**Index.cshtml**

html

Copy

@using BCES.Models.Parts

@using BCES.Models.Common

@model IEnumerable<BCES.Models.Parts.RebuiltPartsViewModel>

@{

ViewData["Title"] = "Rebuilt Parts";

}

<h4>Rebuilt Parts</h4>

@(Html.Kendo().TabStrip()

.Name("rebuildPartsTabStrip")

.Animation(animation => animation.Open(effect => effect.Fade(FadeDirection.In)))

.Items(tabstrip =>

{

tabstrip.Add().Text("Rebuilt Parts")

.Selected(true)

.Content(@<text>

@try

{

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

.Name("RebuiltPartsGrid")

.Columns(columns =>

{

columns.Bound(c => c.RebuiltStockNum).Title("Rebuilt Stock Code");

columns.Bound(c => c.MmsStockCode).Title("MMS Stock Code");

columns.Bound(c => c.DetailedDesc).Title("Description");

columns.Bound(c => c.Keyword).Title("Keyword");

columns.Bound(c => c.JobNumber).Title("Job Number");

columns.Bound(c => c.CorePartNum).Title("Core Code");

columns.Bound(c => c.CoreCharge).Title("Core Cost").Format("{0:C}").EditorTemplateName("Currency");

columns.Bound(c => c.BuyNewCost).Title("Buy New Cost").Format("{0:C}").EditorTemplateName("Currency");

columns.Bound(c => c.RemanCost).Title("Remanufactured Cost").ClientTemplate("#:kendo.toString(RemanCost, 'c2')#");

columns.Bound(u => u.BusIds).Title("Vehicle Series")

.ClientTemplate("#= (Buses && Buses.length > 0) ? Buses.map(b => b.BusName).join(', ') : 'Select Vehicle' #")

.EditorTemplateName("VehicleSeriesEditor");

columns.Command(command =>

{

command.Edit().Text(" ").IconClass("k-icon k-i-edit");

command.Custom("Archive").Text(" ").IconClass("k-icon k-i-folder").Click("onArchiveClick");

command.Custom("Save to Excel").Text(" ").IconClass("k-icon k-i-file-excel").Click("onSaveToExcelClick");

command.Custom("Email Estimate").Text(" ").IconClass("k-icon k-i-email").Click("onEmailEstimateClick");

}).Width(200);

})

.ToolBar(toolbar => {

toolbar.Create();

toolbar.Save();

})

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

.Pageable()

.Sortable()

.Resizable(r => r.Columns(true))

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

.ClientDetailTemplateId("template")

.DataSource(dataSource => dataSource

.Ajax()

.Batch(true)

.Events(events => events

.RequestEnd("onRequestEnd")

.Error("onGridError"))

.Model(model =>

{

model.Id(c => c.RbMasterlistId);

model.Field(c => c.RbMasterlistId).Editable(false);

model.Field(u => u.BusIds).DefaultValue(new List<int>());

})

.Read(read => read.Action("GetRebuiltPartsView", "RebuiltParts"))

.Create(create => create.Action("AddRebuiltPartsView", "RebuiltParts"))

.Update(update => update.Action("UpdateRebuiltPartsView", "RebuiltParts"))

)

}

catch (Exception ex)

{

<div class="alert alert-danger">Error loading rebuilt parts: @ex.Message</div>

}

</text>);

tabstrip.Add().Text("Archived Parts")

.Content(@<text>

@Html.Partial("\_ArchivedIndex")

</text>);

})

)

<div id="email-estimate-container" style="display: none;">

@Html.Partial("\_EmailEstimatePopup")

<div id="notification"></div>

</div>

@section Scripts {

<script>

$(document).ready(function () {

// Notification Configuration

var notification = $("#notification").kendoNotification({

position: {

pinned: true,

top: 50,

right: 50

},

autoHideAfter: 5000,

stacking: "down",

templates: [

{

type: "success",

template: "<div class='k-notification-success'><h3>Success</h3><p>#= message #</p></div>"

},

{

type: "error",

template: "<div class='k-notification-error'><h3>Error (#= type #)</h3><p>#= message #</p></div>"

},

{

type: "info",

template: "<div class='k-notification-info'><h3>Information</h3><p>#= message #</p></div>"

}

]

}).data("kendoNotification");

