**1. BusesModel.cs**

No changes needed here.

csharp

Copy

public class BusesModel

{

public int ListId { get; set; }

public string Description { get; set; } // Should be displayed in the column

}

**2. RebuiltPartsViewModel.cs**

No changes needed here.

csharp

Copy

public class RebuiltPartsViewModel

{

public string RebuiltStockNum { get; set; }

// Other properties...

public List<BusesModel> Buses { get; set; }

}

**3. Index.cshtml**

Add the multi-select column to the grid.

html

Copy

@using BCES.Models.Parts

@using BCES.Models.Common

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

@{

ViewData["Title"] = "Rebuilt Parts";

}

<h4>Rebuilt Parts</h4>

@(Html.Kendo().TabStrip()

.Name("rebuildPartsTabStrip")

.Animation(animation => animation.Open(effect => effect.Fade(FadeDirection.In)))

.Items(tabstrip =>

{

tabstrip.Add().Text("Rebuilt Parts")

.Selected(true) // Ensure the first tab is selected by default

.Content(@<text>

@try

{

@(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.MmsStockCode).Title("MMS Stock Code").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.DetailedDesc).Title("Description").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.Keyword).Title("Keyword").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.JobNumber).Title("Job Number").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.CorePartNum).Title("Core Code").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.CoreCharge).Title("Core Cost").Filterable(ftb => ftb.Cell(cell => cell.Operator("gte"))).Format("{0:C2}").EditorTemplateName("Currency");

columns.Bound(c => c.BuyNewCost).Title("Buy Cost").Filterable(ftb => ftb.Cell(cell => cell.Operator("gte"))).Format("{0:C2}").EditorTemplateName("Currency");

// Add Telerik Multiselect column

columns.Bound(c => c.Buses)

.Title("Vehicle Series")

.ClientTemplate("#= getBusesDescription(Buses) #")

.EditorTemplateName("VehicleSeriesEditor");

columns.Command(command =>

{

command.Edit().Text(" ").IconClass("k-icon k-i-edit").HtmlAttributes(new { title = "Edit" });

command.Custom("Archive").Text(" ").IconClass("k-icon k-i-folder").Click("onArchiveClick").HtmlAttributes(new { title = "Archive" });

command.Custom("Save to Excel").Text(" ").IconClass("k-icon k-i-file-excel").Click("onSaveToExcelClick").HtmlAttributes(new { title = "Save to Excel" });

command.Custom("Email Estimate").Text(" ").IconClass("k-icon k-i-email").Click("onEmailEstimateClick").HtmlAttributes(new { title = "Email Estimate" });

}).Title("Actions").Width(200).HtmlAttributes(new { @class = "action-buttons" });

})

.ToolBar(toolbar => { toolbar.Create(); })

.Editable(editable => editable.Mode(GridEditMode.InLine))

.Pageable()

.Sortable()

.ClientDetailTemplateId("template")

.Resizable(r => r.Columns(true))

.Filterable(ftb => ftb.Mode(GridFilterMode.Row))

.DataSource(dataSource => dataSource

.Ajax()

.Model(model =>

{

model.Id(c => c.RbMasterlistId);

model.Field(c => c.RbMasterlistId);

model.Field(c => c.RebuiltStockNum).Editable(true);

model.Field(c => c.MmsStockCode).Editable(true);

model.Field(c => c.LabourDetailsRebuiltParts).DefaultValue(new List<BCES.Models.Parts.LabourDetailsRebuiltPartsViewModel>()).Editable(false);

model.Field(u => u.Buses).DefaultValue(new List<BCES.Models.Parts.BusesModel>()).Editable(false);

})

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

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

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

)

)

}

catch (Exception ex)

{

<div class="alert alert-danger">Error loading rebuilt parts: @ex.Message</div>

}

</text>);

tabstrip.Add().Text("Archived Parts")

.Content(@<text>

@try

{

@Html.Partial("\_ArchivedIndex")

}

catch (Exception ex)

{

<div class="alert alert-danger">Error loading archived parts: @ex.Message</div>

}

</text>);

