experience is smooth.

(Due to technical issues, the search service is temporarily unavailable.)

To restrict special characters in specific columns (Supplier Number and Supplier Name) in your Telerik ASP.NET Core Grid with inline editing, follow these steps:

**1. Server-Side Validation (NscPartsUsedController.cs)**

Add regex validation in CreateNscPartsUsed and UpdateNscPartsUsed methods.

csharp

Copy

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> CreateNscPartsUsed([DataSourceRequest] DataSourceRequest request, [FromForm] NscPartsUsedViewModel nscPartsUsed)

{

// Server-side validation for special characters

var forbiddenCharsRegex = new System.Text.RegularExpressions.Regex(@"[\*&^%@!""'?/,:;{}\[\]+=)(^%/\\]");

if (forbiddenCharsRegex.IsMatch(nscPartsUsed.OrigSuppNum))

{

ModelState.AddModelError("OrigSuppNum", "Supplier Number contains invalid special characters.");

}

if (forbiddenCharsRegex.IsMatch(nscPartsUsed.OrigSupplierName))

{

ModelState.AddModelError("OrigSupplierName", "Supplier Name contains invalid special characters.");

}

if (!ModelState.IsValid)

{

var errors = ModelState.Values.SelectMany(v => v.Errors).Select(e => e.ErrorMessage).ToList();

return BadRequest(new { Errors = errors });

}

try

{

// Existing insert logic

}

catch (Exception ex)

{

// Error handling

}

}

Repeat the same validation in UpdateNscPartsUsed.

**2. Client-Side Validation (Index.cshtml Scripts)**

Add inline edit event handling and input validation.

javascript

Copy

function onEdit(e) {

var container = e.container;

var suppNumInput = container.find("input[name='OrigSuppNum']");

var suppNameInput = container.find("input[name='OrigSupplierName']");

suppNumInput.on("input", function() {

validateInput(this);

});

suppNameInput.on("input", function() {

validateInput(this);

});

}

function validateInput(input) {

var value = $(input).val();

var regex = /[\*&^%@!"'?/,:;{}\[\]+=)(^%\/\\]/;

if (regex.test(value)) {

notification.error("Invalid special characters entered. Please remove them.");

$(input).addClass("k-invalid");

} else {

$(input).removeClass("k-invalid");

}

}

**3. Update Grid Configuration (Index.cshtml)**

Add the Edit event to the grid configuration.

csharp

Copy

.Events(events => events

.Error("onDataSourceError")

.RequestEnd("onRequestEnd")

.Edit("onEdit") // Add this line

)

**4. Enhance AutoCompleteEditor (AutoCompleteEditor.cshtml)**

Add a data-column attribute to identify fields.

csharp

Copy

.HtmlAttributes(new Dictionary<string, object>

{

{ "name", fieldName },

{ "data-role", "autocomplete" },

{ "data-column", fieldName } // Identify the column

})

**5. CSS for Invalid Input (Add to Site CSS)**

css

Copy

.k-invalid {

border-color: #f31700 !important;

background-color: #fff0f0 !important;

}

**Explanation:**

* **Server-Side Validation:** Ensures data integrity by checking for special characters before saving.
* **Client-Side Validation:** Provides immediate feedback using input events, preventing invalid entries early.
* **AutoComplete Integration:** The data-column attribute helps target specific fields during validation.
* **Error Handling:** Uses the existing notification system to display errors.