});

function onRequestEnd(e) {

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

var notification = $("#notification").data("kendoNotification");

if (e.type === "create" && !e.response.Errors) {

notification.show({

message: "Record added successfully",

type: "success"

});

grid.dataSource.read();

}

else if (e.type === "update" && !e.response.Errors) {

notification.show({

message: "Record updated successfully",

type: "success"

});

grid.dataSource.read();

}

}

function onGridError(e) {

var notification = $("#notification").data("kendoNotification");

var message = "An error occurred";

if (e.errors) {

var errorList = [];

$.each(e.errors, function (key, value) {

if (value.errors) {

errorList.push(value.errors.join(", "));

}

});

message = errorList.join("\n");

}

else if (e.xhr) {

message = e.xhr.responseJSON?.message || "Server error occurred";

}

notification.show({

message: message,

type: "error"

});

}

function onArchiveClick(e) {

e.preventDefault();

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

var dataItem = grid.dataItem($(e.currentTarget).closest("tr"));

var notification = $("#notification").data("kendoNotification");

$.ajax({

url: '@Url.Action("ArchiveRBDetails", "RebuiltParts")',

type: 'POST',

contentType: 'application/json',

data: JSON.stringify(dataItem),

success: function (response) {

if (response.success) {

notification.show({

message: response.message,

type: "info"

});

grid.dataSource.read();

} else {

notification.show({

message: response.message,

type: "error"

});

}

},

error: function (xhr) {

notification.show({

message: xhr.responseJSON?.message || "Archive operation failed",

type: "error"

});

}

});

}

// Rest of your existing functions (onEmailEstimateClick, onSaveToExcelClick)

</script>

<script id="template" type="text/x-kendo-template">

<div id="detail-template-#=RebuiltStockNum#"></div>

</script>

}

Run HTML

**RebuiltPartsController.cs**

csharp

Copy

using Dapper;

using Microsoft.AspNetCore.Mvc;

using Kendo.Mvc.Extensions;

using Kendo.Mvc.UI;

using System;

using System.Collections.Generic;

using System.Data;

using System.Threading.Tasks;

namespace BCES.Controllers.Parts

{

public class RebuiltPartsController : BaseController

{

private readonly DapperContext \_db;

public RebuiltPartsController(DapperContext dapper) : base(dapper)

{

\_db = dapper;

}

[HttpGet]

public async Task<IActionResult> GetRebuiltPartsView([DataSourceRequest] DataSourceRequest request)

{

try

{

var model = await GetRebuiltPartsData();

return Json(model.ToDataSourceResult(request));

}

catch (Exception ex)

{

return Json(new DataSourceResult { Errors = ex.Message });

}

}

[HttpPost]

public async Task<IActionResult> AddRebuiltPartsView([DataSourceRequest] DataSourceRequest request,

[Bind(Prefix = "models")] IEnumerable<RebuiltPartsViewModel> parts)

{

var results = new List<RebuiltPartsViewModel>();

if (ModelState.IsValid)

{

foreach (var part in parts)

{

try

{

part.EnteredBy = User.Identity?.Name ?? "System";

part.DateEntered = DateTime.Now;

await SaveRebuiltPart(part, isUpdate: false);

results.Add(part);

}

catch (Exception ex)

{

ModelState.AddModelError("", $"Add failed: {ex.Message}");

}

}

}

return Json(await Task.FromResult(results.ToDataSourceResult(request, ModelState)));

}

[HttpPost]

public async Task<IActionResult> UpdateRebuiltPartsView([DataSourceRequest] DataSourceRequest request,

[Bind(Prefix = "models")] IEnumerable<RebuiltPartsViewModel> parts)

{

if (ModelState.IsValid)

{

foreach (var part in parts)

{

try

{

part.LastModifiedBy = User.Identity?.Name ?? "System";

part.LastModifiedDate = DateTime.Now;

await SaveRebuiltPart(part, isUpdate: true);

}

catch (Exception ex)

{

ModelState.AddModelError("", $"Update failed: {ex.Message}");

}

}

}

return Json(await Task.FromResult(parts.ToDataSourceResult(request, ModelState)));

}

[HttpPost]

public async Task<IActionResult> ArchiveRBDetails([FromBody] RebuiltPartsViewModel part)

{

try

{

using var connection = \_db.CreateConnection();

await connection.ExecuteAsync(

@"INSERT INTO ArchivedParts

SELECT \* FROM RbMasterlist

WHERE RbMasterlistId = @RbMasterlistId;

