

To integrate the Telerik Report Viewer into your ASP.NET Core MVC application, allowing users to view, print, and email reports, follow these steps:

**Setup Overview**

We will:

1. Configure the Telerik Report Viewer in a Razor View.
2. Serve the report via a custom ReportsController.
3. Add functionality for users to print or email the report.

**Step 1: Install Telerik Reporting Packages**

Run the following commands:

bash

Copy code

dotnet add package Telerik.Reporting

dotnet add package Telerik.Reporting.Services.AspNetCore

dotnet add package Dapper

**Step 2: Create the Report Viewer Controller**

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

This controller will handle serving reports to the Telerik Report Viewer.

csharp

Copy code

using Microsoft.AspNetCore.Mvc;

using Telerik.Reporting.Services;

using Telerik.Reporting.Services.AspNetCore;

namespace BCES.Controllers.Reports

{

[Route("api/reports")]

public class ReportsController : ReportsControllerBase

{

public ReportsController(IReportServiceConfiguration reportServiceConfiguration)

: base(reportServiceConfiguration)

{

}

}

}

**Step 3: Configure Telerik Report Services**

File: Program.cs Update Program.cs to configure the Telerik Report services.

csharp

Copy code

using System.Data.SqlClient;

using Telerik.Reporting.Services;

using Telerik.Reporting.Cache.File;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

// Add database connection

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

builder.Configuration.GetConnectionString("DefaultConnection")

));

// Configure Telerik Reporting Services

builder.Services.Configure<ReportServiceConfiguration>(config =>

{

config.HostAppId = "BCESReports";

config.Storage = new FileStorage("ReportsCache");

config.ReportResolver = new ReportFileResolver(

Path.Combine(builder.Environment.ContentRootPath, "Reports"));

});

var app = builder.Build();

// Middleware configuration

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.MapControllers(); // For Telerik Report Viewer API

app.Run();

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

File: Reports/VehicleReport.trdp

1. Open **Telerik Report Designer**.
2. Create a new report using the SQL data source with the query:

sql

Copy code

SELECT TOP (1000) [VehicleListId],

[VehSeriesCode],

[NumOfVehicles],

[ProjDesc],

[DateEntered],

[EnteredBy],

[ModifiedLastBy],

[ModifiedLastDate],

[Make],

[Model],

[Year],

[Engine],

[Transmission],

[Differential],

[SopNumber]

FROM [SCES].[SCES].[VehicleList]

1. Design the report layout as per your requirements.
2. Save it as VehicleReport.trdp in the Reports folder.

**Step 5: Add the Report Viewer Razor View**

File: Views/Reports/VehicleReportViewer.cshtml

html

Copy code

@{

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

}

<h2>Vehicle Report</h2>

<div id="reportViewer"></div>

<script src="https://kendo.cdn.telerik.com/2023.1.314/js/jquery.min.js"></script>

<script src="https://kendo.cdn.telerik.com/2023.1.314/js/kendo.all.min.js"></script>

<script src="https://cdn.telerik.com/reporting/js/telerikReportViewer-16.1.22.1121.min.js"></script>

<link href="https://cdn.telerik.com/reporting/css/telerikReportViewer-16.1.22.1121.min.css" rel="stylesheet" />

<script>

$(document).ready(function () {

$("#reportViewer").telerik\_ReportViewer({

serviceUrl: "/api/reports/",

templateUrl: "/report-viewer/templates/telerikReportViewerTemplate.html",

reportSource: {

report: "VehicleReport.trdp"

},

viewMode: "INTERACTIVE",

scaleMode: "FIT\_PAGE",

enableAccessibility: false

});

});

</script>

**Step 6: Add Report Viewer HTML Template**

File: wwwroot/report-viewer/templates/telerikReportViewerTemplate.html This file is required for the Telerik Report Viewer to function.

Copy the template from the Telerik Reporting package:

* Path: [YOUR\_TELERIK\_REPORTING\_INSTALLATION]/Html5/ReportViewer/templates/telerikReportViewerTemplate.html
* Save it to: wwwroot/report-viewer/templates/telerikReportViewerTemplate.html

**Step 7: Add Print and Email Options**

Enhance the VehicleReportViewer.cshtml view with print and email buttons:

html

Copy code

<h2>Vehicle Report</h2>

<div id="reportViewer"></div>

<button id="printButton">Print Report</button>

<button id="emailButton">Email Report</button>

<script>

$(document).ready(function () {

var reportViewer = $("#reportViewer").telerik\_ReportViewer({

serviceUrl: "/api/reports/",

templateUrl: "/report-viewer/templates/telerikReportViewerTemplate.html",

reportSource: {

report: "VehicleReport.trdp"

},

viewMode: "INTERACTIVE",

scaleMode: "FIT\_PAGE",

enableAccessibility: false

}).data("telerik\_ReportViewer");

// Print functionality

$("#printButton").click(function () {

reportViewer.printReport();

});

// Email functionality (dummy example)

$("#emailButton").click(function () {

alert("Email functionality not implemented in this example.");

});

});

</script>

**Step 8: Update Routing**

File: Controllers/Reports/ReportController.cs Add an action to navigate to the report viewer.

csharp

Copy code

using Microsoft.AspNetCore.Mvc;

namespace BCES.Controllers.Reports

{

public class ReportController : Controller

{

public IActionResult VehicleReportViewer()

{

return View();

}

}

}

**Step 9: Add Database Configuration**

File: appsettings.json Add your connection string:

json

Copy code

{

"ConnectionStrings": {

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

}

}

**Final Steps**

1. Run your application.
2. Navigate to /Report/VehicleReportViewer to access the Telerik Report Viewer.
3. Use the viewer's toolbar to print, export, or navigate through the report.