The **Telerik MultiSelect** will enable the user to add or remove bus descriptions associated with a RebuiltStockNum. The data updates will then be sent back to the server, where the changes will be persisted in the RBLISTOFBUSES and LISTOFBUSES tables.

Below is the updated implementation to support updating BusesModel for RebuiltPartsViewModel, including the use of a **Telerik MultiSelect control**.

**Revised RebuiltPartsViewModel**

The RebuiltPartsViewModel already has a collection of BusesModel, so no changes are needed in the model itself. However, ensure that it contains the necessary fields for both display and editing.

csharp

CopyEdit

namespace BCES.Models.Parts

{

public class RebuiltPartsViewModel

{

public string RebuiltStockNum { get; set; } // Primary Key

public string Keyword { get; set; } // For search or filtering

public List<BusesModel> Buses { get; set; } = new List<BusesModel>(); // Associated buses

}

public class BusesModel

{

public int ListId { get; set; } // Unique identifier for the bus

public string Description { get; set; } // Description of the bus

}

}

**Revised Controller Logic**

We need to handle updates to the BusesModel (associated buses) in addition to the RebuiltPartsViewModel. When the user updates the buses for a specific RebuiltStockNum, we must update the RBLISTOFBUSES table.

**Updated GetRebuiltPartsView Action**

This action retrieves data for both the grid and the MultiSelect.

csharp

CopyEdit

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

{

try

{

var query = @"

SELECT

rbm.RebuiltStockNum,

rbm.Keyword,

rbl.ListId,

lb.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 rebuiltPartsDict = new Dictionary<string, RebuiltPartsViewModel>();

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

query,

(part, bus) =>

{

if (!rebuiltPartsDict.TryGetValue(part.RebuiltStockNum, out var rebuiltPart))

{

rebuiltPart = part;

rebuiltPartsDict.Add(rebuiltPart.RebuiltStockNum, rebuiltPart);

}

if (bus != null)

{

rebuiltPart.Buses.Add(bus);

}

return rebuiltPart;

},

splitOn: "ListId"

);

return Json(rebuiltPartsDict.Values.ToDataSourceResult(request));

}

catch (Exception ex)

{

return Json(new DataSourceResult { Errors = "Error occurred while fetching rebuilt parts." });

}

}

**Update Update Action**

The Update action now handles both the main RebuiltPartsViewModel and the associated BusesModel.

csharp

CopyEdit

[HttpPost]

public async Task<JsonResult> Update([DataSourceRequest] DataSourceRequest request, RebuiltPartsViewModel model)

{

if (!ModelState.IsValid)

{

return Json(new DataSourceResult { Errors = ModelState });

}

try

{

// Update the RebuiltPartsViewModel data

var updatePartQuery = @"

UPDATE SBCES.RbMasterlist

SET Keyword = @Keyword

WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(updatePartQuery, new { model.RebuiltStockNum, model.Keyword });

// Update the associated BusesModel data (clear and re-insert for simplicity)

var deleteBusesQuery = @"

DELETE FROM SBCES.RBLISTOFBUSES

WHERE RebuiltStockNum = @RebuiltStockNum";

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

if (model.Buses != null && model.Buses.Any())

{

var insertBusesQuery = @"

INSERT INTO SBCES.RBLISTOFBUSES (RebuiltStockNum, ListId)

VALUES (@RebuiltStockNum, @ListId)";

foreach (var bus in model.Buses)

{

await \_dbConnection.ExecuteAsync(insertBusesQuery, new { model.RebuiltStockNum, bus.ListId });

}

}

return Json(new[] { model }.ToDataSourceResult(request));

}

catch (Exception ex)

{

return Json(new DataSourceResult { Errors = "Error occurred while updating the rebuilt part." });

}

}

**Updated Index.cshtml**

Here’s the updated Razor view. The grid includes a **Telerik MultiSelect control** for the Buses column.

html

CopyEdit

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

<h1>Rebuilt Parts</h1>

@(Html.Kendo().Grid<BCES.Models.Parts.RebuiltPartsViewModel>()

.Name("rebuiltPartsGrid")

.Columns(columns =>

{

columns.Bound(p => p.RebuiltStockNum).Title("Rebuilt Stock #").Width(150).Editable(false); // Primary key

columns.Bound(p => p.Keyword).Title("Keyword").Width(200);

// MultiSelect column for buses

columns.Bound(p => p.Buses)

.ClientTemplate("#= Buses.map(function(bus) { return bus.Description; }).join(', ') #")

.EditorTemplateName("MultiSelectEditor") // Use custom MultiSelect editor

.Title("Buses").Width(300);

columns.Command(command =>

{

command.Edit(); // Inline Edit Command

command.Destroy(); // Delete Command

}).Title("Actions").Width(150);

})

.ToolBar(toolbar => toolbar.Create()) // Create Button

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

.Pageable()

.Sortable()

.Filterable()

.DataSource(dataSource => dataSource

.Ajax()

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

.Create(create => create.Url(Url.Action("Create", "RebuiltParts")).Type(HttpVerbs.Post))

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

.Destroy(destroy => destroy.Url(Url.Action("Delete", "RebuiltParts")).Type(HttpVerbs.Post))

.Model(model =>

{

model.Id(p => p.RebuiltStockNum); // Primary key

model.Field(p => p.RebuiltStockNum).Editable(false); // PK is not editable

})

)

)

<script>

$(document).ready(function () {

console.log("Rebuilt Parts page loaded.");

});

</script>

**MultiSelect Editor Template**

The MultiSelectEditor allows users to select buses using a Telerik MultiSelect. This is added as a custom editor template.

**Views/Shared/EditorTemplates/MultiSelectEditor.cshtml**

html

CopyEdit

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

@(Html.Kendo().MultiSelect()

.Name("Buses") // Name must match the field in the model

.DataTextField("Description") // Display description

.DataValueField("ListId") // Bind the value to ListId

.BindTo((System.Collections.IEnumerable)ViewData["BusesList"]) // Pass a list of all buses from the controller

)

**Controller for MultiSelect Data**

Pass the list of all available buses to the MultiSelectEditor using ViewData.

csharp

CopyEdit

public async Task<IActionResult> Index()

{

var query = "SELECT ListId, Description FROM SBCES.LISTOFBUSES";

var busesList = await \_dbConnection.QueryAsync<BusesModel>(query);

ViewData["BusesList"] = busesList;

return View();

}

**Summary**

1. **Telerik MultiSelect**: Allows users to select multiple buses for a RebuiltStockNum.
2. **Controller Updates**:
   * Fetch all buses for the MultiSelectEditor.
   * Handle updates for BusesModel in the Update action.
3. **Custom Editor Template**: The MultiSelect control is defined in an editor template (MultiSelectEditor).