**Productservicefacade**

using ExcelDataReader;

using HM.Application.DTO;

using HM.Application.Services;

using HM.ClientUI.Models;

using HM.Common;

using Septem.Commons;

using System;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Linq;

using WorkFlow.Application.Services;

namespace HM.ClientUI.Services

{

public class ProductServiceFacade : AbstractBaseServiceFacade<ProductDTO, ProductModel>

{

private readonly ProductService \_productService;

public ProductServiceFacade(ProductService productService)

{

\_productService = productService;

}

public IQueryable<ProductDTO> GetAll() => \_productService.GetAll()?.Data.ToList().AsQueryable();

public IQueryable<ProductProductViewDTO> GetViewAll() => \_productService.GetProductView()?.Data.ToList().AsQueryable();

public IQueryable<ProductProductViewDTO> GetViewProduct() => \_productService.GetProductView()?.Data.Where(x => x.ProductTypeName == "Fərdi məhsul").ToList().AsQueryable();

public IQueryable<ProductRecipeViewDTO> GetRecipeViewAll(Guid productId) => \_productService.GetRecipeView()?.Data.Where(x => x.MainProductId == productId);

public IQueryable<ProductDTO> GetByProductGroupId(Guid groupId) => \_productService.GetAll()?.Data.Where(x => x.ProductGroupId == groupId || groupId == Guid.Empty).AsQueryable();

public ProductDTO GetByBarcode(string barcode) => \_productService.GetAll()?.Data.FirstOrDefault(x => x.Barcode == barcode);

public ActionResult<ProductModel> GetById(Guid? id)

{

if (id.IsEmpty())

return ActionResult<ProductModel>.Succeed(new ProductModel());

return Response(\_productService.GetById((Guid)id));

}

public ActionResult Save(ProductModel model)

{

var dto = ModelConverter.ToProductDTO(model);

if (model.IsNew)

return \_productService.Create(dto);

else

return \_productService.Edit(dto);

}

public ActionResult Remove(Guid id)

{

var result = \_productService.GetById(id);

if (result.IsSucceed)

{

result.Data.IsDeleted = true;

\_productService.Edit(result.Data);

}

return result;

}

public override ProductModel ConvertToModel(ProductDTO dto) => ModelConverter.ToProductModel(dto);

public string GenerateRandomBarcode()

{

var \_rnd = new Random(DateTime.Now.Year \* 10000 + DateTime.Now.Millisecond);

while (true)

{

var random = \_rnd.NextDouble();

random \*= 100000000000000;

var tempBarcode = Math.Round(random, 0).ToString(CultureInfo.InvariantCulture);

var existResult = \_productService.GetAll().Data.Any(x => x.Barcode == tempBarcode);

if (existResult || tempBarcode[0] == '0')

{

GenerateRandomBarcode();

}

return tempBarcode;

}

}

public ActionResult ReadProductsFromExcel(ProductModel model)

{

if (model == null || model.File == null || model.File.Length == 0)

{

return ActionResult.Failure(new Exception("Файл не выбран или пустой."));

}

var allowedExtensions = new[] { ".xls", ".xlsx" };

var fileExtension = Path.GetExtension(model.File.FileName).ToLower();

if (!allowedExtensions.Contains(fileExtension))

{

return ActionResult.Failure(new Exception("Неверный формат файла."));

}

var products = new List<ProductModel>();

System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);

try

{

using (var stream = model.File.OpenReadStream())

{

using (var reader = ExcelReaderFactory.CreateReader(stream))

{

int row = 0;

while (reader.Read())

{

row++;

if (row == 1) continue; // Пропускаем заголовок

var columns = reader.FieldCount;

var code = columns > 0 ? reader.GetValue(0)?.ToString()?.Trim() : null;

var name = columns > 1 ? reader.GetValue(1)?.ToString()?.Trim() : null;

var barcode = columns > 2 ? reader.GetValue(2)?.ToString()?.Trim() : null;

decimal? price = null;

if (columns > 3)

{

var priceValue = reader.GetValue(3)?.ToString()?.Trim();

if (!string.IsNullOrWhiteSpace(priceValue) && decimal.TryParse(priceValue, out var parsedPrice))

{

price = parsedPrice;

}

}

Guid? unitId = null;

if (columns > 4)

