The key issues here are related to data binding and Kendo MultiSelect configuration. Let's fix this step by step:

1. **Fix ViewModel Structure**:

csharp

Copy

// RebuiltPartsViewModel.cs

public class RebuiltPartsViewModel

{

// ... other properties

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

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

}

1. **Update Grid Column Configuration**:

html

Copy

@\* Index.cshtml \*@

columns.Bound(c => c.ListOfBusIds)

.Title("Vehicle Series")

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

.EditorTemplateName("VehicleSeriesEditor");

Run HTML

1. **Fix Editor Template**:

html

Copy

@\* VehicleSeriesEditor.cshtml \*@

@model List<int>

@(Html.Kendo().MultiSelect()

.Name("ListOfBusIds") // Match the ViewModel property name

.DataTextField("Description")

.DataValueField("ListId")

.Placeholder("Select vehicle series...")

.AutoBind(false)

.DataSource(source => source

.Read(read => read.Action("GetVehicleSeries", "RebuiltParts"))

.Value(Model)

)

Run HTML

1. **Update Controller Method**:

csharp

Copy

[HttpGet]

[AllowAnonymous]

public async Task<IActionResult> GetVehicleSeries()

{

try

{

var sql = "SELECT ListId, Description FROM SBCES.ListOfBuses";

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

return Json(vehicleSeries.Where(vs => vs.ListId.HasValue));

}

catch (Exception ex)

{

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

}

}

1. **Update Grid DataSource Model**:

csharp

Copy

.Model(model =>

{

// ... other model configurations

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

})

1. **Fix JavaScript Initialization**:  
   Remove this problematic code from your scripts:

javascript

Copy

$(document).on("kendo:editor-opened", function () {

$("#ListOfBusIds").trigger("change");

});

**Key Changes Explained**:

1. The ViewModel now has both:
   * ListOfBusIds for storing selected IDs (used for binding)
   * ListOfBus for displaying full objects (used in client template)
2. The MultiSelect now binds to the IDs list instead of complex objects
3. The controller method now fetches directly from ListOfBuses table instead of through the junction table
4. The editor template now uses proper naming and binding for ID values
5. The SQL query in GetVehicleSeries was simplified to get all available options

**Additional Recommendations**:

1. Update Kendo UI to a newer version (2021.2.511 is over 2 years old)
2. Verify jQuery compatibility with your Kendo version
3. Add error handling to your MultiSelect initialization:

javascript

Copy

.Events(e => e.Change("onVehicleSeriesChange").DataBound("onVehicleSeriesDataBound"))

1. Add loading indicators:

csharp

Copy

.DataSource(source => source

.Read(read => read.Action("GetVehicleSeries", "RebuiltParts"))

.ServerFiltering(true)

)

1. Update your update method to handle the IDs:

csharp

Copy

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> UpdateRebuiltPartsView(...)

{

// ... existing code

// Update buses using ListOfBusIds

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

{

var insertNewBusesSql = @"

INSERT INTO SBCES.RbListOfBuses (RebuiltStockNum, ListId)

VALUES (@RebuiltStockNum, @ListId)";

foreach (var listId in model.ListOfBusIds)

{

await \_dbConnection.ExecuteAsync(insertNewBusesSql, new

{

model.RebuiltStockNum,

ListId = listId

});

}

}

// ... rest of code

}

These changes should resolve both the controller method not being called and the JavaScript error. The MultiSelect will now properly bind to the ID values while maintaining the display of descriptions in the grid.