ЗАТВЕРДЖЕНО

1116130.00898-01 12 01-ЛЗ

СИСТЕМА ЗАПИТУ ДОВІДОК ПРО ВАНТАЖНІ ПЕРЕВЕЗЕННЯ, ІНТЕГРОВАНА В ЄДИНИЙ КОРПОРАТИВНИЙ ІНФОРМАЦІЙНИЙ ПОРТАЛ УКРЗАЛІЗНИЦІ

Текст програми

1116130.00898-01 12 01

Аркушів 48

2016

АНОТАЦІЯ

Документ 1116130.00898-01 12 01 «Система запиту довідок про вантажні перевезення, інтегрована в єдиний корпоративний інформаційний портал Укрзалізниці. Текст програми» входить до складу програмної документації програмного засобу для форму-вання документів оперативної звітності про вантажні перевезення на Укрзалізниці.

Програму «Система запиту довідок про вантажні перевезення, інтегрована в єдиний корпоративний інформаційний портал Укрзалізниці» реалізовано в середовищі Microsoft Visual Studio на мові C# та Javascript (Typescript).

ЗМІСТ

[1 Інформація про класи та файли інтерфейсу «Система запиту довідок про вантажні перевезення, інтегрована в єдиний корпоративний інформаційний портал Укрзалізниці» 5](#_Toc452079409)

[2 Текст програми «Система запиту довідок про вантажні перевезення, інтегрована в єдиний корпоративний інформаційний портал Укрзалізниці» 8](#_Toc452079410)

[2.1 Текст класу DefaultHandler 8](#_Toc452079411)

[2.2 Текст класу FindHandler 8](#_Toc452079412)

[2.3 Текст класу MenuHandler 9](#_Toc452079413)

[2.4 Текст класу FormHandler 10](#_Toc452079414)

[2.5 Текст класу QueryHandler 16](#_Toc452079415)

[2.6 Текст класу AuthenticationHandler 16](#_Toc452079416)

[2.7 Текст класу ExcelHandler 17](#_Toc452079417)

[2.8 Текст класу ConfigurationProvider 17](#_Toc452079418)

[2.9 Текст класу FindProvider 18](#_Toc452079419)

[2.10 Текст класу MenuProvider 19](#_Toc452079420)

[2.11 Текст класу QProvider 20](#_Toc452079421)

[2.12 Текст класу DataProvider 21](#_Toc452079422)

[2.13 Текст класу QueryServiceDataProvider 27](#_Toc452079423)

[2.14 Текст класу UserProvider 27](#_Toc452079424)

[2.15 Текст класу SprItem 29](#_Toc452079425)

[2.16 Текст класу FormItem 29](#_Toc452079426)

[2.17 Текст класу DefaultItem 30](#_Toc452079427)

[2.18 Текст класу QueryModel 31](#_Toc452079428)

[2.19 Текст класу QueryItem 31](#_Toc452079429)

[2.20 Текст класу UserModel 31](#_Toc452079430)

[2.21 Текст класу FindItem 31](#_Toc452079431)

[2.22 Текст класу MenuItem 31](#_Toc452079432)

[2.23 Текст класу SearchModel 32](#_Toc452079433)

[2.24 Текст класу StaticConfiguration 32](#_Toc452079434)

[2.25 Текст класу DynamicConfiguration 33](#_Toc452079435)

[2.26 Текст файлу index.html 33](#_Toc452079436)

[2.27 Текст файлу authPartial.html 33](#_Toc452079437)

[2.28 Текст файлу mainView.html 34](#_Toc452079438)

[2.29 Текст файлу menuView.html 34](#_Toc452079439)

[2.30 Текст файлу searchView.html 34](#_Toc452079440)

[2.31 Текст файлу formView.html 35](#_Toc452079441)

[2.32 Текст файлу formViewPartial.html 35](#_Toc452079442)

[2.33 Текст файлу reportView.html 37](#_Toc452079443)

[2.34 Текст файлу app.ts 37](#_Toc452079444)

[2.35 Текст файлу authPartial.ts 38](#_Toc452079445)

[2.36 Текст файлу authService.ts 38](#_Toc452079446)

[2.37 Текст файлу mainView.ts 38](#_Toc452079447)

[2.38 Текст файлу menuView.ts 39](#_Toc452079448)

[2.39 Текст файлу menuService.ts 40](#_Toc452079449)

[2.40 Текст файлу searchView.ts 41](#_Toc452079450)

[2.41 Текст файлу searchService.ts 41](#_Toc452079451)

[2.42 Текст файлу formView.ts 42](#_Toc452079452)

[2.43 Текст файлу formService.ts 46](#_Toc452079453)

[2.44 Текст файлу reportView.ts 47](#_Toc452079454)

[2.45 Текст файлу reportService.ts 48](#_Toc452079455)

# ІНФОРМАЦІЯ ПРО КЛАСИ ТА ФАЙЛИ ІНТЕРФЕЙСУ «СИСТЕМА ЗАПИТУ ДОВІДОК ПРО ВАНТАЖНІ ПЕРЕВЕЗЕННЯ, ІНТЕГРОВАНА В ЄДИНИЙ КОРПОРАТИВНИЙ ІНФОРМАЦІЙНИЙ ПОРТАЛ УКРЗАЛІЗНИЦІ»

До програми «Система запиту довідок про вантажні перевезення, інтегрована в єдиний корпоративний інформаційний портал Укрзалізниці» входять наступні класи:

* DefaultHandler.cs – клас обробнику, який приймає усі запити з клієнстьскої сторони;
* FindHandler.cs ­­­­– клас обробнику запитів пошуку довідки;
* MenuHandler.cs – клас обробнику запитів на відображення ієрархії меню;
* FormHandler.cs – клас обробнику запітів на відображення форми вхідних параметрів довідки;
* QueryHandler.cs – клас обробнику запитів на завантаження інформації до списку значень параметрів форми;
* ReportHandler.cs – клас обробнику запитів отримання звіту;
* AuthenticationHandler.cs – клас обробнику запитів, які стосуються автентифікації користувача;
* ExcelHandler.cs – клас обробнику запитів на отримання звіту у вигляді файлу Excel;
* ConfigurationProvider.cs – клас, що забезпечує отримання поточних налаштувань;
* FindProvider.cs – клас, що забезпечує пошук довідки;
* MenuProvider.cs – клас, що забезпечує отримання даних меню ієрархії;
* QProvider.cs – клас, що забезпечує отримання даних для параметрів форми;
* DataProvider.cs – клас, що забезпечує отримання даних для форми вхідних параметрів;
* QueryServiceDataProvider.cs – клас, що забезпечує отримання даних звіту від сервісу довідки;
* UserProvider.cs – клас, що забезпечує отримання даних про поточного користувача;
* SprItem.cs – клас моделі даних довідки;
* SimpleFormItem.cs – клас моделі даних загального параметра довідки;
* FormItem.cs – клас моделі даних параметра довідки, що використовується в конкретній довідці;
* DefaultItem.cs – клас моделі даних значення параметра довідки;
* QueryModel.cs – клас моделі даних запиту для отримання значень параметру довідки;
* QueryItem.cs – клас моделі даних параметру довідки, від якого залежить запит отримання значень поточного параметру довідки;
* UserModel.cs – клас моделі даних поточного користувача;
* FindItem.cs – клас моделі даних результату пошуку довідки;
* MenuItem.cs – клас моделі даних пункту меню ієрархії довідок;
* SearchModel – клас моделі даних запиту для пошуку довідки;
* StaticConfiguration.cs – клас моделі даних незмінної конфігурації програми;
* DynamicConfiguration.cs – клас моделі даних конфігурації, що завантажується динамічно із списків порталу.