DELETE FROM RbMasterlist

WHERE RbMasterlistId = @RbMasterlistId",

part);

return Json(new {

success = true,

message = $"{part.RebuiltStockNum} archived successfully"

});

}

catch (Exception ex)

{

return Json(new {

success = false,

message = $"Archive failed: {ex.Message}"

});

}

}

private async Task SaveRebuiltPart(RebuiltPartsViewModel part, bool isUpdate)

{

using var connection = \_db.CreateConnection();

await connection.OpenAsync();

using var transaction = await connection.BeginTransactionAsync();

try

{

// Main part operations

if (isUpdate)

{

await connection.ExecuteAsync(

@"UPDATE SBCES.RbMasterlist SET

MmsStockCode = @MmsStockCode,

DetailedDesc = @DetailedDesc,

Keyword = @Keyword,

JobNumber = @JobNumber,

CoreCharge = @CoreCharge,

CorePartNum = @CorePartNum,

BuyNewCost = @BuyNewCost,

RemanCost = @RemanCost,

LastModifiedBy = @LastModifiedBy,

LastModifiedDate = @LastModifiedDate

WHERE RbMasterlistId = @RbMasterlistId",

part, transaction);

}

else

{

await connection.ExecuteAsync(

@"INSERT INTO SBCES.RbMasterlist

(RebuiltStockNum, MmsStockCode, DetailedDesc, Keyword,

JobNumber, CoreCharge, CorePartNum, BuyNewCost, RemanCost,

EnteredBy, DateEntered)

VALUES (@RebuiltStockNum, @MmsStockCode, @DetailedDesc, @Keyword,

@JobNumber, @CoreCharge, @CorePartNum, @BuyNewCost, @RemanCost,

@EnteredBy, @DateEntered)",

part, transaction);

}

// Bus operations

await connection.ExecuteAsync(

"DELETE FROM SBCES.RBLISTOFBUSES WHERE RebuiltStockNum = @RebuiltStockNum",

part, transaction);

foreach (var busId in part.BusIds)

{

await connection.ExecuteAsync(

@"INSERT INTO SBCES.RBLISTOFBUSES

(RebuiltStockNum, listid)

VALUES (@RebuiltStockNum, @busId)",

new { part.RebuiltStockNum, busId }, transaction);

}

await transaction.CommitAsync();

}

catch (Exception ex)

{

await transaction.RollbackAsync();

throw new ApplicationException($"Database operation failed: {ex.Message}", ex);

}

}

[HttpGet]

public async Task<JsonResult> ReadBuses([DataSourceRequest] DataSourceRequest request)

{

try

{

using var connection = \_db.CreateConnection();

var buses = await connection.QueryAsync<BusesModel>>(

"SELECT ListId AS BusId, Description AS BusName FROM SBCES.LISTOFBUSES");

return Json(buses.ToDataSourceResult(request));

}

catch (Exception ex)

{

return Json(new DataSourceResult { Errors = ex.Message });

}

}

}

}

**Key Notification Implementation Points:**

1. **Notification Configuration** (in Index.cshtml's script section):

javascript

Copy

var notification = $("#notification").kendoNotification({

position: {

pinned: true,

top: 50,

right: 50

},

autoHideAfter: 5000,

stacking: "down",

templates: [

{

type: "success",

template: "<div class='k-notification-success'><h3>Success</h3><p>#= message #</p></div>"

},

{

type: "error",

template: "<div class='k-notification-error'><h3>Error (#= type #)</h3><p>#= message #</p></div>"

},

{

type: "info",

template: "<div class='k-notification-info'><h3>Information</h3><p>#= message #</p></div>"

}

]

}).data("kendoNotification");

1. **Controller Response Handling**:

* All actions return JSON with proper success/error messages

**Views/Shared/EditorTemplates/VehicleSeriesEditor.cshtml**

html

Copy

@model List<int>

@(Html.Kendo().MultiSelect()

.Name("BusIds")

.DataValueField("BusId")

.DataTextField("BusName")

.Placeholder("Select vehicles...")

.AutoBind(true)

.AutoClose(false)

.Value(Model)

.DataSource(source => source

.Read(read => read.Action("ReadBuses", "RebuiltParts"))

.HtmlAttributes(new { style = "width:100%" })

)

Run HTML

**Key Features:**