{

var unitIdValue = reader.GetValue(4)?.ToString()?.Trim();

if (!string.IsNullOrWhiteSpace(unitIdValue) && Guid.TryParse(unitIdValue, out var parsedUnitId))

{

unitId = parsedUnitId;

}

}

Guid? productGroupId = null;

if (columns > 5)

{

var productGroupIdValue = reader.GetValue(5)?.ToString()?.Trim();

if (!string.IsNullOrWhiteSpace(productGroupIdValue) && Guid.TryParse(productGroupIdValue, out var parsedProductGroupId))

{

productGroupId = parsedProductGroupId;

}

}

var description = columns > 6 ? reader.GetValue(6)?.ToString()?.Trim() : null;

// Обязательно Code и Name

if (string.IsNullOrWhiteSpace(code) || string.IsNullOrWhiteSpace(name))

{

continue;

}

// Если уже есть такой продукт — пропускаем

var existingProduct = GetAll().FirstOrDefault(x => x.Code == code);

if (existingProduct != null)

{

continue;

}

var product = new ProductModel

{

Code = code,

Name = name,

Barcode = barcode,

Price = price,

UnitId = unitId ?? Guid.Empty,

ProductGroupId = productGroupId ?? Guid.Empty,

Description = description,

IsDeleted = false

};

products.Add(product);

}

}

}

foreach (var product in products)

{

var result = Save(product);

if (!result.IsSucceed)

{

return ActionResult.Failure(new Exception($"Ошибка сохранения продукта: {product.Code}. Подробности: {result.ExceptionMessage}"));

}

}

return ActionResult.Succeed();

}

catch (Exception ex)

{

return ActionResult.Failure(ex);

}

}

}

}

**Productmodel**

[Display(Name = "ExcelFile")]

public IFormFile File { get; set; }

**\_GridButtonPartial**

@model Tuple<String, String, Boolean>

<br />

<div class="row mb-3">

<div class="col">

<button id="btnNew" type="button" class="btn btn-primary active" onclick="toForm(null);"><i class="fa fa-plus"></i> @stringLocalizer["AddNew"]</button>

@if (Model.Item3)

{

<button id="btnEdit" type="button" class="btn btn-primary active" disabled="disabled" onclick="toEditForm(selectedId);"><i class="fa fa-edit"></i> @stringLocalizer["Edit"]</button>

<button id="btnRemove" type="button" class="btn btn-danger active" disabled="disabled" onclick="remove('@Model.Item1')"><i class="fa fa-trash"></i> @stringLocalizer["Remove"]</button>

<button id="btnExecute" type="button" class="btn btn-info active" disabled="disabled" onclick="getExecuteWorkModal();"><i class="fa fa-send"></i> @stringLocalizer["Execute"]</button>

}

else

{

<button id="btnEdit" type="button" class="btn btn-primary active" onclick="toEditForm(selectedId);"><i class="fa fa-edit"></i> @stringLocalizer["Edit"]</button>

<button id="btnRemove" type="button" class="btn btn-danger active" onclick="remove('@Model.Item1')"><i class="fa fa-trash"></i> @stringLocalizer["Remove"]</button>

}

<button id="btnPrint" type="button" class="btn btn-primary active" onclick="printDoc(selectedId);"><i class="fa fa-send"></i> @stringLocalizer["Print"]</button>

<button type="button" class="btn btn-info active" onclick="dtGrid.reload();"><i class="fa fa-refresh"></i> @stringLocalizer["Refresh"]</button>

<button type="button" class="btn btn-info active" onclick="dtGrid.filter();"><i class="fa fa-search"></i> @stringLocalizer["Search"]</button>

<button id="btnPrintToExcel" type="button" class="btn btn-outline-success active" onclick="printToExcel();"><i class="fa fa-file-excel-o" aria-hidden="true"></i> @stringLocalizer["Export"]</button>

<button id="btnImportExcel" type="button" class="btn btn-warning active" onclick="openImportProductModal();"><i class="fa fa-upload" aria-hidden="true"></i> @stringLocalizer["ImportExcel"]</button>

</div>

</div>

<div id="importProductModalContainer"></div>

<script type="text/javascript">

function remove(url) {

dialog.confirm('@stringLocalizer["AreYouSureRemoveData"]', function () {

$.ajax(url, {

data: { id: selectedId },

method: "POST",

success: function (response) {

dtGrid.reload();

