category.Name;

RestaurantCity = dbRestaurant.City.Name;

}

}

public class SearchViewModel

{

public string LastSortColumn { get; set; }

public string LastSortDirection { get; set; }

public List<SearchItem> SearchItems;

public SearchViewModel()

{

SearchItems = new List<SearchItem>();

}

public SearchViewModel(List<Restaurant> list) : this()

{

list.ForEach(item => SearchItems.Add(new SearchItem(item)));

}

}

===================

Когато имаме само сортиране по дадена колона, action метода Index() в HomeController изглежда така:

public ActionResult Index(string sortColumn, string direction)

{

RestaurantsDBEntities dbContext = new RestaurantsDBEntities ();

IQueryable<Restaurant> records = dbContext.Restaurants.AsQueryable();

string sortColDirection = sortColumn + direction;

switch (sortColDirection)

{

case "Category":

records = records.OrderBy(r => r.Category.Name);

break;

case "CategoryDesc":

records = records.OrderByDescending(r => r.Category.Name);

break;

case "City":

records = records.OrderBy(r => r.City.Name);

break;

case "CityDesc":

records = records.OrderByDescending(r => r.City.Name);

break;

case "NameDesc":

records = records.OrderByDescending(r => r.Name);

break;

default:

records = records.OrderBy(r => r.Name);

break;

}

// convert the IQueryable list to list (i.e. retrieve the data from the database)

List<Restaurant> list = records.ToList();

// create the view model

SearchViewModel viewModel = new SearchViewModel(list);

viewModel.LastSortColumn = sortColumn;

viewModel.LastSortDirection = direction;

return View(viewModel);

}

---------

А съответно view-то:

@using PagingSortingSearch.Models

@model SearchViewModel

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<table class="table" border="1">

<tr>

<th> @Html.ActionLink("Name", "Index",

new { sortColumn = "Name",

direction = Model.LastSortColumn == "Name" &&

Model.LastSortDirection != "Desc" ? "Desc" : ""

})</th>

<th> @Html.ActionLink("Category", "Index",

new { sortColumn = "Category",

direction = Model.LastSortColumn == "Category" &&

Model.LastSortDirection != "Desc" ? "Desc" : ""

})</th>

<th> @Html.ActionLink("City", "Index",

new { sortColumn = "City",

direction = Model.LastSortColumn == "City" &&

Model.LastSortDirection != "Desc" ? "Desc" : ""

})</th>

</tr>

@if (Model.SearchItems.Count() == 0)

{

<tr>

<td colspan="3"> No records found </td>

</tr>

}

else

{

foreach (SearchItem item in Model.SearchItems)

{

<tr>

<td> @Html.DisplayFor(modelItem => item.RestaurantName) </td>

<td> @Html.DisplayFor(modelItem => item.RestaurantCategory) </td>

<td> @Html.DisplayFor(modelItem => item.RestaurantCity) </td>

</tr>

}

}

</table>

--------------

Може да пробваме.

---------------

Нека сега добавим допълнителни филтри за търсене.

public ActionResult Index(string restaurantName, string sortColumn, string direction,

int categoryID = 0)

{

RestaurantsEFCEntities dbContext = .....

List<Category> allCategories = dbContext.Categories.ToList();

ViewBag.AllCategories = new SelectList(allCategories, "ID", "Name", categoryID);

IQueryable<Restaurant> records = .....

// first filter the results

if (string.IsNullOrEmpty(restaurantName) == false)

{

records = records.Where(r => r.Name.Contains(restaurantName));

}

if (categoryID != 0)

{

records = records.Where(r => r.CategoryID == categoryID);

}

// then sort by the specified column

string sortColDirection = ...

switch (sortColDirection) ...

List<Restaurant> list = ...

SearchViewModel viewModel = ...

viewModel.LastSortColumn = ...

viewModel.LastSortDirection = ...

return View(viewModel);

}

------------------

И добавяме във view-то:

.........

<h2>Index</h2>

@using (Html.BeginForm("Index", "Home", FormMethod.Get))

{

<text>Restaurant: </text> @Html.TextBox("restaurantName",

Request.QueryString["restaurantName"])

<br />

<text>Category: </text> @Html.DropDownList("categoryID",

(IEnumerable<SelectListItem>)ViewBag.AllCategories, "")

<input type="submit" value="Search" />

}

<table class="table" border="1">

<tr>

<th> @Html.ActionLink("Name", "Index",

new { sortColumn = "Name",

direction = Model.LastSortColumn == "Name" &&

Model.LastSortDirection != "Desc" ? "Desc" : "",

restaurantName = Request.QueryString["restaurantName"],

categoryID = Request.QueryString["categoryID"]

})</th>

<th> @Html.ActionLink("Category", "Index",

new { sortColumn = "Category",

direction = Model.LastSortColumn == "Category" &&

Model.LastSortDirection != "Desc" ? "Desc" : "",

restaurantName = Request.QueryString["restaurantName"],

categoryID = Request.QueryString["categoryID"]

})</th>

<th> @Html.ActionLink("City", "Index",

new { sortColumn = "City",

direction = Model.LastSortColumn == "City" &&

Model.LastSortDirection != "Desc" ? "Desc" : "",

restaurantName = Request.QueryString["restaurantName"],

categoryID = Request.QueryString["categoryID"]

})</th>

</tr>

@if (Model.SearchItems.Count() == 0)

...