До програми також входять наступні файли інтерфейсу:

* index.html – контейнер форм програми;
* authPartial.html – частина форми із даними про користувача;
* mainView.html – головна форма програми;
* menuView.html – форма меню ієрархії;
* searchView.html – форма із результатми пошуку довідки;
* formView.html – форма вхідних даних довідки;
* formViewPartial.html – форма відображення параметрів вхідних даних;
* reportView.html – форма звіту;
* app.ts – файл із головною логікою клієнтської частини;
* authPartial.ts – файл із логікою автентифікації;
* authService.ts – файл із логікою обміну інформації про авторизацію з сервером;
* mainView.ts – файл із логікою головної форми;
* menuView.ts – файл із логікою відображення меню ієрархії довідок;
* menuService.ts – файл із логікою завантаження інформації про меню ієрархії довідок з сервера;
* searchView.ts – файл із логікою пошуку довідок;
* searchService.ts – файл із логікою запитів довідок
* formView.ts – файл із логікою побудови форми вхідних параметрів;
* formService.ts – файл із логікою завантаження даних форми вхідних параметрів з серверу;
* reportView.ts – файл із логікою відображення звіту;
* reportService.ts – файл із логікою завантаження даних звіту з серверу.

# ТЕКСТ ПРОГРАМИ «СИСТЕМА ЗАПИТУ ДОВІДОК ПРО ВАНТАЖНІ ПЕРЕВЕЗЕННЯ, ІНТЕГРОВАНА В ЄДИНИЙ КОРПОРАТИВНИЙ ІНФОРМАЦІЙНИЙ ПОРТАЛ УКРЗАЛІЗНИЦІ»

## Текст класу DefaultHandler

using System;

using System.Security;

using System.ServiceModel.Channels;

using System.Web;

using System.Web.SessionState;

using AskvpDeskTop.Exceptions;

using AskvpDeskTop.Model;

using Ninject;

using System.Net;

using NLog;

namespace AskvpDeskTop.HttpHandler

{

public class DefaultHandler : IHttpHandler,IRequiresSessionState, IInjectable

{

private IKernel kernel;

#region IHttpHandler Members

public bool IsReusable

{

// Return false in case your Managed Handler cannot be reused for another request.

// Usually this would be false in case you have some state information preserved per request.

get { return false; }

}

public void ProcessRequest(HttpContext context)

{

if (context.Request.HttpMethod == "OPTIONS")

//return;

context.Response.TrySkipIisCustomErrors = true;

//context.Response.Headers["Access-Control-Allow-Origin"] = "\*";

try

{

var requestPath = context.Request.Path;

var handlerName = requestPath

.Substring(requestPath.LastIndexOf('/') + 1)

.ToLower();

var handler = kernel.Get<IHandler>(handlerName);

handler.Process(context);

}

catch (SecurityException)

{

PrepearErrorHttpResponse(context);

context.Response.Write(Errors.Security);

}

catch (SprNotFoundException)

{

PrepearErrorHttpResponse(context);

context.Response.Write(Errors.SprNotFound);

}

catch (QueryServiceDataException ex)

{

PrepearErrorHttpResponse(context);

var logger = LogManager.GetLogger("QueryServiceData");

var err = new LogEventInfo(LogLevel.Error, "QueryServiceData", null);

err.Properties.Add("sprcode", ex.SprCode);

err.Properties.Add("serviceurl", ex.ServiceUrl);

err.Properties.Add("inputxml", ex.InputXML);

err.Properties.Add("outputxml", ex.OutputXML);

logger.Log(err);

context.Response.Write(Errors.Default);

}

catch (Exception ex)

{

PrepearErrorHttpResponse(context);

var logger = LogManager.GetLogger("AskpvDesktopError");

var err = new LogEventInfo(LogLevel.Error, "QueryServiceData", ex.Message);

err.Properties.Add("stacktrace", ex.StackTrace);

logger.Log(err);

context.Response.Write(Errors.Default);

}

}

#endregion

private void PrepearErrorHttpResponse(HttpContext context)

{

context.Response.ClearContent();

context.Response.ContentType = "text/plain";

context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;

}

#region IInjectable Members

public void Inject(IKernel kernel)

{

this.kernel = kernel;

}

#endregion;

}

}

## 

## Текст класу FindHandler

using System;

using System.Linq;

using System.ServiceModel;

using AskvpDeskTop.Model;

using AskvpDeskTop.Provider;

using System.Web;

using Newtonsoft.Json.Linq;

using Newtonsoft.Json;

using System.IO;

namespace AskvpDeskTop.HttpHandler

{

public class FindHandler : IHandler

{

private readonly IFindProvider \_findProvider;

private readonly IUserProvider \_userProvider;

public FindHandler(IFindProvider findProvider, IUserProvider userProvider)

{

\_findProvider = findProvider;

\_userProvider = userProvider;

}

public void Process(HttpContext context)

{

var source = string.Empty;

using (var sr = new StreamReader(context.Request.InputStream))

source = sr.ReadToEnd();

var searchItem = JsonConvert.DeserializeObject<SearchModel>(source);

if (string.IsNullOrEmpty(searchItem.Pattern))

throw new ArgumentException("Query is null or empty", "Query");

var user = \_userProvider.GetUser();

var findItems = \_findProvider.Find(searchItem.Pattern, searchItem.Personalized, user.UserID).ToList();

var resultJArray = new JArray();

var countOfItems = 0;

if (findItems.Any())

{

countOfItems = findItems.Count;

var offset = (searchItem.PageNum - 1)\*searchItem.ItemsPerPage;

var lastItems = countOfItems - offset;

resultJArray =

JArray.FromObject(findItems.GetRange(offset,

lastItems > searchItem.ItemsPerPage ? searchItem.ItemsPerPage: lastItems));

}

context.Response.Write(new JObject(

new JProperty("query", searchItem.Pattern),

new JProperty("countOfItems",countOfItems),

new JProperty("suggestions", resultJArray)

).ToString(Formatting.None));

}

}

}

## 

## Текст класу MenuHandler

using System;

using System.Web;

using AskvpDeskTop.Model;

using AskvpDeskTop.Provider;

using Microsoft.SharePoint.WebControls;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System.Linq;

using System.Collections.Generic;

namespace AskvpDeskTop.HttpHandler

{

public class MenuHandler : IHandler

{

private readonly IMenuProvider \_menuProvider;

private readonly IUserProvider \_userProvider;

/// <summary>

/// Специальный символ для сортировки меню

/// </summary>

private const char SpecChar = '$';

public MenuHandler(IMenuProvider menuProvider, IUserProvider userProvider)

{

\_menuProvider = menuProvider;

\_userProvider = userProvider;

}

public void Process(HttpContext context)

