You are ASp.NET Core and Telerik ASP.NET Core UI expert. I need your help to use Telerik report in my application. Application is ASP.NET Core MVC v8.0. Application doesn't use any services. Application use dapper to make direct SQL calls from controller. Pls provide me controller, model and other files (may be Telerik template to render report). Provide all fullSQL files and location (folder path). For controller use namespace BCES.Controllers.Reports/ For Model use namespace BCES.Models.Reports. Assume dbconnection and dbcontext is available. I prefer servide calls. Add comments and exception handling in the code Use following SQL code to generate report SQL : SELECT TOP (1000) [VehicleListId] ,[VehSeriesCode] ,[NumOfVehicles] ,[ProjDesc] ,[DateEntered] ,[EnteredBy] ,[ModifiedLastBy] ,[ModifiedLastDate] ,[Make] ,[Model] ,[Year] ,[Engine] ,[Transmission] ,[Differential] ,[SopNumber] FROM [SCES].[SCES].[VehicleList]

**ChatGPT said:**



To integrate a Telerik report into your ASP.NET Core MVC application using Dapper for database operations, here’s a step-by-step guide, including the necessary files and folder structure. This will help you generate a report from the provided SQL query.

**1. Folder and Namespace Setup**

We will use the following structure:

* **Controllers**: BCES.Controllers.Reports
* **Models**: BCES.Models.Reports
* **Views**: Views/Reports
* **Reports**: Reports/VehicleReport.trdp (Telerik Report Definition file)

**2. SQL Query Integration**

We’ll use your provided SQL query to fetch data from the VehicleList table.

**Step 1: Install Required Packages**

Ensure your project includes the necessary Telerik Reporting and Dapper NuGet packages:

bash

Copy code

dotnet add package Telerik.Reporting

dotnet add package Dapper

**Step 2: Create the Model**

File: Models/Reports/VehicleModel.cs Namespace: BCES.Models.Reports

csharp

Copy code

using System;

namespace BCES.Models.Reports

{

public class VehicleModel

{

public int VehicleListId { get; set; }

public string VehSeriesCode { get; set; }

public int NumOfVehicles { get; set; }

public string ProjDesc { get; set; }

public DateTime DateEntered { get; set; }

public string EnteredBy { get; set; }

public string ModifiedLastBy { get; set; }

public DateTime? ModifiedLastDate { get; set; }

public string Make { get; set; }

public string Model { get; set; }

public int Year { get; set; }

public string Engine { get; set; }

public string Transmission { get; set; }

public string Differential { get; set; }

public string SopNumber { get; set; }

}

}

**Step 3: Controller**

File: Controllers/Reports/ReportController.cs Namespace: BCES.Controllers.Reports

csharp

Copy code

using BCES.Models.Reports;

using Dapper;

using Microsoft.AspNetCore.Mvc;

using System.Data;

using Telerik.Reporting;

using Telerik.Reporting.Processing;

namespace BCES.Controllers.Reports

{

public class ReportController : Controller

{

private readonly IDbConnection \_dbConnection;

public ReportController(IDbConnection dbConnection)

{

\_dbConnection = dbConnection;

}

[HttpGet]

public IActionResult GenerateVehicleReport()

{

try

{

// Fetch data from the database

string query = @"

SELECT TOP (1000) [VehicleListId],

[VehSeriesCode],

[NumOfVehicles],

[ProjDesc],

[DateEntered],

[EnteredBy],

[ModifiedLastBy],

[ModifiedLastDate],

[Make],

[Model],

[Year],

[Engine],

[Transmission],

[Differential],

[SopNumber]

FROM [SCES].[SCES].[VehicleList]";

var vehicles = \_dbConnection.Query<VehicleModel>(query).ToList();

// Generate Telerik Report

var report = new Telerik.Reporting.Report();

var reportProcessor = new ReportProcessor();

// Load the report template

var reportSource = new UriReportSource

{

Uri = "Reports/VehicleReport.trdp" // Path to the report template

};

// Add data source to the report

reportSource.Parameters.Add(new Parameter("VehicleData", vehicles));

// Render report

var result = reportProcessor.RenderReport("PDF", reportSource, null);

return File(result.DocumentBytes, "application/pdf", "VehicleReport.pdf");

}

catch (Exception ex)

{

// Log exception and return error

Console.Error.WriteLine(ex);

return StatusCode(500, "An error occurred while generating the report.");

}

}

}

}

**Step 4: Create the Telerik Report Template**

File: Reports/VehicleReport.trdp

1. Open **Telerik Report Designer**.
2. Create a new report.
3. Add a **SQL Data Source** to connect to the VehicleList table.
4. Design the report layout with fields like:
   * VehicleListId
   * VehSeriesCode
   * NumOfVehicles
   * ProjDesc, etc.
5. Save the file in the Reports folder as VehicleReport.trdp.

**Step 5: Configure Dependency Injection**

In Program.cs:

csharp

Copy code

using System.Data.SqlClient;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

builder.Services.AddTransient<IDbConnection>(sp => new SqlConnection(

builder.Configuration.GetConnectionString("DefaultConnection")

));

var app = builder.Build();

// Configure the HTTP request pipeline.

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.Run();

Add the database connection string to appsettings.json:

json

Copy code

{

"ConnectionStrings": {

"DefaultConnection": "Server=YOUR\_SERVER;Database=YOUR\_DATABASE;Trusted\_Connection=True;"

}

}

**Step 6: Create the View**

Create an empty view in Views/Reports/GenerateVehicleReport.cshtml:

html

Copy code

@{

ViewData["Title"] = "Generate Vehicle Report";

}

<h2>Vehicle Report</h2>

<a href="/Report/GenerateVehicleReport" target="\_blank">Download Report</a>

**Final Steps**

1. Run the application.
2. Navigate to /Report/GenerateVehicleReport to generate and download the report as a PDF.

This approach covers everything, including fetching data via Dapper, using a Telerik Report definition file, and generating a downloadable PDF.