**Updated Model: RebuiltPartsViewModel**

This includes the RebuiltStockNum, ListOfBus (displayed buses), and ListOfBusIds (selected IDs for the editor).

csharp

CopyEdit

namespace BCES.Models.Parts

{

public class RebuiltPartsViewModel

{

public string RebuiltStockNum { get; set; } // Unique identifier for the part

// Full list of bus objects for display purposes

public List<BusesModel> ListOfBus { get; set; } = new List<BusesModel>();

// List of selected bus IDs for inline editing

public List<int?> ListOfBusIds { get; set; } = new List<int?>();

}

public class BusesModel

{

public int? ListId { get; set; } // Nullable to handle cases where IDs are not set

public string Description { get; set; }

}

}

**2. Updated Controller: RebuiltPartsController**

The controller handles the GetRebuiltPartsView (grid data), UpdateRebuiltPart, and DeleteRebuiltPart actions. Query strings are used to pass RebuiltStockNum.

**Controller Code:**

csharp

CopyEdit

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Dapper;

namespace BCES.Controllers

{

public class RebuiltPartsController : Controller

{

private readonly IDbConnection \_dbConnection;

public RebuiltPartsController(IDbConnection dbConnection)

{

\_dbConnection = dbConnection;

}

// GET: Fetch RebuiltParts data for the grid

[HttpGet]

public async Task<JsonResult> GetRebuiltPartsView([DataSourceRequest] DataSourceRequest request)

{

try

{

var sql = @"

SELECT

rbm.RebuiltStockNum,

rbl.listid AS ListId,

lb.description AS Description

FROM

SBCES.RbMasterlist rbm

LEFT JOIN

SBCES.RBLISTOFBUSES rbl ON rbm.RebuiltStockNum = rbl.RebuiltStockNum

LEFT JOIN

SBCES.LISTOFBUSES lb ON rbl.listid = lb.listid";

var rebuiltParts = await \_dbConnection.QueryAsync<RebuiltPartsViewModel, BusesModel, RebuiltPartsViewModel>(

sql,

(rebuiltPart, bus) =>

{

if (bus != null)

{

rebuiltPart.ListOfBus.Add(bus);

if (bus.ListId.HasValue)

{

rebuiltPart.ListOfBusIds.Add(bus.ListId.Value);

}

}

return rebuiltPart;

},

splitOn: "ListId"

);

// Group data to aggregate buses for each part

var groupedData = rebuiltParts

.GroupBy(rp => rp.RebuiltStockNum)

.Select(group =>

{

var rebuiltPart = group.First();

rebuiltPart.ListOfBus = group.SelectMany(g => g.ListOfBus).Distinct().ToList();

rebuiltPart.ListOfBusIds = group.SelectMany(g => g.ListOfBusIds).Distinct().ToList();

return rebuiltPart;

})

.ToList();

return Json(groupedData.ToDataSourceResult(request));

}

catch (Exception ex)

{

return Json(new { error = $"Error loading rebuilt parts: {ex.Message}" });

}

}

// POST: Update a RebuiltPart

[HttpPost]

public async Task<IActionResult> UpdateRebuiltPart(string RebuiltStockNum, [FromBody] RebuiltPartsViewModel model)

{

try

{

if (model == null || string.IsNullOrEmpty(RebuiltStockNum))

{

return BadRequest("Invalid data.");

}

// Update the RebuiltPart in the database

var updatePartSql = @"

UPDATE SBCES.RbMasterlist

SET ... -- Add fields to update here

WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(updatePartSql, model);

// Update associated buses

var deleteOldBusesSql = "DELETE FROM SBCES.RBLISTOFBUSES WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(deleteOldBusesSql, new { RebuiltStockNum });

if (model.ListOfBusIds?.Any() == true)

{

var insertNewBusesSql = @"

INSERT INTO SBCES.RBLISTOFBUSES (RebuiltStockNum, ListId)