{

var personalized = !string.IsNullOrEmpty(context.Request.QueryString["option"]) &&

bool.Parse(context.Request.QueryString["option"]);

IEnumerable<MenuItem> menuItems;

var user = \_userProvider.GetUser();

if (personalized)

{

menuItems = \_menuProvider.GetPersonalMenu(user.UserID);

}

else

{

var strFolderId = context.Request.QueryString["folderid"];

if (string.IsNullOrEmpty(strFolderId)) throw new ArgumentException("Is null or empty", "folderid");

int folderId;

if (!int.TryParse(strFolderId, out folderId)) throw new ArgumentException("Is not integer", "folderid");

menuItems = \_menuProvider.GetMenu(folderId,user.UserID).ToList();

if (menuItems.Any()) menuItems = SortingMenu(menuItems);

}

context.Response.Write(new JObject(

new JProperty("menu",

JArray.FromObject(menuItems))).ToString(Formatting.None));

}

/// <summary>

/// Удаление номеров необходимых (1$...) для сортировки меню

/// </summary>

private static string DeleteMenuCount(string menuStr)

{

var indexOfSpecChar = menuStr.IndexOf(SpecChar);

if (indexOfSpecChar != -1)

{

menuStr = menuStr.Substring(indexOfSpecChar + 1);

}

return menuStr;

}

/// <summary>

/// Сортировка меню

/// </summary>

/// <param name="menuItems"></param>

private static IEnumerable<MenuItem> SortingMenu(IEnumerable<MenuItem> menuItems)

{

var sortedFolderItems = (from s in menuItems

where s.Title.Contains(SpecChar) && s.ItemType == MenuItemType.Folder

orderby double.Parse(s.Title.Substring(0, s.Title.IndexOf(SpecChar)))

select new MenuItem

{

Title = DeleteMenuCount(s.Title),

ID = s.ID,

ItemType = s.ItemType

})

.Union(

menuItems.Where(i => !i.Title.Contains(SpecChar) && i.ItemType == MenuItemType.Folder).Select(s => s).OrderBy(s => s.Title)).ToList();

var sortedItems = menuItems.Where(i => i.ItemType == MenuItemType.Link).Select(s => s).OrderBy(s => s.ID).ToList();

return sortedFolderItems.Union(sortedItems);

}

}

}

## 

## Текст класу FormHandler

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Web;

using System.Web.SessionState;

using AskvpDeskTop.Model;

using AskvpDeskTop.Provider;

using System.Net;

using Microsoft.SharePoint;

using Microsoft.SharePoint.WebControls;

using System.IO;

using Newtonsoft.Json.Linq;

using Newtonsoft.Json;

using Ninject.Activation;

using Ninject.Syntax;

namespace AskvpDeskTop.HttpHandler

{

public class FormHandler : IHandler

{

private readonly IDataProvider dataProvider;

private readonly IUserProvider userProvider;

public FormHandler(IDataProvider dataProvider, IUserProvider userProvider)

{

this.dataProvider = dataProvider;

this.userProvider = userProvider;

}

public void MainProcess(HttpContext context)

{

if (context.Request.QueryString["id\_spr"] != null)

{

var sprCode = context.Request.QueryString["id\_spr"];

if (string.IsNullOrEmpty(sprCode)) throw new ArgumentException("sprCode is null or empty", "sprCode");

if (userProvider.UserHasPermission(sprCode))

{

var sprItem = new SprItem();

/\*if (context.Session[sprCode] != null)

{

/\*получаем объект и делаем его копию, так как будем его менять #1#

var serialized = JsonConvert.SerializeObject(context.Session[sprCode] as SprItem);

sprItem = JsonConvert.DeserializeObject<SprItem>(serialized);

}

else

{\*/

sprItem = dataProvider.CreateForm(sprCode);

// context.Session[sprCode] = sprItem;

//}

var tabSessionID = context.Request.QueryString["tsid"];

if (tabSessionID != null)

{

var selectedItems =

context.Session[StaticConfiguration.CreateSessionKey(

StaticConfiguration.FormSessionToken, sprCode, tabSessionID)] as List<SimpleFormItem>;

/\* данные ,которые выбрал пользователь, отсутствуют. Устанавливаем персонализированные данные\*/

selectedItems = selectedItems ?? dataProvider.GetUserConfig(sprCode, userProvider.GetUser().UserID);

if (selectedItems != null)

{

var selectedParametrNameItems = selectedItems.Select(el => el.ParametrName).ToList();

foreach (

var formItem in

sprItem.FormItems.Where(

formItem => selectedParametrNameItems.Contains(formItem.ParametrName)))

{

formItem.DefaultValue =

selectedItems.Where(el => el.ParametrName == formItem.ParametrName)

.Select(elem => elem.DefaultValue).First();

}

}

}

var rows = new List<List<JObject>>();

var row = new List<JObject>();

var schemaJObject = new JObject();

foreach (var formItem in sprItem.FormItems)

{

/\*межстроковая подпись расположенная перед элементом\*/

if (!String.IsNullOrEmpty(formItem.TextRowBefore))

{

rows.Add(new List<JObject>()

{

new JObject

{

new JProperty("Description", formItem.TextRowBefore),

new JProperty("TypeParameter", "Label" )

}

});

}

var formJObject = new JObject(

new JProperty("Description", formItem.Description),

new JProperty("ParametrAlign", formItem.ParametrAlign.ToString()),

new JProperty("TypeParameter", formItem.TypeParameter.ToString()),

new JProperty("DescriptionAlign", formItem.DescriptionAlign.ToString()),

new JProperty("ParametrName", formItem.ParametrName),

new JProperty("DescrGridMobile", formItem.DescriptionGridMobile),

new JProperty("DescrGridMobileOffset", formItem.DescriptionGridMobileOffset),

new JProperty("DescrGridDesktop", formItem.DescriptionGridDesktop),

new JProperty("DescrGridDesktopOffset", formItem.DescriptionGridDesktopOffset),

new JProperty("ParamGridMobile", formItem.ParametrGridMobile),

new JProperty("ParamGridMobileOffset", formItem.ParametrGridMobileOffset),

new JProperty("ParamGridDesktop", formItem.ParametrGridDesktop),

new JProperty("ParamGridDesktopOffset", formItem.ParametrGridDesktopOffset));

/\* var formJProperty = new JProperty(formItem.ParametrName,

formJObject

);

\*/

var schemJObject = new JObject();

var schemaJProperty = new JProperty(formItem.ParametrName, schemJObject);

AppendMainProp(formItem, schemJObject);

switch (formItem.TypeParameter)

{

case TypeParameters.DropDownList:

case TypeParameters.ListBox:

case TypeParameters.AutoComplete:

{

formJObject.Add(new JProperty("Multiselect", formItem.MultiSelect));

AppendDropDownProp(formItem, schemJObject);

}

break;

case TypeParameters.Calendar:

case TypeParameters.CalendarWithoutTime:

{

AppendCalendarProp(formItem, schemJObject);

}

break;

case TypeParameters.CheckBox:

{

AppendCheckBoxProp(formItem, schemJObject);

}

break;

case TypeParameters.TimeBox:

{

}

break;

case TypeParameters.TextBox:

{

formJObject.Add(new JProperty("CharacterLength", formItem.CharacterLength));

formJObject.Add(new JProperty("TextRows", formItem.TextRows));

formJObject.Add(new JProperty("TextModes", formItem.TextMode.ToString()));

AppendTextBoxProp(formItem, schemJObject);

}

break;

case TypeParameters.RadioButton:

{

AppendRadioButtonProp(formItem, schemJObject);

}

break;

}

schemaJObject.Add(schemaJProperty);

/\*межстроковая подпись расположенная после элемента\*/

if (!String.IsNullOrEmpty(formItem.TextRowAfter))

{

row.Add(formJObject);

rows.Add(row);

row = new List<JObject>();

rows.Add(new List<JObject>()

{

new JObject

{

new JProperty("TypeParameter", "Label"),

new JProperty("Description", formItem.TextRowAfter)

}

});

}

else

{

if (!formItem.NextRow)

{

row.Add(formJObject);

continue;

}

/\* в свойствах элемента указано перевод на след строку,

\* но перевод строки был уже сделан, так как до этого

\* была указана строка после элемента\*/

if(row.Count > 0) rows.Add(row);

row = new List<JObject>();

row.Add(formJObject);

}

}

if (row.Count > 0) rows.Add(row);

context.Response.Write(new JObject(new JProperty("schema", schemaJObject),

new JProperty("form", new JObject(new JProperty("rows", JArray.FromObject(rows)))),

new JProperty("Title",sprItem.Title),new JProperty("ID",sprCode)

).ToString(

Formatting.None));

}

}

if (context.Request.QueryString["savepd"] != null)

{

string sprCode = context.Request.QueryString["savepd"];

if (userProvider.UserHasPermission(sprCode))

{

var user = userProvider.GetUser();

var source = string.Empty;

using (var sr = new StreamReader(context.Request.InputStream))

source = sr.ReadToEnd();

dataProvider.UpdateUserConfig(sprCode, user.UserID, source);

}

}

if (context.Request.QueryString["save"] != null)

{

string sprCode = context.Request.QueryString["save"];

if (userProvider.UserHasPermission(sprCode))

{

var tabSessionID = context.Request.QueryString["tsid"];

var source = string.Empty;

using (var sr = new StreamReader(context.Request.InputStream))

source = sr.ReadToEnd();

var selectedItems = JsonConvert.DeserializeObject<List<SimpleFormItem>>(source);

context.Session[StaticConfiguration.CreateSessionKey(StaticConfiguration.FormSessionToken,

sprCode, tabSessionID)] = selectedItems;

}

}

}

public void Process(HttpContext context)

{

MainProcess(context);

}

private void AppendMainProp(FormItem formItem, JContainer jProperty)

{

jProperty.Add(new JProperty("TypeParameter", formItem.TypeParameter.ToString()));

jProperty.Add(new JProperty("FilteringAction", formItem.FilteringAction.ToString()));

jProperty.Add(new JProperty("SprFilteringAction", formItem.SprFilteringAction.ToString()));

if (String.IsNullOrEmpty(formItem.AffectNameParametr) ||

formItem.AffectSourceObject == AffectSourceObjects.None ||

(formItem.AffectSource.Count <= 0 && String.IsNullOrEmpty(GetExistDasMethod(formItem.AffectDasMethod))) ||

formItem.DependAction == DependActions.None) return;

jProperty.Add(new JProperty("AffectNameParametr", formItem.AffectNameParametr));

jProperty.Add(new JProperty("AffectSourceObject", formItem.AffectSourceObject.ToString()));

jProperty.Add(new JProperty("DependAction", formItem.DependAction.ToString()));

switch (formItem.AffectSourceObject)

{

case AffectSourceObjects.DAS:

{

jProperty.Add(new JProperty("AffectDasMethod", GetExistDasMethod(formItem.AffectDasMethod)));

}

break;

case AffectSourceObjects.List:

{

jProperty.Add(new JProperty("AffectSource", JArray.FromObject(formItem.AffectSource.Select(item=>item.Value))));

//jProperty.Add(new JProperty("AffectSource", JArray.FromObject(formItem.AffectSource)));

}

break;

case AffectSourceObjects.DateTimeList:

{

/\*var listSourceDictionary = (from listSourceItem in formItem.AffectSource

let tempListSourceItemKey = ConvertDateTimeVariable(listSourceItem.Value)

select new DefaultItem

{

Value = tempListSourceItemKey,

NValue = listSourceItem.NValue ?? tempListSourceItemKey

}).ToList();\*/

var keysList = formItem.AffectSource.Select(item => item.Value).ToList();

keysList.ForEach(el => ConvertDateTimeVariable(el));

jProperty.Add(new JProperty("AffectSource", JArray.FromObject(keysList)));

}

break;

}

}

private void AppendCalendarProp(FormItem formItem, JContainer jProperty)

{

DateTime? tempDefaultValue = DateTime.Now;

if (formItem.DefaultValue != null && formItem.DefaultValue.Count > 0)

{

var defaultVal = formItem.DefaultValue[0].Value;

if (!String.IsNullOrEmpty(defaultVal))

{

tempDefaultValue = ConvertCalendarVariable(defaultVal);

}

}

/\*var strTempDefaultValue = formItem.TypeParameter == TypeParameters.Calendar ? tempDefaultValue.ToString("dd.MM.yyyy HH:mm:ss") :tempDefaultValue.ToShortDateString();\*/

/\*jProperty.Add(new JProperty("DefaultValue",

JArray.FromObject(new List<DefaultItem> {new DefaultItem {Value = tempDefaultValue.ToString("O")}})));\*/

jProperty.Add(new JProperty("DefaultValue",

JArray.FromObject(new List<DefaultItem> {new DefaultItem {Value = !tempDefaultValue.HasValue ? null: tempDefaultValue.Value.ToString(new CultureInfo("uk-UA"))}})));

}

private void AppendDropDownProp(FormItem formItem, JContainer jProperty)

{

jProperty.Add(new JProperty("SourceObject", formItem.SourceObject.ToString()));

switch (formItem.SourceObject)

{

#region DAS

case SourceObjects.DAS:

{

if (formItem.DependOn.Count > 0)

jProperty.Add(new JProperty("DependOn", formItem.DependOn));

jProperty.Add(new JProperty("KeyName",formItem.KeyName));

jProperty.Add(new JProperty("DasNamedParams", formItem.DasNamedParams));

jProperty.Add(new JProperty("DasType", formItem.DasType.ToString()));

AppendPropArrayDefaultVal(formItem, jProperty);

}

break;

#endregion

#region LIST or DateTimeLIST

case SourceObjects.LIST:

case SourceObjects.DateTimeLIST:

{

if (formItem.DefaultValue != null && formItem.DefaultValue.Count > 0)

{

var defaultVal = formItem.DefaultValue[0].Value;

if (!String.IsNullOrEmpty(defaultVal))

{

ConvertDateTimeVariable(defaultVal);

}

}

var listSourceDictionary = (from listSourceItem in formItem.Data

let tempListSourceItemKey = ConvertDateTimeVariable(listSourceItem.Value)

select new DefaultItem

{

Value = tempListSourceItemKey,

NValue = listSourceItem.NValue ?? tempListSourceItemKey

}).ToList();