})

)

@section Scripts {

<script>

function getBusesDescription(buses) {

if (buses && buses.length > 0) {

return buses.map(function(bus) {

return bus.Description;

}).join(", ");

}

return "";

}

function onBusChange(e) {

var selectedBuses = this.value();

console.log("Selected buses:", selectedBuses);

$.ajax({

url: '@Url.Action("GetSelectedBuses", "RebuiltParts")',

type: 'POST',

contentType: 'application/json',

data: JSON.stringify({ buses: selectedBuses }),

success: function (data) {

console.log('Success:', data);

},

error: function (error) {

console.error('Error:', error);

}

});

}

function onBusDataBound() {

console.log("event: busdataBound");

}

</script>

}

Run HTML

**4. EditorTemplate/VehicleSeriesEditor.cshtml**

Create this file if it doesn't exist.

html

Copy

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

@(Html.Kendo().MultiSelectFor(m => m)

.Name("VehicleSeries")

.DataTextField("Description")

.DataValueField("ListId")

.Placeholder("Select vehicle ...")

.Filter("contains")

.BindTo(Model)

.AutoBind(true)

.DataSource(source =>

{

source.Read(read =>

{

read.Url(Url.Action("GetVehicleSeries", "RebuiltParts")).Type(HttpVerbs.Get);

})

.ServerFiltering(true);

})

.Events(e =>

{

e.Change("onBusChange").DataBound("onBusDataBound");

})

)

Run HTML

**5. RebuiltPartsController.cs**

Ensure the GetVehicleSeries action is correctly implemented.

csharp

Copy

public class RebuiltPartsController : Controller

{

private readonly IDbConnection \_dbConnection;

public RebuiltPartsController(IDbConnection dbConnection)

{

\_dbConnection = dbConnection;

}

private async Task<IEnumerable<RebuiltPartsViewModel>> GetRebuiltPartsData()

{

try

{

var rebuiltPartsQuery = @"

SELECT rbm.RbMasterlistId,

rbm.[RebuiltStockNum],

rbm.[MmsStockCode],

rbm.[Keyword],

rbm.JobNumber,

rbm.[CorePartNum],

rbm.[DetailedDesc],

rbm.[CoreCharge],

rbm.[EstimatedCost],

rbm.[BuyNewCost],

rbm.[RemanCost],

rbm.[ExternalCost],

rbl.listid,

lb.description

FROM [SBCES].[RbMasterlist] rbm

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

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

var rebuiltParts = (await \_dbConnection.QueryAsync<RebuiltPartsViewModel>(rebuiltPartsQuery)).ToList();

return rebuiltParts;

}

catch (Exception ex)

{

// Handle errors gracefully

return null;

}

}

[HttpGet]

public async Task<IActionResult> GetVehicleSeries()

{

try

{

var sql = @"

SELECT

RBLISTOFBUSES.listid AS ListId,

LISTOFBUSES.description AS Description

FROM

SBCES.RBLISTOFBUSES

INNER JOIN

SBCES.LISTOFBUSES ON RBLISTOFBUSES.listid = LISTOFBUSES.listid";

var vehicleSeries = await \_dbConnection.QueryAsync<BusesModel>(sql);

return Json(vehicleSeries);

}

catch (Exception ex)

{

return StatusCode(500, "Internal server error: " + ex.Message);

}

}

[HttpPost]

public IActionResult GetSelectedBuses([FromBody] List<int> buses)

{

try

{

// Process the selected buses here

return Json(new { success = true });

}

catch (Exception ex)

{

return StatusCode(500, "Internal server error: " + ex.Message);

}

}

}

**Summary of Changes:**

1. **Index.cshtml**: Added the multi-select column to the grid and included a client template to display the selected buses.
2. **EditorTemplate/VehicleSeriesEditor.cshtml**: Created a new editor template for the multi-select column.
3. **RebuiltPartsController.cs**: Ensured the GetVehicleSeries action is correctly implemented and added a new GetSelectedBuses action to handle the selected buses.