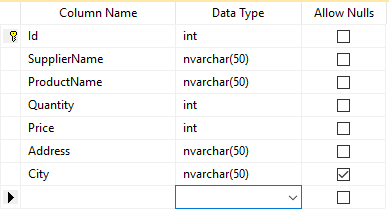
**ASP.NET CORE 2.0 :: How to display and Export PDF data in Grouping using Generic Repository**

**Introduction:** In this article all about Grouping and Export PDF(Download PDF) concept in easy way using Asp.Net CORE 2.0. In this article City wise grouping data display and after download PDF file display **Total record, Total Price, Total quantity of City.**

**Database:**

Create one databse in your Sql Server.After that create table as well as display below.

Table Name : OrderDetails



Step 1: Install packages. Installing below display Packages. PdfRpt.Core package for export data in pdf file. It use to download and export file.

Install package :

JQWidgets

Newtonsoft.Json

PdfRpt.Core

Download Below JS and CSS

CSS:

<https://jqwidgets.com/public/jqwidgets/styles/jqx.base.css>

JS:

<https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js>

<https://www.jqwidgets.com/jquery-widgets-demo/jqwidgets/jqxcore.js>

<https://www.jqwidgets.com/jquery-widgets-demo/jqwidgets/jqxdata.js>

<https://www.jqwidgets.com/jquery-widgets-demo/jqwidgets/jqxbuttons.js>

<https://www.jqwidgets.com/jquery-widgets-demo/jqwidgets/jqxscrollbar.js>

jqxdatatable.js

demos.js

Step 2: Now Create Database and table and set connection string in project. In you project open appsettings.json file and paste your connection string.

Connectionstring: appsettings.json

{

"ConnectionStrings": {

"TestNetCoreEF": "Data Source=111.111.1.111;Initial Catalog=RAJ\_GenericRepository\_Core\_Crud;User ID=raj;Password=\*\*\*\*\*\*\*;"

},

"Logging": {

"IncludeScopes": false,

"LogLevel": {

"Default": "Warning"

}

}

}

Step 3:

Create Models **OrderDetails.cs : Models/ OrderDetails.cs.**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace Core\_GenericRepository\_Crud.Models

{

[Table("OrderDetails")]

public class OrderDetails

{

public int Id { get; set; }

public string SupplierName { get; set; }

public string ProductName { get; set; }

public int Quantity { get; set; }

public int Price { get; set; }

public string Address { get; set; }

public string City { get; set; }

}

}

Step 4: Create Context class: Models/ OrderDBContext.cs.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

namespace Core\_GenericRepository\_Crud.Models

{

public class OrderDBContext : DbContext

{

public OrderDBContext(DbContextOptions<OrderDBContext> options)

: base(options)

{ }

public DbSet<OrderDetails> OrderDetails { get; set; }

}

}

Step 5: Add services of contextclass in Startup.cs in

public void ConfigureServices(IServiceCollection services)

{

// Add framework services.

services.AddDbContext<OrderDBContext>(options => options.UseSqlServer(Configuration.GetConnectionString("TestNetCoreEF")));

//----------for export PDF =>

services.AddNodeServices();

services.AddMvc();

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseBrowserLink();

}

else

{

app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseMvc(routes =>

{ routes.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{id?}");

});

}

Step 6:

Now create Controller ex. HomeController. In this controller first Get

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Core\_GenericRepository\_Crud.Models;

using Core\_GenericRepository\_Crud.Repository;

using Newtonsoft.Json;

using System.IO;

using iTextSharp.text.pdf;

using iTextSharp.text;

using Microsoft.AspNetCore.Hosting;

namespace Core\_GenericRepository\_Crud.Controllers

{

public class HomeController : Controller

{

private OrderDBContext db;

private readonly IHostingEnvironment \_environment;

public HomeController(OrderDBContext context, IHostingEnvironment IHostingEnvironment)

{

db = context;

\_environment = IHostingEnvironment;

}

public IActionResult Index()

{

return View();

}

public JsonResult GetOrderDetails()

{

var sa = new JsonSerializerSettings();

var DbOrdersDetails = from d in db.OrderDetails

select new

{

d.SupplierName,

d.ProductName,

d.Quantity,

d.Price,

d.Address,

d.City

};

return Json(DbOrdersDetails,sa);

}

public FileResult ExportPDF()

{

//stuff

// <th>

var columnsHeader = new List<string>{

"Sr.No",

"Supplier Name",

"Product Name",

"Quantity",

"Price",

"Address",

"City"

};

//heading

string heading = "Report";

var Orderlist = (from s in db.OrderDetails group s by s.City into ab select new { City = ab.Key }).ToList().ToArray();

int a=Orderlist.Count();

//end stuff

var document = new iTextSharp.text.Document();

var outputMS = new MemoryStream();

var writer = PdfWriter.GetInstance(document, outputMS);

var webRoot = \_environment.WebRootPath;

document.Open();

/\* ----- Add logo ------\*/

var file = System.IO.Path.Combine(webRoot, "logo.png");

iTextSharp.text.Image jpg = iTextSharp.text.Image.GetInstance(file);

