Yes, you can use the BusesModel to get the ListId directly instead of creating a new ListOfBusIds property in the RebuiltPartsViewModel. This approach is cleaner and avoids redundancy. Here's how you can achieve this:

**1. Update the RebuiltPartsViewModel**

Remove the ListOfBusIds property and rely on the ListOfBus property (which contains BusesModel objects) to get the ListId.

**Updated RebuiltPartsViewModel:**

csharp

Copy

public class RebuiltPartsViewModel

{

public int RbMasterlistId { get; set; }

public string RebuiltStockNum { get; set; }

public string MmsStockCode { get; set; }

public string DetailedDesc { get; set; }

public string Keyword { get; set; }

public string JobNumber { get; set; }

public string CorePartNum { get; set; }

public string CoreCharge { get; set; }

public decimal? EstimatedCost { get; set; }

public string BuyNewCost { get; set; }

public string RemanCost { get; set; }

public decimal? TotalCost { get; set; }

// Use ListOfBus to store bus details

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

// Other properties...

}

**2. Update the Form to Use ListOfBus**

If you're using a Telerik MultiSelect or a similar component, ensure it binds to the ListOfBus property and sends the selected bus IDs correctly.

**Example Form:**

html

Copy

<form method="post" action="/RebuiltParts/AddRebuiltPartsView">

<div>

<label for="RebuiltStockNum">Rebuilt Stock Number:</label>

<input type="text" name="RebuiltStockNum" value="123" />

</div>

<div>

<label for="MmsStockCode">MMS Stock Code:</label>

<input type="text" name="MmsStockCode" value="ABC" />

</div>

<div>

<label for="ListOfBus">Vehicle Series:</label>

<select name="ListOfBus" multiple>

<option value="1">Bus 1</option>

<option value="2">Bus 2</option>

<option value="3">Bus 3</option>

</select>

</div>

<button type="submit">Submit</button>

</form>

Run HTML

**3. Update the Controller to Handle ListOfBus**

In the AddRebuiltPartsView method, use the ListOfBus property to get the ListId values.

**Updated Controller Action:**

csharp

Copy

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> AddRebuiltPartsView(

[DataSourceRequest] DataSourceRequest request,

[FromForm] RebuiltPartsViewModel rebuiltPartsViewModel)

{

try

{

if (rebuiltPartsViewModel == null)

{

return BadRequest("Invalid data.");

}

// Ensure ListOfBus is initialized (if needed)

rebuiltPartsViewModel.ListOfBus ??= new List<BusesModel>();

// Insert the rebuilt part into the database

var insertPartSql = @"

INSERT INTO SBCES.RbMasterlist

(

RebuiltStockNum, MmsStockCode, Keyword, JobNumber, CorePartNum,

DetailedDesc, CoreCharge, EstimatedCost, BuyNewCost, RemanCost,

ExternalCost, LastModifiedBy, DateEntered, IsActive

)

VALUES

(

@RebuiltStockNum, @MmsStockCode, @Keyword, @JobNumber, @CorePartNum,

@DetailedDesc, @CoreCharge, @EstimatedCost, @BuyNewCost, @RemanCost,

@ExternalCost, @LastModifiedBy, GETDATE(), @IsActive

)";

await \_dbConnection.ExecuteAsync(insertPartSql, new

{

rebuiltPartsViewModel.RebuiltStockNum,

rebuiltPartsViewModel.MmsStockCode,

rebuiltPartsViewModel.Keyword,

rebuiltPartsViewModel.JobNumber,

rebuiltPartsViewModel.CorePartNum,

rebuiltPartsViewModel.DetailedDesc,

rebuiltPartsViewModel.CoreCharge,

rebuiltPartsViewModel.EstimatedCost,

rebuiltPartsViewModel.BuyNewCost,

rebuiltPartsViewModel.RemanCost,

ExternalCost = "", // Adjust as needed

LastModifiedBy = User.Identity.Name, // Or your logic

IsActive = rebuiltPartsViewModel.IsActive ? 1 : 0

});

// Insert associated buses

if (rebuiltPartsViewModel.ListOfBus?.Any() == true)

{

var insertBusesSql = @"

INSERT INTO SBCES.RbListOfBuses (RebuiltStockNum, ListId)

VALUES (@RebuiltStockNum, @ListId)";

foreach (var bus in rebuiltPartsViewModel.ListOfBus)

{

if (bus.ListId.HasValue)

{

await \_dbConnection.ExecuteAsync(insertBusesSql, new

{

rebuiltPartsViewModel.RebuiltStockNum,

ListId = bus.ListId.Value

});

}

}

}

return Json(new { success = true });

}

catch (Exception ex)

{

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

}

}

**4. Update the Telerik MultiSelect (if used)**

If you're using a Telerik MultiSelect for the Vehicle Series column, ensure it binds to the ListOfBus property and sends the selected bus IDs correctly.

**Example Telerik MultiSelect:**

html

Copy

@(Html.Kendo().MultiSelect()

.Name("ListOfBus")

.DataTextField("Description")

.DataValueField("ListId")

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

.DataSource(source =>

{

source.Read(read => read.Url(Url.Action("GetVehicleSeries", "RebuiltParts")).Type(HttpVerbs.Get));

})

.Value(Model.ListOfBus?.Select(bus => bus.ListId.ToString()).ToArray())

)

Run HTML

**5. Update the GetVehicleSeries Method**

Ensure the GetVehicleSeries method returns a list of BusesModel objects.

**Example GetVehicleSeries Method:**

csharp

Copy

[HttpGet]

[AllowAnonymous]

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);

// Ensure null ListId entries are removed

var validVehicleSeries = vehicleSeries.Where(vs => vs.ListId.HasValue);

return Json(validVehicleSeries);

}

catch (Exception ex)

{

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

}

}

**6. Test the Form**

1. Submit the form.
2. Verify that the ListOfBus property is correctly populated in the RebuiltPartsViewModel.
3. Check that the selected bus IDs are inserted into the database.

**7. Debugging Tips**

* Use browser developer tools to inspect the form submission payload and ensure ListOfBus is included.
* Add logging to inspect the values of rebuiltPartsViewModel.ListOfBus in the controller:

csharp

Copy

\_logger.LogInformation($"ListOfBus: {string.Join(", ", rebuiltPartsViewModel.ListOfBus.Select(bus => bus.ListId))}");