AppendPropArrayDefaultVal(formItem, jProperty);

jProperty.Add(new JProperty("Data", JArray.FromObject(listSourceDictionary)));

}

break;

#endregion

}

}

private static void AppendPropArrayDefaultVal(FormItem formItem, JContainer jProperty)

{

JArray defaulVal = null;

if (formItem.DefaultValue != null && formItem.DefaultValue.Count > 0)

{

defaulVal = JArray.FromObject(formItem.DefaultValue);

}

jProperty.Add(new JProperty("DefaultValue", defaulVal));

}

private void AppendCheckBoxProp(FormItem formItem, JContainer jProperty)

{

string defaulValue = "0";

if (formItem.DefaultValue != null && formItem.DefaultValue.Count > 0)

{

string defaultVal = formItem.DefaultValue[0].Value;

if (!String.IsNullOrEmpty(defaultVal))

{

defaulValue = defaultVal;

}

}

jProperty.Add(new JProperty("DefaultValue",

JArray.FromObject(new List<DefaultItem> { new DefaultItem { Value = defaulValue } })));

}

private void AppendRadioButtonProp(FormItem formItem, JContainer jProperty)

{

string defaulValue = "0";

if (formItem.DefaultValue != null && formItem.DefaultValue.Count > 0)

{

string defaultVal = formItem.DefaultValue[0].Value;

if (!String.IsNullOrEmpty(defaultVal))

{

defaulValue = defaultVal;

}

}

jProperty.Add(new JProperty("RadioGroup", formItem.RadioGroup));

jProperty.Add(new JProperty("DefaultValue",

JArray.FromObject(new List<DefaultItem> { new DefaultItem { Value = defaulValue } })));

}

private void AppendTextBoxProp(FormItem formItem, JContainer jProperty)

{

string defaulValue = null;

if (formItem.DefaultValue != null && formItem.DefaultValue.Count > 0)

{

string defaultVal = formItem.DefaultValue[0].Value;

if (!String.IsNullOrEmpty(defaultVal))

{

defaulValue = defaultVal;

}

}

jProperty.Add(new JProperty("DefaultValue",

JArray.FromObject(new List<DefaultItem> { new DefaultItem { Value = defaulValue } })));

}

private string GetExistDasMethod(string inputString)

{

if (!inputString.StartsWith("exist\_")) return null;

int stringLenght = "exist\_".Length;

return inputString.Substring(stringLenght - 1, inputString.Length - stringLenght);

}

private string ConvertDateTimeVariable(string dateTimeVariable)

{

//today или now

if (dateTimeVariable.Contains("today") ||

dateTimeVariable.Contains("now") ||

dateTimeVariable.Contains("railday"))

{

//today (время не учитывается), now (время не учитывается)

DateTime dt = DateTime.Now;

var uaCultureInfo = new CultureInfo("uk-UA");

int kol = 0, sign = 1, pos\_format = dateTimeVariable.IndexOf(',');

string hStr = "", period = "", date\_format = "";

if (pos\_format != -1)

date\_format = dateTimeVariable.Substring(pos\_format + 1).Trim();

;

if (dateTimeVariable.Length > 5 && dateTimeVariable.ToLower().Substring(0, 5) == "today" &&

dateTimeVariable.Length > 6 && dateTimeVariable[5] == '.')

{

dt = DateTime.Today;

hStr = dateTimeVariable.ToLower().Substring(6).Trim();

}

else if (dateTimeVariable.Length > 3 && dateTimeVariable.ToLower().Substring(0, 3) == "now" &&

dateTimeVariable.Length > 4 && dateTimeVariable[3] == '.')

{

hStr = dateTimeVariable.ToLower().Substring(4).Trim();

}

else if (dateTimeVariable.Length > 7 && dateTimeVariable.ToLower().Substring(0, 7) == "railday" &&

dateTimeVariable.Length > 8 && dateTimeVariable[7] == '.')

{

if (DateTime.Now >= Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo) + " 17:00:01",uaCultureInfo) &&

DateTime.Now <= Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo) + " 23:59:59",uaCultureInfo))

dt = Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo),uaCultureInfo);

else

dt = Convert.ToDateTime(DateTime.Today.AddDays(-1).ToString("d", uaCultureInfo),uaCultureInfo);

hStr = dateTimeVariable.ToLower().Substring(8).Trim();

}

if (hStr != "")

{

if (hStr.Length >= 3 &&

hStr.Substring(0, 3) == "day")

{

period = "day";

hStr = hStr.Substring(3).Trim();

}

else if (hStr.Length >= 5 &&

hStr.Substring(0, 5) == "month")

{

period = "month";

hStr = hStr.Substring(5).Trim();

}

else if (hStr.Length >= 4 &&

hStr.Substring(0, 4) == "year")

{

period = "year";

hStr = hStr.Substring(4).Trim();

}

else if (hStr.Length >= 7 &&

hStr.Substring(0, 7) == "seconds")

{

period = "seconds";

hStr = hStr.Substring(7).Trim();

}

else if (hStr.Length >= 7 &&

hStr.Substring(0, 7) == "minutes")

{

period = "minutes";

hStr = hStr.Substring(7).Trim();

}

else if (hStr.Length >= 5 &&

hStr.Substring(0, 5) == "hours")

{

period = "hours";

hStr = hStr.Substring(5).Trim();

}

if (period != "" && hStr.Length > 1)

{

if (hStr[0] == '-' || hStr[0] == '+')

{

if (hStr[0] == '-')

sign = -1;

hStr = hStr.Substring(1).Trim();

}

pos\_format = hStr.IndexOf(',');

if (pos\_format != -1)

{

if (pos\_format > 0)

kol = Convert.ToInt32(hStr.Substring(0, pos\_format).Trim());

}

else

kol = Convert.ToInt32(hStr.Substring(0).Trim());

if (pos\_format == -1 || pos\_format > 0)

{

switch (period)

{

case "day":

dt = dt.AddDays(sign\*kol);

break;

case "month":

dt = dt.AddMonths(sign\*kol);

break;

case "year":

dt = dt.AddYears(sign\*kol);

break;

case "seconds":

dt = dt.AddSeconds(sign\*kol);

break;

case "minutes":

dt = dt.AddMinutes(sign\*kol);

break;

case "hours":

dt = dt.AddHours(sign\*kol);

break;

}

}

}

}

if (date\_format == "")

{

if (dateTimeVariable.Length > 5 && dateTimeVariable.ToLower().Substring(0, 5) == "today" ||

dateTimeVariable.Length > 7 && dateTimeVariable.ToLower().Substring(0, 7) == "railday")

return dt.ToString("d", uaCultureInfo);

return dt.ToString(new CultureInfo("uk-UA"));

}

return dt.ToString(date\_format,uaCultureInfo);

}

return dateTimeVariable;

}

private DateTime? ConvertCalendarVariable(string calendarVariable)

{

var calDt = DateTime.Now;

var uaCultureInfo = new CultureInfo("uk-UA");

var sign = 1;

var tmp = "";

/\* если пробел, то calendar у нас "пустой" \*/

if (string.IsNullOrEmpty(calendarVariable.Trim())) return null;

if (string.IsNullOrEmpty(calendarVariable)) return DateTime.Now;

if (calendarVariable.Substring(0, 3).ToLower() == "now")