//Resize image depend upon your need

jpg.ScaleToFit(140f, 120f);

//Give space before image

jpg.SpacingBefore = 10f;

//Give some space after the image

jpg.SpacingAfter = 1f;

jpg.Alignment = Element.ALIGN\_CENTER;

document.Add(jpg);

var font = FontFactory.GetFont(FontFactory.HELVETICA, 12);

var HeaderFont = FontFactory.GetFont(FontFactory.HELVETICA\_BOLD, 12);

document.Add(new Phrase(Environment.NewLine));

var count = columnsHeader.Count;

var table = new PdfPTable(count);

float[] widths = new float[] { 2f, 5f, 6f, 3f, 3f, 6f, 4f};

table.SetWidths(widths);

table.WidthPercentage = 100;

var cell = new PdfPCell(new Phrase(heading));

cell.Colspan = count;

for (int i = 0; i < count; i++)

{

var headerCell = new PdfPCell(new Phrase(columnsHeader[i], HeaderFont));

headerCell.BackgroundColor = BaseColor.Orange;

//headerCell.BackgroundColor = new BaseColor(55, 54, 45);

table.AddCell(headerCell);

}

int gTotal = 0;

int qTotal = 0;

foreach (var ct in Orderlist)

{

var sn = 1;

cell = new PdfPCell(new Phrase("City : " + ct.City.ToString().ToUpper(),HeaderFont));

cell.BackgroundColor = new BaseColor(240,240,240);

cell.Colspan = count;

table.AddCell(cell);

var users = db.OrderDetails.Where(t => t.City == ct.City).ToList();

int b = users.Count();

int price = 0;

int quanty = 0;

foreach (var item in users)

{

table.AddCell(new Phrase(sn.ToString(), font));

table.AddCell(new Phrase(item.SupplierName, font));

table.AddCell(new Phrase(item.ProductName, font));

table.AddCell(new Phrase(item.Quantity.ToString(), font));

table.AddCell(new Phrase(item.Price.ToString(), font));

table.AddCell(new Phrase(item.Address, font));

table.AddCell(new Phrase(item.City, font));

sn++;

price += item.Price;

quanty += item.Quantity;

}

gTotal += price;

qTotal += quanty;

var SrCell = new PdfPCell(new Phrase("Total Record : " + (sn-1)));

SrCell.Colspan = 2;

SrCell.BackgroundColor = new BaseColor(211, 235, 214);

//SrCell.HorizontalAlignment = Element.ALIGN\_RIGHT;

table.AddCell(SrCell);

var QtyCell = new PdfPCell(new Phrase("Total Quantity : " + quanty));

QtyCell.BackgroundColor = new BaseColor(211, 235, 214);

QtyCell.Colspan = 2;

QtyCell.HorizontalAlignment = Element.ALIGN\_RIGHT;

table.AddCell(QtyCell);

var PriceCell = new PdfPCell(new Phrase("Total Price : " + price));

PriceCell.BackgroundColor = new BaseColor(211, 235, 214);

PriceCell.Colspan = 3;

//PriceCell.HorizontalAlignment = Element.ALIGN\_RIGHT;

table.AddCell(PriceCell);

}

var gQtyCell = new PdfPCell(new Phrase("Grand Total Quantity : " + qTotal));

gQtyCell.BackgroundColor = new BaseColor(240, 240, 240);

gQtyCell.Colspan = 3;

gQtyCell.HorizontalAlignment = Element.ALIGN\_RIGHT;

table.AddCell(gQtyCell);

var gTcell = new PdfPCell(new Phrase("Grand Total Price : " + gTotal));

gTcell.BackgroundColor = new BaseColor(240, 240, 240);

gTcell.Colspan = 4;

gTcell.HorizontalAlignment = Element.ALIGN\_RIGHT;

table.AddCell(gTcell);

document.Add(table);

document.Close();

var result = outputMS.ToArray();

return File(result, "application/pdf", "Order Details.pdf");

}

public IActionResult About()

{

ViewData["Message"] = "Your application description page.";

return View();

}

public IActionResult Contact()

{

ViewData["Message"] = "Your contact page.";

return View();

}

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

Step 7:

Now create partial view of Index Views/Home/Index.cshtml

Index.cshtml

@model Core\_GenericRepository\_Crud.Models.OrderDetails

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Group</title>

<**environment** **include**="Development">

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />

<link rel="stylesheet" href="~/css/site.css" />

</**environment**>

<**environment** **exclude**="Development">

<**link** rel="stylesheet" **href**="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"

**asp-fallback-href**="~/lib/bootstrap/dist/css/bootstrap.min.css"

**asp-fallback-test-class**="sr-only" **asp-fallback-test-property**="position" **asp-fallback-test-value**="absolute" />

<**link** rel="stylesheet" **href**="~/css/site.min.css" **asp-append-version**="true" />

</**environment**>

<!--CSS -->

<link href="https://jqwidgets.com/public/jqwidgets/styles/jqx.base.css" rel="stylesheet" />

