The error occurs because the RebuiltPartsController (and other controllers) do not have a **parameterless constructor**. Instead, they rely on dependency injection to provide their dependencies (e.g., DapperContext and IHttpContextAccessor).

To resolve this issue, we need to **inject the required dependencies** into the ExportController and use them to instantiate the other controllers. Here's how we can fix this:

**Step 1: Inject Dependencies into ExportController**

We need to inject the dependencies required by the other controllers (e.g., DapperContext and IHttpContextAccessor) into the ExportController.

**Updated ExportController.cs**

csharp

Copy

using Microsoft.AspNetCore.Mvc;

using OfficeOpenXml;

using System.Collections.Generic;

using System.IO;

using System.Threading.Tasks;

using BCES.Models.Parts;

using Microsoft.Extensions.Logging;

using BCES.Data;

using Microsoft.AspNetCore.Http;

using System;

public class ExportController : Controller

{

private readonly ILogger<ExportController> \_logger;

private readonly DapperContext \_dapperContext;

private readonly IHttpContextAccessor \_httpContextAccessor;

public ExportController(

ILogger<ExportController> logger,

DapperContext dapperContext,

IHttpContextAccessor httpContextAccessor)

{

\_logger = logger;

\_dapperContext = dapperContext;

\_httpContextAccessor = httpContextAccessor;

}

[HttpPost]

public async Task<IActionResult> SaveToExcel(string rebuiltPartNum)

{

try

{

// Fetch data from existing controllers

var mainGridData = await GetDataFromController<RebuiltPartsViewModel>("RebuiltParts", "GetRebuiltPartsView");

var labourDetails = await GetDataFromController<LabourDetailsRebuiltPartsViewModel>("LabourDetails", "GetLabourDetails", new { rebuiltPartNum });

var labourHourSummary = await GetDataFromController<LabourDetailsRebuiltPartsViewModel>("LabourHourSummaryInRbParts", "GetLabourHourSummaryInRbParts", new { rebuiltPartNum });

var materialCostSummary = await GetDataFromController<MaterialCostSummaryInRbViewModel>("MaterialCostSummaryInRbParts", "GetMaterialCostSummaryInRbParts", new { rebuiltPartNum });

var scParts = await GetDataFromController<StockCodedPartsViewModel>("ScPartsUsedInRbParts", "GetScPartsUsedInRbParts", new { rebuiltPartNum });

var nscParts = await GetDataFromController<NscPartsUsedViewModel>("NscPartsUsedInRbParts", "GetNscPartsUsedInRbParts", new { rebuiltPartNum });

var rbParts = await GetDataFromController<RebuiltPartsViewModel>("RbPartsUsedInRbParts", "GetRbPartsUsedInRbParts", new { rebuiltPartNum });

var totalCost = await GetDataFromController<TotalCostInRbModel>("TotalCostInRbParts", "GetTotalCostInRbParts", new { rebuiltPartNum });

// Create Excel package

using (var package = new ExcelPackage())

{

// Add main grid data

var mainGridSheet = package.Workbook.Worksheets.Add("Main Grid");

mainGridSheet.Cells.LoadFromCollection(mainGridData, true);

// Add subgrid data

var labourDetailsSheet = package.Workbook.Worksheets.Add("Labour Details");

labourDetailsSheet.Cells.LoadFromCollection(labourDetails, true);

var labourHourSummarySheet = package.Workbook.Worksheets.Add("Labour Hour Summary");

labourHourSummarySheet.Cells.LoadFromCollection(labourHourSummary, true);

var materialCostSummarySheet = package.Workbook.Worksheets.Add("Material Cost Summary");

materialCostSummarySheet.Cells.LoadFromCollection(materialCostSummary, true);

var scPartsSheet = package.Workbook.Worksheets.Add("SC Parts");

scPartsSheet.Cells.LoadFromCollection(scParts, true);

var nscPartsSheet = package.Workbook.Worksheets.Add("NSC Parts");

nscPartsSheet.Cells.LoadFromCollection(nscParts, true);

var rbPartsSheet = package.Workbook.Worksheets.Add("RB Parts");

rbPartsSheet.Cells.LoadFromCollection(rbParts, true);

var totalCostSheet = package.Workbook.Worksheets.Add("Total Cost");

totalCostSheet.Cells.LoadFromCollection(totalCost, true);

// Save the Excel file

var filePath = Path.Combine(Path.GetTempPath(), $"{rebuiltPartNum}\_Estimate.xlsx");

package.SaveAs(new FileInfo(filePath));

// Return the file for download

var fileBytes = await System.IO.File.ReadAllBytesAsync(filePath);

return File(fileBytes, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", $"{rebuiltPartNum}\_Estimate.xlsx");

}

}

catch (Exception ex)

{

\_logger.LogError(ex, "An error occurred while generating the Excel file.");

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

}

}

private async Task<IEnumerable<T>> GetDataFromController<T>(string controllerName, string actionName, object routeValues = null)

{

try

{

// Resolve the controller type

var controllerType = Type.GetType($"BCES.Controllers.Parts.{controllerName}Controller");

if (controllerType == null)

{

throw new InvalidOperationException($"Controller {controllerName} not found.");

}

// Create an instance of the controller using dependency injection

var controller = ActivatorUtilities.CreateInstance(HttpContext.RequestServices, controllerType);

// Get the action method

var methodInfo = controllerType.GetMethod(actionName);

if (methodInfo == null)

{

throw new InvalidOperationException($"Action {actionName} not found in controller {controllerName}.");

}

// Invoke the action method

var result = methodInfo.Invoke(controller, new[] { routeValues }) as Task<IActionResult>;

if (result == null)

{

throw new InvalidOperationException($"Action {actionName} did not return a valid result.");

}

// Await the result

var actionResult = await result;

if (actionResult is JsonResult jsonResult)

{

return (IEnumerable<T>)jsonResult.Value;

}

throw new InvalidOperationException($"Action {actionName} did not return a JSON result.");

}

catch (Exception ex)

{

\_logger.LogError(ex, $"An error occurred while fetching data from {controllerName}.{actionName}.");

throw;

}

}

}

**Step 2: Register Dependencies in Startup.cs (or Program.cs)**

Ensure that the required dependencies (DapperContext and IHttpContextAccessor) are registered in the dependency injection container.

**Startup.cs (or Program.cs in .NET 6+)**

csharp

Copy

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddLogging();

services.AddHttpContextAccessor();

services.AddScoped<DapperContext>(); // Register DapperContext

}

**Step 3: Explanation of Changes**

1. **Dependency Injection**:
   * The ExportController now receives DapperContext and IHttpContextAccessor via dependency injection.
   * These dependencies are passed to other controllers when they are instantiated.
2. **Controller Instantiation**:
   * Instead of using Activator.CreateInstance, we use ActivatorUtilities.CreateInstance to create controller instances with their dependencies.
3. **Error Handling**:
   * Added error handling to log issues when fetching data from other controllers.