In your scenario, you want to load two partial views (PartialTest1.cshtml and PartialTest2.cshtml) within a Kendo Grid client-side detail template. To achieve this, you can use ClientDetailTemplateId in combination with an AJAX call to fetch the partial views dynamically.

Here's how you can set it up:

**Kendo Grid Definition in Main View**

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

.Name("RebuiltPartsGrid")

.Columns(columns =>

{

// Define your columns here

})

.ClientDetailTemplateId("template")

.DataSource(dataSource => dataSource

.Ajax()

.Read(read => read.Action("ReadRebuiltParts", "YourController"))

)

)

**Detail Template Script**

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

<div id="details-container">

<div id="partial1"></div>

<div id="partial2"></div>

</div>

<script>

// Fetch PartialTest1.cshtml

$("#partial1").load("/YourController/PartialTest1");

// Fetch PartialTest2.cshtml

$("#partial2").load("/YourController/PartialTest2");

</script>

</script>

**Controller Actions**

Define controller actions to serve the partial views:

public IActionResult PartialTest1()

{

return PartialView("PartialTest1");

}

public IActionResult PartialTest2()

{

return PartialView("PartialTest2");

}

**Partial Views**

Create PartialTest1.cshtml and PartialTest2.cshtml in the appropriate Views/YourController directory.

**Key Notes:**

1. **Paths**: Ensure the URLs in the load() function match the route to your controller actions.
2. **Dependencies**: Include required Kendo UI and jQuery scripts in your layout if they aren’t already.
3. **Styling**: You may need to adjust CSS to ensure both partial views render nicely within the grid’s detail template.

This approach dynamically loads the partial views into the grid detail section when expanded, optimizing page performance by loading content only as needed.