=============================================

Сега ще добавим страниране. Първо отиваме в SearchViewModel и там добавяме една константа за това колко записа да има на една страница (PageSize), няколко допълнителни пропъртита и два допълнителни параметъра в конструктора с параметри.

public class SearchViewModel

{

public const int PageSize = 3;//

...

public int CurrentPageIndex { get; set; }

public int TotalPagesCount { get; set; }

public int TotalItemsCount { get; set; }

public bool HasFirstPage

{

get { return CurrentPageIndex > 1; }

}

public bool HasLastPage

{

get { return CurrentPageIndex < TotalPagesCount; }

}

public bool HasPrevPage

{

get { return CurrentPageIndex > 1; }

}

public bool HasNextPage

{

get { return CurrentPageIndex < TotalPagesCount; }

}

public SearchViewModel(List<Restaurant> list, int pageIndex, int recordsCount)

: this()

{

//list.ForEach(item => SearchItems.Add(new SearchItem(item)));

foreach (Restaurant restaurant in list)

{

SearchItems.Add(new SearchItem(restaurant));

}

TotalItemsCount = recordsCount;

TotalPagesCount = ((recordsCount - 1) / PageSize) + 1;

CurrentPageIndex = pageIndex;

}

}

---------------

Отиваме в HomeController-а и в Index() добавяме още един параметър pageIndex за номера на страницата. Освен това променяме и финалното извилчане на данните.

public ActionResult Index(string restaurantName, string sortColumn, string direction, int pageIndex = 1, int categoryID = 0)

{

...

//List<Restaurant> list = records.ToList();//old

List<Restaurant> list = records

.Skip((pageIndex - 1) \* SearchViewModel.PageSize)

.Take(SearchViewModel.PageSize)

.ToList();

int allRecordsCount = records.Count(); //

// create the view model

// SearchViewModel viewModel = new SearchViewModel(list);//old

SearchViewModel viewModel = new SearchViewModel(list, pageIndex, allRecordsCount);

...

return View(viewModel);

}

----------------

Сега остана да променим и view-то

@using PagingSortingSearch\_.Models

@model SearchViewModel

@{

ViewBag.Title = "Index";

var SetPagerItemVisibility = new Func<bool, string>(isVisible => (isVisible) ? "" : "visibility:hidden");

}

<h2>Index</h2>

@using (Html.BeginForm ...

{ ...

}

<div>

Found @Model.TotalItemsCount restaurants

</div>

<table class="table" border="1">

<tr>

<th class="col-md-4"> @Html.ActionLink(…)</th>

<th class="col-md-4"> @Html.ActionLink(…)</th>

<th class="col-md-4"> @Html.ActionLink(…)</th>

</tr>

…

</table>

<div>

<ul class="pager">

<li style="@SetPagerItemVisibility(Model.HasFirstPage)">

@Html.ActionLink(" << ", "Index",

new { pageIndex = 1, restaurantName = Request.QueryString["restaurantName"],

categoryID = Request.QueryString["categoryID"] })

</li>

<li style="@SetPagerItemVisibility(Model.HasPrevPage)">

@Html.ActionLink(" < ", "Index",

new { pageIndex = Model.CurrentPageIndex - 1,

restaurantName = Request.QueryString["restaurantName"],

categoryID = Request.QueryString["categoryID"] })

</li>

<li>

<span>Page @Model.CurrentPageIndex of @Model.TotalPagesCount</span>

</li>

<li style="@SetPagerItemVisibility(Model.HasNextPage)">

@Html.ActionLink(" > ", "Index",

new { pageIndex = Model.CurrentPageIndex + 1,

restaurantName = Request.QueryString["restaurantName"],

categoryID = Request.QueryString["categoryID"] })

</li>

<li style="@SetPagerItemVisibility(Model.HasLastPage)">

@Html.ActionLink(" >> ", "Index",

new { pageIndex = Model.TotalPagesCount,

restaurantName = Request.QueryString["restaurantName"],

categoryID = Request.QueryString["categoryID"] })

</li>

</ul>

</div>

<style>

.pager li > span {

border-radius: 0 !important;

border: 0 !important;

}

</style>