To achieve the desired functionality of using the Telerik MultiSelect control to display vehicle series in the Kendo Grid, you need to make several adjustments to your code. Here's a step-by-step guide to help you implement this:

### 1. Update the Model

Ensure that the `RebuiltPartsViewModel` can handle multiple vehicle series. You already have a `List<ListOfBusesModel> VehicleSeries` property, which is good.

### 2. Update the Controller

Modify the `GetRebuiltPartsData` method to populate the `VehicleSeries` property correctly.

### 3. Create the Editor Template

Create an editor template for the MultiSelect control.

### 4. Update the Grid Configuration

Configure the grid to use the editor template for the `VehicleSeries` column.

### 5. Update the View

Ensure the view is correctly set up to use the editor template.

### Step-by-Step Implementation

#### Step 1: Update the Model

Your model is already set up correctly, so no changes are needed here.

#### Step 2: Update the Controller

Modify the `GetRebuiltPartsData` method to populate the `VehicleSeries` property correctly.

```csharp

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

{

try

{

var query = @"

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],

lb.ListId,

lb.Description AS VehicleSeries

FROM [SBCES].[RbMasterlist] rbm

INNER JOIN [SBCES].[RbListOfBuses] rlb ON rlb.[RebuiltStockNum] = rbm.[RebuiltStockNum]

INNER JOIN [SBCES].[ListOfBuses] lb ON rlb.ListId = lb.ListId";

var rbParts = \_dbConnection.Query<RebuiltPartsViewModel, ListOfBusesModel, RebuiltPartsViewModel>(

query,

(rbPart, vehicle) =>

{

rbPart.VehicleSeries.Add(vehicle);

return rbPart;

},

splitOn: "ListId"

).GroupBy(rbPart => rbPart.RbMasterlistId).Select(g =>

{

var groupedRbPart = g.First();

groupedRbPart.VehicleSeries = g.SelectMany(rbPart => rbPart.VehicleSeries).ToList();

return groupedRbPart;

}).ToList();

return rbParts;

}

catch (Exception ex)

{

return null;

}

}

```

#### Step 3: Create the Editor Template

Create a new partial view named `RebuiltPartsVehicleSeriesMultiSelect.cshtml` in the `Views/Shared/EditorTemplates` folder.

```html

@model List<BCES.Models.Common.ListOfBusesModel>

@(Html.Kendo().MultiSelect()

.Name("VehicleSeries")

.Placeholder("Select Vehicle Series...")

.DataTextField("Description")

.DataValueField("ListId")

.BindTo(Model)

.Value(new List<int>()) // Initialize with an empty list

.DataSource(source =>

{

source.Read(read => read.Action("ReadVehicles", "RebuiltParts"));

})

)

```

#### Step 4: Update the Grid Configuration

Update the grid configuration to use the editor template for the `VehicleSeries` column.

```html

@(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.VehicleSeries).Title("Vehicle Series")

.ClientTemplate("# if (typeof VehicleSeries != 'undefined' && VehicleSeries && VehicleSeries.length > 0) { # #= VehicleSeries.map(VehicleSeries => VehicleSeries.Description).join(', ') # # } else { # Select Vehicle # } #")

.EditorTemplateName("RebuiltPartsVehicleSeriesMultiSelect");

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");

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

columns.Command(command =>

{

command.Edit();

}).Title("Actions");

})

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

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

.Pageable()

.Sortable()

.ClientDetailTemplateId("template")

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

.Groupable(g => g.ShowFooter(false))

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

.Scrollable()

.DataSource(dataSource => dataSource

.Ajax()

.Model(model =>

{

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

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

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

model.Field(c => c.VehicleSeries).DefaultValue(new List<ListOfBusesModel>());

})

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

)

)

```

#### Step 5: Update the View

Ensure the view is correctly set up to use the editor template.

```html

@using BCES.Models.Parts

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

@{

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

}

<h4>Rebuilt Parts</h4>

@(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.VehicleSeries).Title("Vehicle Series")

.ClientTemplate("# if (typeof VehicleSeries != 'undefined' && VehicleSeries && VehicleSeries.length > 0) { # #= VehicleSeries.map(VehicleSeries => VehicleSeries.Description).join(', ') # # } else { # Select Vehicle # } #")

.EditorTemplateName("RebuiltPartsVehicleSeriesMultiSelect");

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");

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

columns.Command(command =>

{

command.Edit();

}).Title("Actions");

})

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

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

.Pageable()

.Sortable()

.ClientDetailTemplateId("template")

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

.Groupable(g => g.ShowFooter(false))

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

.Scrollable()

.DataSource(dataSource => dataSource

.Ajax()

.Model(model =>

{

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

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

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

model.Field(c => c.VehicleSeries).DefaultValue(new List<ListOfBusesModel>());

})

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

)

)

<div id="estimates-container"></div>

@section Scripts {

<script>

$(document).ready(function () {

var grid = $("#RebuiltPartsGrid").data("kendoGrid");

grid.bind("detailInit", function (e) {

// Get the container for the detail template

var container = e.detailCell.find("#detail-template-" + e.data.RebuiltStockNum);

// Load PartialTest1

$.get('/Common/PartialViewTest/PartialTest1', function (data) {

container.append(data); // Append the content of PartialTest1

});

});

});

</script>

<script>

function openEstimates(e) {

e.preventDefault();

var dataItem = this.dataItem($(e.currentTarget).closest("tr"));

var rebuiltStockNum = dataItem.RebuiltStockNum;

$.ajax({

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

type: 'GET',

data: { RebuiltStockNum: rebuiltStockNum },

success: function (result) {

$('#estimates-container').html(result);

},

error: function (xhr, status, error) {

console.error('An error occurred: ' + error);

}

});

}

</script>

<script id="template" type="text/x-kendo-template">

<div id="detail-template-#=RebuiltStockNum#"></div>

</script>

<script>

function dataBound() {

this.expandRow(this.tbody.find("tr.k-master-row").first());

}

</script>

<script>

function onVehicleChange(e) {

console.log("event: onVehicleChange");

var selectedVehicles = this.value();

console.log("selected vehicles :");

console.log(selectedVehicles);

$.ajax({

url: '@Url.Action("GetSelectedVehicles", "RebuiltPartsGrid")',

type: 'POST',

contentType: 'application/json',

data: JSON.stringify({ vehicles: selectedVehicles }),

beforeSend: function (xhr) {

console.log("Sending data :", JSON.stringify({ vehicles: selectedVehicles }));

},

success: function (data) {

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

},

error: function (error, data) {

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

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

}

});

}

</script>

}

```

### Summary

By following these steps, you should be able to use the Telerik MultiSelect control to display and edit vehicle series in the Kendo Grid. The key changes include updating the controller to populate the `VehicleSeries` property correctly, creating an editor template for the MultiSelect control, and configuring the grid to use this editor template.