**Root Cause Analysis**

The exception is likely caused by one of the following:

* **Model is null**: The Buses property of your RebuiltPartsViewModel might be null, causing the Value(Model.Select(b => b.ListId)) in the MultiSelect to throw an exception.
* **ViewData["BusesList"] is null**: The BindTo method is trying to bind to a null or improperly cast ViewData["BusesList"], which throws an exception.
* **ViewData.TemplateInfo.HtmlFieldPrefix is null**: If the HtmlFieldPrefix is null, the Name() method could fail.
* **Improper column or MultiSelect configuration**: If the Buses column is configured incorrectly or the grid cannot render the column properly, it might fail before the EditorTemplate is even invoked.

**2. Immediate Fixes to Validate Dependencies**

**Fix for Model Null Check**

To prevent exceptions related to Model.Select(b => b.ListId), ensure the Model is never null in the EditorTemplate. Update the VehicleSeriesEditor.cshtml as follows:

razor

CopyEdit

@model List<BCES.Models.Parts.BusesModel> // Ensure Model is always passed

@(Html.Kendo().MultiSelect()

.Name((string)ViewData.TemplateInfo.HtmlFieldPrefix ?? "DefaultFieldPrefix") // Prevent null exceptions

.DataTextField("Description")

.DataValueField("ListId")

.Value(Model?.Select(b => b.ListId) ?? new List<int>()) // Use an empty list if Model is null

.BindTo((System.Collections.IEnumerable)(ViewData["BusesList"] ?? new List<BCES.Models.Parts.BusesModel>())) // Ensure BindTo is safe

)

Here:

* **Model?.Select(b => b.ListId) ?? new List<int>()** ensures that even if Model is null, the Value property receives a valid empty list.
* **ViewData["BusesList"] ?? new List<BCES.Models.Parts.BusesModel>()** ensures that the BindTo method binds to an empty list instead of null.

**Fix for Null ViewData["BusesList"]**

Ensure that your controller action sets the ViewData["BusesList"] properly. For example:

csharp

CopyEdit

public IActionResult RebuiltPartsGrid()

{

// Populate the list of buses

var busesList = \_dbContext.Buses.Select(bus => new BusesModel

{

ListId = bus.Id,

Description = bus.Name

}).ToList();

ViewData["BusesList"] = busesList;

// Pass the grid model (e.g., RebuiltPartsViewModel) to the view

var rebuiltParts = \_dbContext.RebuiltParts.Select(part => new RebuiltPartsViewModel

{

RebuiltStockNum = part.StockNum,

Buses = new List<BusesModel> // Example pre-selected buses

{

new BusesModel { ListId = 1, Description = "Bus 1" },

new BusesModel { ListId = 2, Description = "Bus 2" }

}

}).ToList();

return View(rebuiltParts);

}

Verify:

* The ViewData["BusesList"] is populated with valid data.
* The Buses property in each RebuiltPartsViewModel instance is initialized with at least an empty list, to avoid null issues.

**3. Debugging at the Controller or Grid Level**

If you suspect the exception occurs before the EditorTemplate is invoked, try adding debugging statements or logging at the controller or grid level.

**Add Logging to Your Controller**

You can use Console.WriteLine, Debug.WriteLine, or a logger (like ILogger) to verify what data is being passed to the view:

csharp

CopyEdit

public IActionResult RebuiltPartsGrid()

{

var busesList = \_dbContext.Buses.Select(bus => new BusesModel

{

ListId = bus.Id,

Description = bus.Name

}).ToList();

Console.WriteLine("BusesList count: " + busesList.Count); // Log busesList count

ViewData["BusesList"] = busesList;

var rebuiltParts = \_dbContext.RebuiltParts.Select(part => new RebuiltPartsViewModel

{

RebuiltStockNum = part.StockNum,

Buses = new List<BusesModel>

{

new BusesModel { ListId = 1, Description = "Bus 1" },

new BusesModel { ListId = 2, Description = "Bus 2" }

}

}).ToList();

Console.WriteLine("RebuiltParts count: " + rebuiltParts.Count); // Log rebuiltParts count

return View(rebuiltParts);

}

**Debugging Grid Columns**

To ensure the Grid columns are properly set up, temporarily comment out the EditorTemplate reference and log the grid configuration:

razor

CopyEdit

@(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)));

// MultiSelect column for buses

columns.Bound(p => p.Buses)

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

//.EditorTemplateName("VehicleSeriesEditor") // Comment this temporarily

.Title("Buses")

.Width(300);

})

)

If the grid works when the EditorTemplate is commented out, the problem is likely with the EditorTemplate's dependencies (e.g., Model or ViewData["BusesList"]).

**4. Additional Recommendations**

* **Default Values for Missing Data**: Ensure that all nullable fields have fallback values. For example:
  + Use Model ?? new List<BusesModel>() for the MultiSelect.
  + Use ViewData["BusesList"] ?? new List<BusesModel>() for the BindTo.
* **Custom Validation Logic**: Add validation for null or empty properties in your controller, and throw meaningful exceptions (e.g., "BusesList is null") to pinpoint the issue.

**5. Enhanced Exception Handling**

Modify the catch block in your grid code to include more details about the error:

razor

CopyEdit

@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)));

// MultiSelect column for buses

columns.Bound(p => p.Buses)

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

.EditorTemplateName("VehicleSeriesEditor")

.Title("Buses")

.Width(300);

})

)

}

catch (Exception ex)

{

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

<div class="alert alert-danger">@ex.StackTrace</div> <!-- Print stack trace -->

}

**Summary of Steps**

1. Ensure ViewData["BusesList"] and Model are properly initialized in the controller.
2. Validate data dependencies using null checks (?? fallback values) in the VehicleSeriesEditor.
3. Log or debug at the controller and grid level to pinpoint the issue.
4. Add enhanced exception handling to capture and print error details.