{

//минимальное количество при добавлении минут = 5

if (calendarVariable.Length > 4)

{

tmp = calendarVariable.Substring(3).Trim();

if (tmp.Length > 1)

{

if (tmp[0] == '-')

sign = -1;

string minutes = tmp.Substring(1).Trim();

calDt = calDt.AddMinutes(Convert.ToInt32(minutes)\*sign);

}

}

}

else

{

string days = "";

if (calendarVariable.Substring(0, 5).ToLower() == "today")

{

//минимальное количество при добавлении дней = 7

if (calendarVariable.Length > 6)

{

tmp = calendarVariable.Substring(5).Trim();

if (tmp.Length > 1)

{

if (tmp[0] == '-')

sign = -1;

days = tmp.Substring(1).Trim();

calDt = calDt.AddDays(Convert.ToInt32(days)\*sign);

}

}

}

else

{

if (calendarVariable.Substring(0, 7).ToLower() == "railday")

{

if (DateTime.Now >= Convert.ToDateTime(DateTime.Today.ToString("d",uaCultureInfo) + " 17:00:01", uaCultureInfo) &&

DateTime.Now <= Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo) + " 23:59:59", uaCultureInfo))

calDt = Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo) + " 17:00:00", uaCultureInfo);

else

calDt = Convert.ToDateTime(DateTime.Today.AddDays(-1).ToString("d", uaCultureInfo) + " 17:00:00", uaCultureInfo);

//минимальное количество при добавлении дней = 9

if (calendarVariable.Length > 8)

{

tmp = calendarVariable.Substring(7).Trim();

if (tmp.Length > 1)

{

if (tmp[0] == '-')

sign = -1;

days = tmp.Substring(1).Trim();

calDt = calDt.AddDays(Convert.ToInt32(days)\*sign);

}

}

}

else if (calendarVariable.Substring(0, 7).ToLower() == "railday1")

{

if (DateTime.Now >= Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo) + " 17:00:01",uaCultureInfo) &&

DateTime.Now <= Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo) + " 23:59:59",uaCultureInfo))

calDt = Convert.ToDateTime(DateTime.Today.ToString("d", uaCultureInfo) + " 17:01:00",uaCultureInfo);

else

calDt = Convert.ToDateTime(DateTime.Today.AddDays(-1).ToString("d", uaCultureInfo) + " 17:01:00",uaCultureInfo);

//минимальное количество при добавлении дней = 9

if (calendarVariable.Length > 8)

{

tmp = calendarVariable.Substring(7).Trim();

if (tmp.Length > 1)

{

if (tmp[0] == '-')

sign = -1;

days = tmp.Substring(1).Trim();

calDt = calDt.AddDays(Convert.ToInt32(days)\*sign);

}

}

}

else

calDt = Convert.ToDateTime(calendarVariable,new CultureInfo("uk-UA"));

}

}

return calDt;

}

}

}

## 

## Текст класу QueryHandler

using System;

using System.Linq;

using System.Web;

using System.Collections.Generic;

using AskvpDeskTop.Model;

using System.IO;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using IQueryProvider = AskvpDeskTop.Provider.IQProvider;

namespace AskvpDeskTop.HttpHandler

{

public class QueryHandler : IHandler

{

private readonly IQueryProvider queryProvider;

public QueryHandler(IQueryProvider queryProvider)

{

this.queryProvider = queryProvider;

}

public void Process(HttpContext context)

{

var source = string.Empty;

using (var sr = new StreamReader(context.Request.InputStream))

source = sr.ReadToEnd();

context.Response.Write(MainProcess(source));

}

private string MainProcess(string source)

{

var queryModel = JsonConvert.DeserializeObject<QueryModel>(source);

if (string.IsNullOrEmpty(queryModel.QueryName)) throw new ArgumentException("query");

var defaultItems = queryProvider.GetData(queryModel);

if (defaultItems == null || defaultItems.Count <= 0)

{

return new JObject(

new JProperty("data", new JArray())

).ToString(Formatting.None);

}

if (queryModel.SearchPattern != null)

{

queryModel.SearchPattern = queryModel.SearchPattern.ToLower();

defaultItems = defaultItems.Where(defaultItem => defaultItem.Value.ToLower().Contains(queryModel.SearchPattern) || defaultItem.NValue.ToLower().Contains(queryModel.SearchPattern)).ToList();

}

var obj = new JObject(new JProperty("isb", defaultItems.Count > 100));

obj.Add(new JProperty("data",JArray.FromObject(defaultItems)));

return obj.ToString(Formatting.None);

}

}

}

## 

## Текст класу AuthenticationHandler

using AskvpDeskTop.Provider;

using Microsoft.SharePoint;

using System;

using System.Web;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

namespace AskvpDeskTop.HttpHandler

{

public class AuthenticationHandler : IHandler

{

private readonly IUserProvider \_userProvider;

public AuthenticationHandler(IUserProvider userProvider)

{

\_userProvider = userProvider;

}

public void Process(HttpContext context)

{

if (context.Request.QueryString["usermenu"] != null)

{

string username;

//string loginPath = "/\_layouts/Authenticate.aspx?Source=" + HttpContext.Current.Request.Url.AbsolutePath;

string loginPath = "/\_layouts/Authenticate.aspx?Source=" + "/\_layouts/askvpdesktop.v2/main/dist/app/index.html";

if (SPContext.Current != null && SPContext.Current.Web.CurrentUser != null)

{

username = SPContext.Current.Web.CurrentUser.Name;

}

else

{

username = null;

}

var obj = new JObject(new JProperty("username", username), new JProperty("loginPath", loginPath));

context.Response.Write(obj.ToString(Formatting.Indented));

}

else

{

var user = \_userProvider.GetUser();

if (user != null) context.Response.Write(user.UserName);

}

}

}

}

## 

## Текст класу ExcelHandler

using System;

using System.IO;

using System.Text;

using System.Web;

using System.Web.UI.WebControls;

using System.Xml;

using AskvpDeskTop.Model;

using Microsoft.SharePoint.WebPartPages;

namespace AskvpDeskTop.HttpHandler

{

public class ExcelHandler : IHandler

{

public void Process(HttpContext context)

{

var sprCode = context.Request.QueryString["id\_spr"];

var tabSessionID = context.Request.QueryString["tsid"];

if (string.IsNullOrEmpty(sprCode) || string.IsNullOrEmpty(tabSessionID)) throw new ArgumentException("SprCode or TabSessionID is null","SprCode,TabSessionID");

var sessionVal = context.Session[StaticConfiguration.CreateSessionKey(StaticConfiguration.ExcelSessionToken,sprCode,tabSessionID)];

if (sessionVal == null) throw new Exception("Excel : Session is null");

context.Response.Clear();

//чтобы открыть Эксель-файл

context.Response.ContentType = "application/vnd.ms-excel";

context.Response.AddHeader("Content-Disposition",

"attachment; filename=ExcelFile" + sprCode + ".xls");

context.Response.ContentEncoding = Encoding.Unicode;

context.Response.BinaryWrite(Encoding.Unicode.GetPreamble());

var stringWriter = MainProcess(sessionVal.ToString());

context.Response.ContentEncoding = stringWriter.Encoding;

var clearedXML = clearResultExcelXML(stringWriter.ToString());

