**1. Update the View to Pass the Grid Identifier**

First, you need to modify the **TotalCost.chtml** view to pass an additional query parameter that identifies the grid. You can use a hidden field or a query parameter in the URL.

*// TotalCost.chtml*

@(Html.Kendo().Grid<MaterialCostSummaryInRbViewModel>()

.Name("TotalCostGrid")

.Columns(columns =>

{

columns.Bound(c => c.MaterialCost);

columns.Bound(c => c.LaborCost);

columns.Bound(c => c.TotalCost);

})

.DataSource(dataSource => dataSource

.Ajax()

.Read(read => read.Url(Url.Action("GetMaterialCostSummaryInRbParts", "MaterialCostSummaryInRbParts") + "?rebuiltPartNum=" + ViewBag.RebuiltPartNum + "&gridType=" + ViewBag.GridType).Type(HttpVerbs.Get))

)

)

Copied

**2. Update the Controller to Handle the Grid Identifier**

Next, update the **MaterialCostSummaryInRbPartsController** to accept the **gridType** parameter and use it to determine which SQL query to execute.

public class MaterialCostSummaryInRbPartsController : Controller

{

private readonly YourDbContext \_dbContext;

public MaterialCostSummaryInRbPartsController(YourDbContext dbContext)

{

\_dbContext = dbContext;

}

public async Task<JsonResult> GetMaterialCostSummaryInRbParts([DataSourceRequest] DataSourceRequest request, [FromQuery] string partNum, [FromQuery] string gridType)

{

var model = await GetMaterialCostSummaryInRbPartsData(partNum, gridType);

return Json(model.ToDataSourceResult(request));

}

private async Task<IEnumerable<MaterialCostSummaryInRbViewModel>> GetMaterialCostSummaryInRbPartsData(string partNum, string gridType)

{

string query = "";

switch (gridType)

{

case "VehicleGrid":

query = "SELECT \* FROM VehicleList WHERE VehicleNum = @partNum";

break;

case "MakeVsBuyGrid":

query = "SELECT \* FROM MbList WHERE MbNumber = @partNum";

break;

case "RBPartsGrid":

query = "SELECT \* FROM RBList WHERE RebuiltPartNum = @partNum";

break;

default:

throw new ArgumentException("Invalid grid type");

}

using (var connection = \_dbContext.Database.GetDbConnection())

{

var result = await connection.QueryAsync<MaterialCostSummaryInRbViewModel>(query, new { partNum});

return result.ToList();

}

}

}

Copied

**3. Update the View to\_Pass\_GridType**

Finally, ensure that\_the\_view\_passes\_the\_grid\_type\_to\_the\_controller\_action. You\_can\_do\_this\_by\_setting\_ViewBag.GridType\_in\_the\_controller\_action\_that\_opens\_the\_Telerik\_tabs.

*//Controller*

public class YourController : Controller

{

public IActionResult YourAction()

{

ViewBag.RebuiltPartNum = "examplePartNum";

ViewBag.GridType = "RBPartsGrid"; *// Set this\_based\_on\_which\_grid\_is\_selected*

return View();

}

}

Copied

**4. Example of Setting ViewBag.GridType in\_the\_Controller**

Here’s an example of how you might set **ViewBag.GridType** in the controller action that opens the Telerik tabs:

public class YourController : Controller

{

public IActionResult OpenTabs(string gridType, string partNum)

{

ViewBag.Rebuilt\_part\_num = partNum;

ViewBag.GridType = gridType; *//\_set\_this\_based\_on\_which\_grid\_is\_selected*

return View("TotalCost");

}

}

Copied

**5. Ensure the View Passes the Correct\_parameters**

Make sure that the view that opens the Telerik tabs passes\_the\_correct\_parameters\_to\_the\_grid.

//\_Your\_View\_that\_opens\_the\_Telerik\_tabs

@{

<div>

<button onclick="openTabs('VehicleGrid', 'exampleVehicleNum')">Open Vehicle Grid</button>

<button onclick="openTabs('MakeVsBuyGrid', 'exampleMbNumber')">Open MakeVsBuy Grid</button>

<button onclick="openTabs('RBPartsGrid', 'exampleRebuiltPart\_num')">Open RBParts Grid</button>

</div>

<div id="tabs">

<div id="tab1">

@(Html.Kendo().TabStrip().Name("tabStrip")

.Items(items =>

{

items.Add().Text("Labour\_Details").Content(@<text>

@Html.Partial("\_Labour\_Details", new { partNum = ViewBag.Rebuilt\_part\_num, gridType = ViewBag.Grid\_type })

</text>);

items.Add().Text("Total\_Cost").Content(@<text>

@Html.Partial("\_Total\_Cost", new { partNum = ViewBag.Rebuilt\_part\_num, grid\_type = ViewBag.GridType })

</text>);

})

)

</div>

</div>

}

<script>

function openTabs(gridType, partNum) {

var url = "@Url.Action("Get\_material\_cost\_summary\_in\_rb\_parts", "MaterialCostSummaryIn\_Rb\_parts") + "?rebuilt\_part\_num=" + partNum + "&gridType=" + gridType;

$("#tab1").load(url);

}

</script>

Copied

**Summary**

By following these steps, you can dynamically determine which grid is being accessed and execute the appropriate SQL query based on the grid type. This approach ensures that the **GetMaterialCostSummary\_in\_rb\_parts** action is reusable and can handle different grids by passing the **gridType** parameter.