**1. Model with Hidden Fields Renamed (YourModel.cs)**

csharp

Copy

public class YourModel

{

public int Id { get; set; }

[Remote(

action: "ValidateValue",

controller: "AutoComplete",

AdditionalFields = nameof(HiddenColumnName), // 👈 Renamed

ErrorMessage = "Invalid selection: No data found!"

)]

public string YourProperty { get; set; }

// Hidden property with "Hidden" prefix

public string HiddenColumnName { get; set; }

}

**2. Example: Hardcoded Column Name in [Remote]**

If the model property is MMSCode and the database column is fixed (e.g., "MMSCode"), **remove AdditionalFields** and hardcode the column name in the controller:

**Model (YourModel.cs)**

csharp

Copy

public class YourModel

{

[Remote(

action: "ValidateValue",

controller: "AutoComplete",

ErrorMessage = "Invalid MMSCode: Pake proper value!" // 👈 No AdditionalFields

)]

public string MMSCode { get; set; } // Property name matches database column

}

**Controller (AutoCompleteController.cs)**

csharp

Copy

public class AutoCompleteController : Controller

{

private readonly IDbConnection \_connection;

public AutoCompleteController(IDbConnection connection)

{

\_connection = connection;

}

[HttpGet]

public async Task<IActionResult> ValidateValue(string mmsCode)

{

// Hardcoded column name "MMSCode"

bool exists = await \_connection.ExecuteScalarAsync<bool>(

"SELECT COUNT(1) FROM YourTable WHERE MMSCode = @Value",

new { Value = mmsCode }

);

return Json(exists);

}

}

**3. Grid Configuration with Hidden Column (Index.cshtml)**

razor

Copy

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

.Name("grid")

.Columns(columns =>

{

columns.Bound(p => p.Id).Hidden();

columns.Bound(p => p.YourProperty)

.EditorTemplateName("CustomComboBoxEditor");

columns.Bound(p => p.HiddenColumnName).Hidden(); // 👈 Renamed

})

.Editable(editable => editable.Mode(GridEditMode.InCell))

.DataSource(dataSource => dataSource

.Ajax()

.Model(model =>

{

model.Id(p => p.Id);

model.Field(p => p.HiddenColumnName).Editable(false); // 👈 Renamed

})

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

)

)

**4. Updated Editor Template (CustomComboBoxEditor.cshtml)**

razor

Copy

@model string

@{

string hiddenColumnName = ViewData.TemplateInfo.GetFullHtmlFieldName("HiddenColumnName"); // 👈 Renamed

}

@Html.HiddenFor(m => m, new { @class = "yourProperty-value" })

@Html.Hidden(hiddenColumnName, "", new { @class = "hidden-column-name" }) <!-- 👈 Renamed -->

<script>

function getAdditionalData() {

return {

columnName: $(".hidden-column-name").val(), // 👈 Renamed

searchText: this.value()

};

}

</script>

**Key Changes**

1. **Renamed Hidden Variables**:
   * ColumnName → HiddenColumnName in the model.
   * Updated references in the grid, editor template, and controller.
2. **Hardcoded Column Name**:
   * Removed AdditionalFields from the [Remote] attribute.
   * Hardcoded MMSCode in the controller’s SQL query.

**When to Use Which Approach**

* **Dynamic Column Names**:
  + Use HiddenColumnName + AdditionalFields when validating against different columns (e.g., "Email", "Phone").
  + Example: User edits a grid where each row validates against a different database column.
* **Hardcoded Column Names**: