**1. Debugging Model.Select(b => b.ListId)**

To validate the Model and Model.Select(b => b.ListId) values, add a block of Razor code at the top of your VehicleSeriesEditor.cshtml file, like this:

razor

CopyEdit

@{

// Check if the Model is null

if (Model == null)

{

<div style="color: red;">Error: Model is null</div>

}

else

{

// If Model is not null, print the values

<div style="color: green;">Model is not null. ListId values:</div>

foreach (var item in Model)

{

<div>ListId: @item.ListId, Description: @item.Description</div>

}

}

}

This will help you confirm whether the Model or any of its items are null and whether the ListId property is accessible.

**2. Debugging ViewData.TemplateInfo.HtmlFieldPrefix**

To check the value of ViewData.TemplateInfo.HtmlFieldPrefix and ensure it’s not null, you can log or print it like so:

razor

CopyEdit

@{

var htmlFieldPrefix = (string)ViewData.TemplateInfo.HtmlFieldPrefix;

if (string.IsNullOrEmpty(htmlFieldPrefix))

{

<div style="color: red;">Error: ViewData.TemplateInfo.HtmlFieldPrefix is null or empty</div>

}

else

{

<div style="color: green;">ViewData.TemplateInfo.HtmlFieldPrefix: @htmlFieldPrefix</div>

}

}

If ViewData.TemplateInfo.HtmlFieldPrefix is null, this may indicate an issue in how the template is being called or bound in your grid.

**3. Debugging ViewData["BusesList"]**

Finally, check if ViewData["BusesList"] is null or improperly set by adding the following code:

razor

CopyEdit

@{

var busesList = ViewData["BusesList"] as IEnumerable<BCES.Models.Parts.BusesModel>;

if (busesList == null)

{

<div style="color: red;">Error: ViewData["BusesList"] is null or not properly cast</div>

}

else

{

<div style="color: green;">ViewData["BusesList"] is not null. Available buses:</div>

foreach (var bus in busesList)

{

<div>ListId: @bus.ListId, Description: @bus.Description</div>

}

}

}

This will help verify that the ViewData["BusesList"] object is properly initialized and contains the data you expect.

**Full Updated VehicleSeriesEditor.cshtml**

Here’s how your VehicleSeriesEditor.cshtml might look after adding debugging statements:

razor

CopyEdit

@{

// Debugging Model

if (Model == null)

{

<div style="color: red;">Error: Model is null</div>

}

else

{

<div style="color: green;">Model is not null. ListId values:</div>

foreach (var item in Model)

{

<div>ListId: @item.ListId, Description: @item.Description</div>

}

}

// Debugging HtmlFieldPrefix

var htmlFieldPrefix = (string)ViewData.TemplateInfo.HtmlFieldPrefix;

if (string.IsNullOrEmpty(htmlFieldPrefix))

{

<div style="color: red;">Error: ViewData.TemplateInfo.HtmlFieldPrefix is null or empty</div>

}

else

{

<div style="color: green;">ViewData.TemplateInfo.HtmlFieldPrefix: @htmlFieldPrefix</div>

}

// Debugging ViewData["BusesList"]

var busesList = ViewData["BusesList"] as IEnumerable<BCES.Models.Parts.BusesModel>;

if (busesList == null)

{

<div style="color: red;">Error: ViewData["BusesList"] is null or not properly cast</div>

}

else

{

<div style="color: green;">ViewData["BusesList"] is not null. Available buses:</div>

foreach (var bus in busesList)

{

<div>ListId: @bus.ListId, Description: @bus.Description</div>

}

}

}

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

@(Html.Kendo().MultiSelect()

.Name((string)ViewData.TemplateInfo.HtmlFieldPrefix) // Dynamically set the name

.DataTextField("Description") // Display Description in the MultiSelect dropdown

.DataValueField("ListId") // Bind to ListId for the values

.Value(Model.Select(b => b.ListId)) // Bind only the ListId values

.BindTo((System.Collections.IEnumerable)ViewData["BusesList"]) // Provide the full list of available buses

)