//это нужно, чтобы открылся переданный хмл

context.Response.Write(clearedXML);

}

private static string clearResultExcelXML(string html)

{

XmlDocument doc = new XmlDocument();

var br = new StringBuilder();

doc.LoadXml(html.Replace("&nbsp;", "&amp;nbsp;"));

var node = doc.SelectSingleNode(@"/table/tr[2]/td[7]");

br.Append("<html><table cellspacing=\"0\" rules=\"all\" border=\"1\" style=\"border-collapse:collapse;\"><tr>");

br.Append(node.OuterXml);

br.Append("</tr></table></html>");

return br.ToString();

}

private static StringWriter MainProcess(string sessionValue)

{

var html = sessionValue.Replace("&nbsp;", " ");

var br = new StringBuilder();

br.Append("<html>");

br.Append(html);

br.Append("</html>");

html = br.ToString();

// html = html.Substring(html.ToLower().IndexOf("<html"));

//убираем ссылку <LINK REL="STYLESHEET" TYPE="text/css" HREF="Report.css" />

var link\_i1 = html.ToLower().IndexOf("<link");

if (link\_i1 != -1)

{

var link\_i2 = html.ToLower().IndexOf(">", link\_i1);

var link\_str = html.Substring(link\_i1, link\_i2 - link\_i1 + 1);

html = html.Replace(link\_str, "");

}

//готовим потоки

var stringWriter = new StringWriter();

var hw = new System.Web.UI.HtmlTextWriter(stringWriter);

//это переработанный хтмл в датагриде

var xx = new XmlDocument();

xx.LoadXml(html);

var dg = new DataGrid()

{

Visible = true,

DataSource = xx,

};

dg.DataBind();

var t = dg.Items;

dg.RenderControl(hw);

return stringWriter;

}

}

}

## 

## Текст класу ConfigurationProvider

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Security.Policy;

using System.Text;

using AskvpDeskTop.Model;

using Microsoft.SharePoint;

using Microsoft.SharePoint.Linq;

namespace AskvpDeskTop.Provider

{

public class ConfigurationProvider : IConfigurationProvider

{

public DynamicConfiguration LoadConfiguration()

{

var dynamicConfigurationItem = new DynamicConfiguration();

SPSecurity.RunWithElevatedPrivileges(delegate()

{

using (

var askvDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

EntityList<ConfigPortalЭлемент> configPortalItems =

askvDesktopListsDataContext.GetList<ConfigPortalЭлемент>(StaticConfiguration.ConfigPortal);

const string DasUrlItemName = "UrlDas";

const string PortalUrlItemName = "UrlPortal";

const string XSLUrlItemName = "UrlXSL";

const string serverServiceReportItemName = "ServerServiceReport";

const string EmailAddress = "EmailAddress";

const string SMTPserver = "SMTPserver";

var configItemsName = new List<string>()

{

DasUrlItemName,

PortalUrlItemName,

XSLUrlItemName,

serverServiceReportItemName,

EmailAddress,

SMTPserver

};

var configItems = (from el in configPortalItems

where

configItemsName.Contains(el.NamePar)

select new

{

el.NamePar,

el.Value

}

);

foreach (var configItem in configItems)

{

switch (configItem.NamePar)

{

case DasUrlItemName:

dynamicConfigurationItem.DasUrl = configItem.Value;

break;

case PortalUrlItemName:

dynamicConfigurationItem.PortalUrl = configItem.Value;

break;

case XSLUrlItemName:

dynamicConfigurationItem.XSLUrl = configItem.Value;

break;

case serverServiceReportItemName:

dynamicConfigurationItem.ServerServiceUrl = configItem.Value;

break;

case SMTPserver:

dynamicConfigurationItem.SMTPserver = configItem.Value;

break;

case EmailAddress:

dynamicConfigurationItem.EmailAddress = configItem.Value;

break;

}

}

}

});

return dynamicConfigurationItem;

}

}

}

## 

## Текст класу FindProvider

using System.Collections.Generic;

using System.Linq;

using AskvpDeskTop.Model;

using Microsoft.SharePoint;

namespace AskvpDeskTop.Provider.Real

{

public class FindProvider : IFindProvider

{

public IEnumerable<FindItem> Find(string searchPattern, bool personalized, int userId)

{

if (string.IsNullOrEmpty(searchPattern)) return new List<FindItem>();

using (

var askvDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var sprListSiteItems =

askvDesktopListsDataContext.GetList<SprListSiteЭлемент>(StaticConfiguration.SprListSite);

var askvpDesktopUserConfigListItems =

askvDesktopListsDataContext.GetList<AskvpDesktopUserConfigListItem>(StaticConfiguration.AskvpDesktopUserConfig);

var personalListSiteItems = new List<FindItem>();

SPSecurity.RunWithElevatedPrivileges(delegate

{

if (askvpDesktopUserConfigListItems.Any())

{

personalListSiteItems = (from o in askvpDesktopUserConfigListItems

where (o.Sprcode.Kod + " " + o.Sprcode.TitleSpr).ToLower().Contains(searchPattern.ToLower()) && o.UserId == userId

select new FindItem

{

Title = o.Sprcode.TitleSpr,

ID = o.Sprcode.Kod,

Personalized = true

}

).Distinct().OrderBy(t => t.ID).ToList();

}

});

if (personalized) return personalListSiteItems;

var globalsprListSiteItems = new List<FindItem>();

if (sprListSiteItems.ScopeToFolder("Lists/" + StaticConfiguration.SprListSite, true).Any())

{

globalsprListSiteItems = (from o in sprListSiteItems.ScopeToFolder("Lists/" + StaticConfiguration.SprListSite, true)

where (o.Kod.Kod + " " + o.Kod.TitleSpr).ToLower().Contains(searchPattern.ToLower())

select new FindItem

{

Title = o.Kod.TitleSpr,

ID = o.Kod.Kod,

Personalized = false

}

).Distinct().OrderBy(t => t.ID).ToList();

}

return !personalListSiteItems.Any() ? globalsprListSiteItems : personalListSiteItems.Union(globalsprListSiteItems);

}

}

}

}

## 

## Текст класу MenuProvider

using AskvpDeskTop.Model;

using System.Collections.Generic;

using System.Linq;

using Microsoft.SharePoint;

using Microsoft.SharePoint.Linq;

namespace AskvpDeskTop.Provider.Real

{

public class MenuProvider : IMenuProvider

{

public IEnumerable<MenuItem> GetMenu(int folderId, int userId)

{

var resultMenuItems = new List<MenuItem>();

if (folderId < 0) return resultMenuItems;

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

EntityList<SprListSiteFolderItem> sprListSiteFolderItems =

askvpDesktopListsDataContext.GetList<SprListSiteFolderItem>(StaticConfiguration.SprListSite);

EntityList<SprListSiteЭлемент> sprListSiteItems =

askvpDesktopListsDataContext.GetList<SprListSiteЭлемент>(StaticConfiguration.SprListSite);

#region Fix\_Do\_Not\_Delete

EntityList<ReportsLibraryЭлемент> reportsLibraryItems =

askvpDesktopListsDataContext.GetList<ReportsLibraryЭлемент>(StaticConfiguration.ReportsLibrary);

