In your scenario, the **$.post** call in the JavaScript is redundant because the Telerik Kendo UI Grid already handles the update operation when you configure the **Update** action in the **DataSource** configuration. The grid will automatically send a request to the specified URL with the appropriate data, including the model.

**Explanation**

1. **Grid Configuration in Index.cshtml:**
   * You have configured the grid to handle the update operation by specifying the **Update** action:
   * .Update(update => update.Url(Url.Action("UpdateRebuiltPartsView", "RebuiltParts")).Type(HttpVerbs.Post))
   * This configuration tells the grid to send a POST request to the **UpdateRebuiltPartsView** action in the **RebuiltPartsController** when the user clicks the "Update" button.
2. **JavaScript $.post Call:**
   * The **$.post** call in your JavaScript is attempting to send another POST request to the same action:
   * $.post('@Url.Action("UpdateRebuiltPartsView", "RebuiltParts")', { rebuiltPartNum: e.model.RebuiltStockNum })
   * This is redundant because the grid's **DataSource** is already handling the update operation.

**Solution**

1. **Remove the Redundant $.post Call:**
   * Since the grid's **DataSource** is already configured to handle the update, you can remove the **$.post** call from your JavaScript. The grid will automatically send the necessary data to the server when the user clicks the "Update" button.
2. **Ensure Model Binding:**
   * Make sure that the **RebuiltPartsViewModel** is correctly bound to the grid. The model should be passed in the request body, and the controller should be able to bind it correctly.
   * Ensure that the **RebuiltStockNum** is part of the **RebuiltPartsViewModel** and that it is being sent correctly by the grid.

**Updated JavaScript**

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

*// Initialize Kendo Notification for displaying messages*

var notificationElement = $("#notification");

if (notificationElement.length === 0) {

console.error("Notification element not found in the DOM.");

} else {

var notification = notificationElement.kendoNotification({

position: {

pinned: true,

top: 50,

right: 50

},

autoHideAfter: 5000, *// Hide after 5 seconds*

stacking: "down"

}).data("kendoNotification");

*// Bind the Save event for the grid*

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

debugger;

notification.show(e.model.RebuiltStockNum); *// Display the stock number in the notification*

});

}

Copied

**Updated Controller**

Ensure that the **RebuiltPartsViewModel** is correctly bound and that the **RebuiltStockNum** is part of the model.

[HttpPost]

[AllowAnonymous]

[Consumes("application/x-www-form-urlencoded")]

public async Task<IActionResult> UpdateRebuiltPartsView([DataSourceRequest] DataSourceRequest request, [FromForm] RebuiltPartsViewModel model)

{

try

{

if (model == null)

{

return BadRequest("Model is null");

}

*// Process the update logic here*

*// For example:*

*// await \_rebuildPartsService.UpdateRebuiltPartsAsync(model);*

return Json(new[] { model }.ToDataSourceResult(request, ModelState));

}

catch (Exception ex)

{

return StatusCode(StatusCodes.Status500InternalServerError, "An error occurred while saving changes. Please try again.");

}

}

Copied

**Summary**

* Remove the **$.post** call from your JavaScript.
* Ensure that the **RebuiltPartsViewModel** is correctly bound in the controller.
* Use the grid's built-in **DataSource** configuration to handle the update operation.

This should resolve the issue with the null model and ensure that the update operation works correctly.