<!--JS -->

<script src="~/js/jquery-1.10.2.min.js"></script>

<script src="~/js/jqxcore.js"></script>

<script src="~/js/jqxdata.js"></script>

<script src="~/js/jqxbuttons.js"></script>

<script src="~/js/jqxscrollbar.js"></script>

<script src="~/js/jqxdatatable.js"></script>

<script src="~/js/demos.js"></script>

<script type="text/javascript">

$(document).ready(function () {

// prepare the data

var source =

{

dataType: "json",

dataFields: [

{ name: 'SupplierName', type: 'string' },

{ name: 'ProductName', type: 'string' },

{ name: 'Quantity', type: 'number' },

{ name: 'Price', type: 'number' },

{ name: 'Address', type: 'string' },

{ name: 'City', type: 'string' }

],

url: 'Home/GetOrderDetails'

//url.Action("GetOrderDetails", "Home")

};

var dataAdapter = new $.jqx.dataAdapter(source);

// create jqxDataTable.

$("#dataTable").jqxDataTable(

{

source: dataAdapter,

pageable: true,

altRows: true,

sortable: true,

groups: ['City'],

width: 1250,

groupsRenderer: function (value, rowData, level) {

return "City Name: " + value;

},

columns: [

{ text: 'Supplier Name', cellsAlign: 'left', align: 'left', dataField: 'SupplierName', width: 280 },

{ text: 'Product Name', cellsAlign: 'left', align: 'left', dataField: 'ProductName', width: 250 },

{ text: 'Quantity', dataField: 'Quantity', cellsformat: 'd', cellsAlign: 'right', align: 'right', width: 80 },

{ text: 'Price', dataField: 'Price', cellsformat: 'c2', align: 'right', cellsAlign: 'right', width: 90 },

{ text: 'Address', cellsAlign: 'center', align: 'center', dataField: 'Address', width: 250 },

{ text: 'City', cellsAlign: 'center', align: 'center', dataField: 'City' }

]

});

});

</script>

</head>

<body>

<h2> Grouping List</h2>

<div id="dataTable">

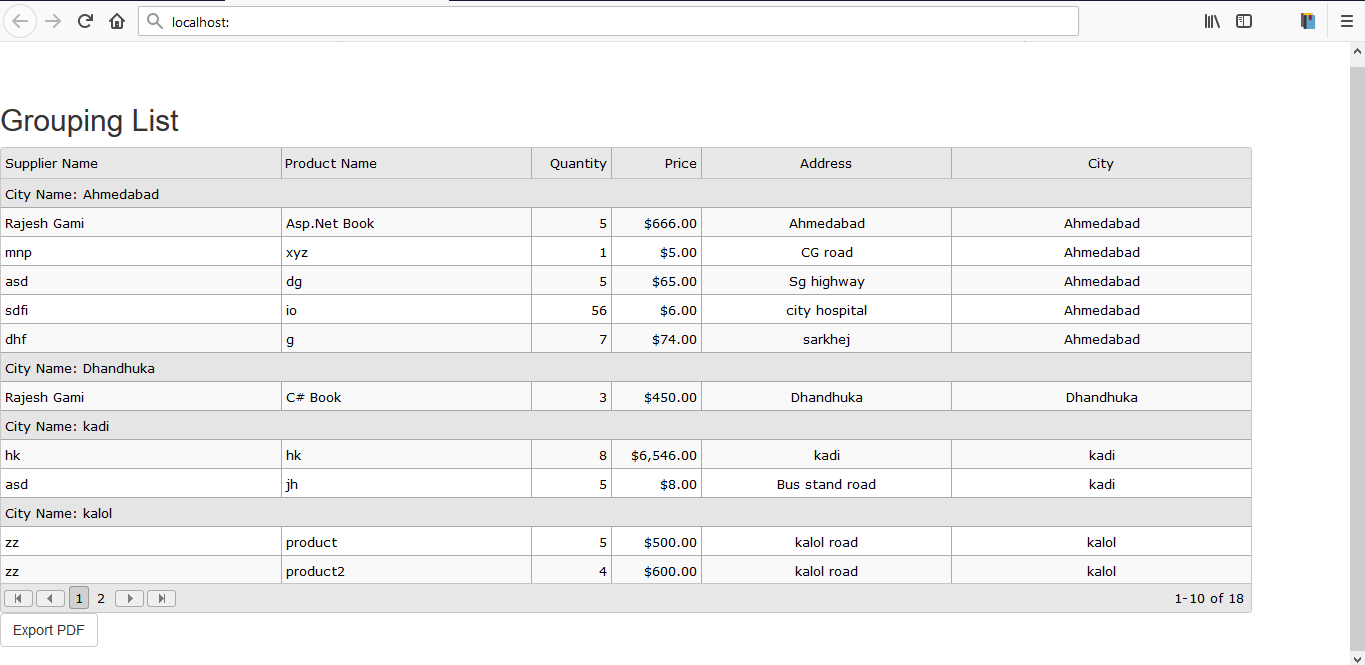
</div>

<**a** class="btn btn-default" **asp-action**="ExportPDF">Export PDF</**a**>

</body>

</html>

**ScreenShots**:::::::: Index view : Display data with city Group.



Export PDF button

Click on **Export PDF** then save pdf and Open : show below image of PDF view