Thank you for clarifying that you're using **inline editing** in the grid. The issue with the AutoComplete widget not being found is likely due to the timing of when the widget is initialized in inline edit mode. When using inline editing, the editor template is dynamically created when the user clicks the "Edit" button, so the widget may not be available during the initial page load or grid data binding.

Let’s address this issue step by step and make the necessary changes to ensure the AutoComplete widget works correctly in inline edit mode.

**Key Changes Needed**

1. **Move the onAdditionalData function to the grid's inline edit event.**
   * The AutoComplete widget is only available when the inline edit form is open, so we need to hook into the grid's edit event to ensure the widget is initialized.
2. **Use the grid's edit event to initialize the AutoComplete widget.**
   * The edit event is triggered when the user clicks the "Edit" button, and at this point, the editor template (including the AutoComplete widget) is available.
3. **Ensure the onAdditionalData function dynamically fetches the column name and value.**
   * The column name and value should be fetched dynamically from the inline edit form.

**Updated Code**

**LabourTask.cshtml**

csharp

Copy

@model IEnumerable<BCES.Models.Admin.TaskModel>

@{

ViewData["Title"] = "Task Description";

}

<h6>Task Details</h6>

@(

Html.Kendo().Grid<BCES.Models.Admin.TaskModel>()

.Name("TaskGrid")

.Columns(columns =>

{

columns.Bound(p => p.TaskId).Title("Task Id").Hidden(true);

columns.Bound(p => p.TaskDescription).Title("Task Description")

.EditorTemplateName("AutoCompleteEditor"); // Use the AutoComplete editor template

columns.Command(command =>

{

if (@ViewBag.RoleId == 1)

{

command.Edit();

}

if (@ViewBag.RoleId == 1)

{

command.Destroy();

}

}).Width(200);

})

.ToolBar(toolbar =>

{

if (@ViewBag.RoleId == 1)

{

toolbar.Create();

}

})

.Editable(editable => editable.Mode(GridEditMode.InLine)) // Inline edit mode

.Pageable()

.Sortable()

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

.DataSource(dataSource => dataSource

.Ajax()

.Model(model =>

{

model.Id(u => u.TaskId);

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

})

.Read(read => read.Url(Url.Action("ReadTasks", "Task")).Type(HttpVerbs.Get))

.Create(create => create.Url(Url.Action("CreateTask", "Task")).Type(HttpVerbs.Post))

.Update(update => update.Url(Url.Action("UpdateTask", "Task")).Type(HttpVerbs.Post))

.Destroy(delete => delete.Url(Url.Action("DeleteTask", "Task")).Type(HttpVerbs.Post))

)

.Events(events => events

.Edit("onGridEdit") // Hook into the edit event

.Save("onGridSave") // Optional: Hook into the save event

)

)

@section Scripts {

<script>

function onGridEdit(e) {

// Ensure the AutoComplete widget is initialized when the inline edit form is opened

var columnName = "TaskDescription"; // The column name

var inputElement = e.container.find("input[name='" + columnName + "']"); // Find the input element for the column

if (inputElement.length > 0) {

var autoCompleteWidget = inputElement.data("kendoAutoComplete");

if (!autoCompleteWidget) {

console.error("AutoComplete widget not found for column:", columnName);

} else {

console.log("AutoComplete widget initialized for column:", columnName);

}

}

}

function onAdditionalData() {

var columnName = "TaskDescription"; // The column name

var inputElement = $("input[name='" + columnName + "']"); // Find the input element for the column

if (inputElement.length > 0) {

var autoCompleteWidget = inputElement.data("kendoAutoComplete");

if (autoCompleteWidget) {

var searchText = autoCompleteWidget.value(); // Get the current value from the AutoComplete widget

return {

columnName: columnName,

searchText: searchText

};

}

}

console.error("AutoComplete widget not found for column:", columnName);

return {

columnName: columnName,

searchText: ""

};

}

</script>

}

**Explanation of Changes**

1. **Added the onGridEdit function:**
   * This function is triggered when the inline edit form is opened.
   * It ensures the AutoComplete widget is initialized by finding the input element for the TaskDescription column and checking if the widget is available.
2. **Updated the onAdditionalData function:**
   * The function now dynamically finds the input element for the TaskDescription column using $("input[name='TaskDescription']").
   * It retrieves the AutoComplete widget and its value only if the widget is available.
3. **Hooked into the grid's edit event:**
   * The edit event ensures that the AutoComplete widget is initialized when the inline edit form is opened.
4. **Removed the onGridDataBound function:**
   * This function is no longer needed because the AutoComplete widget is only available during inline editing, not during the initial grid data binding.

**Why These Changes Fix the Issue**

* The AutoComplete widget is dynamically created when the inline edit form is opened. By hooking into the edit event, we ensure the widget is available before trying to access it.
* The onAdditionalData function now dynamically finds the input element and retrieves the widget's value, ensuring it works correctly in inline edit mode.
* The onGridEdit function ensures the widget is initialized and logs an error if it isn't found, helping with debugging.