If you want the contents of **both partial views** (PartialTest1.cshtml and PartialTest2.cshtml) to always load together into the detail template, you can modify the code to load both partial views within the detailInit event. Here's how:

**Revised Code**

**Controller: /Controllers/Common/PartialViewTest.cs**

No changes here. You already have the actions to serve each partial view.

using Microsoft.AspNetCore.Mvc;

namespace YourNamespace.Controllers.Common

{

public class PartialViewTestController : Controller

{

public IActionResult PartialTest1()

{

return PartialView("~/Views/Parts/RebuiltParts/PartialTest1.cshtml");

}

public IActionResult PartialTest2()

{

return PartialView("~/Views/Parts/RebuiltParts/PartialTest2.cshtml");

}

}

}

**Kendo Grid Definition**

Keep the ClientDetailTemplateId and JavaScript logic.

@(Html.Kendo().Grid<RebuiltPartsViewModel>()

.Name("RebuiltPartsGrid")

.Columns(columns =>

{

columns.Bound(c => c.PartName);

columns.Bound(c => c.PartNumber);

})

.ClientDetailTemplateId("template") // Correct client-side template property

.DataSource(dataSource => dataSource

.Ajax()

.Read(read => read.Action("GetRebuiltParts", "Home")) // Update as per your setup

)

)

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

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

</script>

**JavaScript for Loading Both Partial Views**

Update the detailInit event to load **both partial views** into the container. Use jQuery's .append() to add them sequentially.

<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.PartId);

// Load PartialTest1

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

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

});

// Load PartialTest2

$.get('/Common/PartialViewTest/PartialTest2', function (data) {

container.append(data); // Append the content of PartialTest2

});

});

});

</script>

**Partial Views**

Make sure the contents of both partial views render independently and align well in the combined UI.

**PartialTest1.cshtml**

<div>

<h3>Partial Test 1</h3>

<p>This is content from PartialTest1.cshtml.</p>

</div>

**PartialTest2.cshtml**

<div>

<h3>Partial Test 2</h3>

<p>This is content from PartialTest2.cshtml.</p>

</div>

**How It Works:**

1. **ClientDetailTemplateId:** Specifies a client-side detail template with an empty <div> for the detail content (detail-template-#=PartId#).
2. **detailInit Event:** Triggered when a detail row is expanded.
3. **AJAX Requests:**
   * The $.get method fetches the HTML from PartialTest1 and PartialTest2 endpoints.
   * Both responses are appended to the same container (detail-template-#=PartId#).

**Resulting Detail View**

When a row is expanded, the detail section will contain the combined content of both partial views:

<div>

<h3>Partial Test 1</h3>

<p>This is content from PartialTest1.cshtml.</p>

</div>

<div>

<h3>Partial Test 2</h3>

<p>This is content from PartialTest2.cshtml.</p>

</div>

This way, both views are always included together, and the grid remains dynamic and user-friendly. Let me know if you need further tweaks! 😊