VALUES (@RebuiltStockNum, @ListId)";

foreach (var listId in model.ListOfBusIds.Where(id => id.HasValue))

{

await \_dbConnection.ExecuteAsync(insertNewBusesSql, new { RebuiltStockNum, ListId = listId.Value });

}

}

return Json(new { success = true });

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

// POST: Delete a RebuiltPart

[HttpPost]

public async Task<IActionResult> DeleteRebuiltPart(string RebuiltStockNum)

{

try

{

if (string.IsNullOrEmpty(RebuiltStockNum))

{

return BadRequest("Invalid data.");

}

var deletePartSql = "DELETE FROM SBCES.RbMasterlist WHERE RebuiltStockNum = @RebuiltStockNum";

var deleteBusesSql = "DELETE FROM SBCES.RBLISTOFBUSES WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(deleteBusesSql, new { RebuiltStockNum });

await \_dbConnection.ExecuteAsync(deletePartSql, new { RebuiltStockNum });

return Json(new { success = true });

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

}

}

**3. View: Index.cshtml**

The grid includes **inline editing**, and query strings are used in the Read, Update, and Delete actions.

html

CopyEdit

@model IEnumerable<BCES.Models.Parts.RebuiltPartsViewModel>

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

.Name("RebuiltPartsGrid")

.Columns(columns =>

{

columns.Bound(c => c.RebuiltStockNum)

.Title("Rebuilt Stock Code")

.Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.ListOfBus)

.Title("Vehicle Series")

.ClientTemplate("# if (ListOfBus && ListOfBus.length > 0) { # #= ListOfBus.map(bus => bus.Description).join(', ') # # } else { # - # } #");

columns.Command(command =>

{

command.Edit().Text("Edit");

command.Destroy().Text("Delete");

}).Title("&nbsp;").Width(150);

})

.Editable(editable => editable.Mode(GridEditMode.InLine)) // Inline editing

.Pageable()

.Sortable()

.Scrollable()

.Filterable()

.DataSource(dataSource => dataSource

.Ajax()

.Read(read => read.Url(Url.Action("GetRebuiltPartsView", "RebuiltParts")).Type(HttpVerbs.Get))

.Update(update => update.Url(Url.Action("UpdateRebuiltPart", "RebuiltParts")).Data("sendQuery").Type(HttpVerbs.Post))

.Destroy(destroy => destroy.Url(Url.Action("DeleteRebuiltPart", "RebuiltParts")).Data("sendQuery").Type(HttpVerbs.Post))

.Model(model =>

{

model.Id(p => p.RebuiltStockNum); // Specify the primary key

model.Field(p => p.ListOfBusIds).DefaultValue(new List<int?>());

})

)

)

<script>

function sendQuery(data) {

data.RebuiltStockNum = $("#RebuiltPartsGrid").data("kendoGrid").dataItem($(data).closest("tr")).RebuiltStockNum;

}

</script>

**4. Editor Template: VehicleSeriesEditor.cshtml**

This template provides a dropdown with multiple selections.

html

CopyEdit

@model List<int?>

<select id="VehicleSeries" name="ListOfBusIds" class="form-control" multiple>

<option value="">-- Select Vehicle Series --</option>

</select>

<script>

$(document).ready(function () {

$.getJSON('@Url.Action("GetVehicleSeries", "RebuiltParts")', function (data) {

var select = $("#VehicleSeries");

select.empty();

$.each(data, function (index, item) {

select.append($("<option>").val(item.listId).text(item.description));

});

var selectedValues = @Html.Raw(Json.Serialize(Model));

select.val(selectedValues);

});

});

</script>

This implementation ensures:

1. Query strings (RebuiltStockNum) are used for **Read**, **Update**, and **Delete** operations.
2. Full RebuiltPartsViewModel (including ListOfBusIds and ListOfBus) is passed during updates.
3. Exception handling is added to all actions.