},

error: function (error) {

toastr.error('Artıq istifadə olunub!');

}

});

});

}

function toForm(id) {

window.open( '@Model.Item2/' + id);

}

function toEditForm(id) {

window.open('@Model.Item2/' + id, "\_blank");

}

function printToExcel() {

$("#dt-grid").dxDataGrid("instance").exportToExcel(false);

}

function openImportProductModal() {

$.get('@Url.Action("GetImportModal", "Product")', function(data) {

$('#importProductModalContainer').html(data);

$('#importProductModal').modal('show');

}).fail(function(xhr, status, error) {

console.error('AJAX Error:', xhr.responseText);

toastr.error('@stringLocalizer["ErrorLoadingModal"]');

});

}

</script>

**Productcontroller**

[HttpPost]

[Route("import-excel-products")]

[ValidateAntiForgeryToken]

public IActionResult ImportExcelProducts(ProductModel model)

{

var result = \_productServiceFacade.ReadProductsFromExcel(model);

return Responce(result);

}

[HttpGet]

[Route("GetImportModal")]

public IActionResult GetImportModal()

{

return PartialView("\_ProductImportModal", new ProductModel());

}

\_ProductImportModal

@using HM.ClientUI.Models

@model ProductModel

<div id="importProductModal" class="modal fade show" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" style="display: block; padding-right: 16.9957px;" aria-modal="true">

<**form** class="form-horizontal m-t-20"

**asp-action**="ImportExcelProducts"

**asp-controller**="Product"

enctype="multipart/form-data"

data-ajax="true"

data-ajax-method="POST"

data-ajax-begin="util.setLoader(true);"

data-ajax-success="successImportProductModal"

data-ajax-failure="actionUtil.failure"

data-ajax-complete="util.setLoader(false);">

@Html.AntiForgeryToken()

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title mt-0" id="myModalLabel">@stringLocalizer["ImportProducts"]</h5>

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

</div>

<div class="modal-body">

<div class="container">

<div class="form-group row">

<div class="col-sm-12">

<div class="alert alert-info">

<strong>@stringLocalizer["ImportInstructions"]:</strong>

<ul class="mt-2">

<li>@stringLocalizer["ExcelFormat"]: Code, Name, Barcode, Price, UnitId, ProductTypeId, ProductGroupId, Description</li>

<li>@stringLocalizer["FirstRowHeaders"]</li>

<li>@stringLocalizer["RequiredFields"]: Code, Name</li>

<li>@stringLocalizer["ExistingProductsSkipped"]</li>

</ul>

</div>

</div>

</div>

<div class="form-group row">

<div class="col-sm-12 col-md-12 col-lg-12">

<h6 class="sub-title">@stringLocalizer["SelectExcelFile"]</h6>

<div class="form-group row">

<div class="col-sm-12">

<label>@stringLocalizer["SelectExcelFile"]</label>

<input type="file" name="File" class="form-control" accept=".xls,.xlsx" required />

</div>

</div>

</div>

</div>

</div>

</div>

<div class="modal-footer">

@(Html.DevExtreme().Button()

.ID("importProductBtn")

.Text(stringLocalizer["Import"])

.Type(DevExtreme.AspNet.Mvc.ButtonType.Success)

.UseSubmitBehavior(true))

<button type="button" class="btn btn-secondary" data-dismiss="modal">@stringLocalizer["Close"]</button>

</div>

</div>

</div>

</**form**>

</div>

<script>

$(function () {

$("#productFileUploader").dxFileUploader({

allowedFileExtensions: [".xls", ".xlsx"],

uploadMode: "useForm",

selectButtonText: "@stringLocalizer["SelectFile"]",

labelText: "@stringLocalizer["DragFileHere"]"

});

});

function successImportProductModal(data) {

if (data.isSucceed) {

toastr.success(data.message || '@stringLocalizer["ImportSuccessful"]');

$('#importProductModal').modal('hide');

if (typeof dtGrid !== 'undefined') {

dtGrid.reload();

}

} else {

toastr.error(data.message || '@stringLocalizer["ImportFailed"]');

}

}

function openImportProductModal() {

$('#importProductModal').modal('show');

}

</script>