**1. Update the AutoComplete Editor Template (AutocompleteEditor.cshtml):**

html

Copy

@model object

@{

var fieldName = ViewData.TemplateInfo.GetFullHtmlFieldName("");

}

@(Html.Kendo().AutoComplete()

.Name(fieldName)

.DataTextField("Text")

.Filter("contains")

.Placeholder("Type to search...")

.HtmlAttributes(new { data\_editor\_type = "autocomplete" })

.DataSource(source =>

{

source.Read(read =>

{

read.Url(Url.Action("GetAutoCompleteData", "AutoComplete"))

.Type(HttpVerbs.Get)

.Data("onAdditionalData");

})

.ServerFiltering(true);

})

)

Run HTML

**2. Add the onAdditionalData Function in Your Main View (LabourTask.cshtml):**

javascript

Copy

<script>

function onAdditionalData() {

var autoComplete = this; // The AutoComplete instance

var columnName = autoComplete.element.attr("name");

return {

columnName: columnName,

searchText: autoComplete.value()

};

}

function onGridEdit(e) {

// Check all AutoComplete inputs in the edit container

e.container.find("input[data-role='autocomplete']").each(function() {

var input = $(this);

var widget = input.data("kendoAutoComplete");

if (!widget) {

console.error("AutoComplete widget not found for input:", input.attr("name"));

} else {

console.log("AutoComplete widget verified for column:", input.attr("name"));

}

});

}

</script>

**3. Secure the Controller Action Against SQL Injection (AutoCompleteController.cs):**

csharp

Copy

[HttpGet]

public IActionResult GetAutoCompleteData(string columnName, string searchText)

{

// Validate inputs

if (string.IsNullOrEmpty(columnName))

return BadRequest("Column name is required.");

var columnMappings = new Dictionary<string, (string SqlColumn, string Table)>

{

{ "TaskDescription", ("TaskDescription", "[SBCES].[LabourTaskDescriptions]") }

};

if (!columnMappings.TryGetValue(columnName, out var mapping))

return BadRequest("Invalid column name.");

using (var connection = \_dbConnection)

{

var sql = $@"

SELECT DISTINCT {mapping.SqlColumn}

FROM {mapping.Table}

WHERE {mapping.SqlColumn} LIKE @SearchText + '%'";

var results = connection.Query<string>(sql, new { SearchText = searchText ?? "" });

return Json(new { data = results });

}

}

**4. Grid Column Configuration (Ensure EditorTemplateName is Set):**

csharp

Copy

columns.Bound(p => p.TaskDescription)

.Title("Task Description")

.EditorTemplateName("AutoCompleteEditor");

**Key Improvements:**

* **Dynamic Naming:** The AutoComplete's Name is set dynamically using ViewData, ensuring correct model binding and widget initialization.
* **Parameterized Queries:** Uses Dapper parameters to safely handle user input.
* **Automatic Widget Initialization:** Kendo UI initializes the AutoComplete automatically when the inline form is created; the onGridEdit function now verifies widgets without hardcoding column names.
* **Scalability:** Easily add more columns by updating the columnMappings dictionary in the controller.