#endregion

var menuFolderItems = new List<MenuItem>();

var menuItems = new List<MenuItem>();

if (folderId != 0)

{

var result = sprListSiteFolderItems.ScopeToFolder("Lists/" + StaticConfiguration.SprListSite, true)

.Where(f => f.Id == folderId)

.Select(f => f.Path + "/" + f.Title);

if (result.Any())

{

var folderPath = result.First();

var folderQueryable = sprListSiteFolderItems.ScopeToFolder(

folderPath, false);

var menuItemsQueryable =

sprListSiteItems.ScopeToFolder(folderPath,

false);

if (folderQueryable.Any())

{

menuFolderItems = (from o in folderQueryable

where o.FolderChildCountId.HasValue

select new MenuItem

{

Title = o.Title,

ID = o.Id.ToString(),

ItemType = MenuItemType.Folder

}).ToList();

}

if (menuItemsQueryable.Any())

{

menuItems = (from o in menuItemsQueryable

where o.Kod != null

select new MenuItem

{

Title = o.Kod.TitleSpr,

ID = o.Kod.Kod,

ItemType = MenuItemType.Link

}).OrderBy(o => o.Title).ToList();

}

}

}

else

{

menuFolderItems = (from o in sprListSiteFolderItems

where o.FolderChildCountId.HasValue

select new MenuItem

{

Title = o.Title,

ID = o.Id.ToString(),

ItemType = MenuItemType.Folder

}).ToList();

if (sprListSiteItems.Any())

{

menuItems = (from o in sprListSiteItems

where o.Kod != null

select new MenuItem

{

Title = o.Kod.TitleSpr,

ID = o.Kod.Kod,

ItemType = MenuItemType.Link

}).OrderBy(o => o.Title).ToList();

}

}

if (menuFolderItems.Any()) resultMenuItems = menuFolderItems;

if (!menuItems.Any()) return resultMenuItems;

EntityList<AskvpDesktopUserConfigListItem> askvpDesktopUserConfigListItems =

askvpDesktopListsDataContext.GetList<AskvpDesktopUserConfigListItem>(

StaticConfiguration.AskvpDesktopUserConfig);

var personalListSiteItems = new List<MenuItem>();

if (askvpDesktopUserConfigListItems.Any())

{

personalListSiteItems = (from o in askvpDesktopUserConfigListItems

where menuItems.Select(t => t.ID).Contains(o.Sprcode.Kod)

&& o.UserId == userId

select new MenuItem

{

Title = o.Sprcode.TitleSpr,

ID = o.Sprcode.Kod,

Personalized = true,

ItemType = MenuItemType.Link

}

).OrderBy(t => t.ID).ToList();

}

if (personalListSiteItems.Any())

menuItems = personalListSiteItems.Union(menuItems).ToList();

resultMenuItems = resultMenuItems.Union(menuItems).ToList();

}

return resultMenuItems;

}

public IEnumerable<MenuItem> GetPersonalMenu(int userId)

{

var userConfigCol = new List<MenuItem>();

SPSecurity.RunWithElevatedPrivileges(delegate

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

EntityList<AskvpDesktopUserConfigListItem> askvpDesktopUserConfigListItems =

askvpDesktopListsDataContext.GetList<AskvpDesktopUserConfigListItem>(

StaticConfiguration.AskvpDesktopUserConfig);

userConfigCol = (from o in askvpDesktopUserConfigListItems

where o.UserId == userId

select new MenuItem

{

ID = o.Sprcode.Kod,

Title = o.Sprcode.TitleSpr,

Personalized = true,

ItemType = MenuItemType.Link

}).ToList();

}

});

return userConfigCol;

}

}

}

## 

## Текст класу QProvider

using AccessDAS;

using AskvpDeskTop.Model;

using System;

using System.Collections.Generic;

using System.Linq;

namespace AskvpDeskTop.Provider.Real

{

public class QProvider: IQProvider

{

private readonly IConfigurationProvider \_configurationProvider;

public QProvider(IConfigurationProvider configurationProvider)

{

\_configurationProvider = configurationProvider;

}

public List<DefaultItem> GetData(QueryModel queryModel)

{

var configuration = \_configurationProvider.LoadConfiguration();

var defaultItems = new List<DefaultItem>();

if(string.IsNullOrEmpty(configuration.DasUrl)) throw new Exception("Invalid URL DAS : " + configuration.DasUrl);

var accessDas = new AccessClass(configuration.DasUrl);

var hasData = queryModel.QueryData.Any();

var hasArrayInQueryData = hasData && queryModel.QueryData.Any(el => el.DefaultItems.Count > 1);

accessDas.SetReportName(queryModel.AffectDasMethod ??

(hasArrayInQueryData ? "param\_array\_" : "param\_") + queryModel.QueryName);

if (hasData) SetAccessClassParams(accessDas, queryModel);

if (!accessDas.Execute()) throw new Exception("DAS : " + accessDas.GetErrorMessage());

var ds = accessDas.GetTable("Data");

if (ds == null || ds.Rows.Count <= 0) return defaultItems;

for (var i = 0; i < ds.Rows.Count; i++)

{

var value = ds.Rows[i]["VALUE"].ToString();

//если в запросе только одно поле - ЗНАЧЕНИЕ - его же записываем и в ТЕКСТ

var name = ds.Columns.IndexOf("NAME") >= 0 ? ds.Rows[i]["NAME"].ToString() : value;

var tempDefaultItem = new DefaultItem {Value = value, NValue = name};

// проверяем на наличии одинаковых элементов

if (!defaultItems.Contains(tempDefaultItem))

defaultItems.Add(tempDefaultItem);

}

return defaultItems;

}

private static void SetAccessClassParams(AccessClass accessDas, QueryModel queryModel)

{

foreach (var queryDataItem in queryModel.QueryData)

{

/\* Array? \*/

if (queryDataItem.DefaultItems.Count > 1)

{

#region Array

for (var i = 0; i < queryDataItem.DefaultItems.Count; i++)

{

switch (queryDataItem.DasType)

{

case DasTypes.Date:

{

if (queryModel.DasNamedParams)

{

queryDataItem.DefaultItems.ForEach(

el =>

accessDas.AddArrayParam(queryDataItem.ParametrName,

DateTime.Parse(el.Value)));

}

else

{

queryDataItem.DefaultItems.ForEach(

el => accessDas.AddArrayParam(i, DateTime.Parse(el.Value)));

}

}

break;

case DasTypes.Integer:

{

if (queryModel.DasNamedParams)

{

queryDataItem.DefaultItems.ForEach(

el =>

accessDas.AddArrayParam(queryDataItem.ParametrName,

int.Parse(el.Value)));

}

else

{

queryDataItem.DefaultItems.ForEach(

el => accessDas.AddArrayParam(i, int.Parse(el.Value)));

}

}

break;

default:

{

if (queryModel.DasNamedParams)

queryDataItem.DefaultItems.ForEach(

el => accessDas.AddArrayParam(queryDataItem.ParametrName, el.Value));

else

{

queryDataItem.DefaultItems.ForEach(

el => accessDas.AddArrayParam(i, el.Value));

}

}

break;

}

}

#endregion

}

else

{

#region not Array

var defaultItem = queryDataItem.DefaultItems.ElementAt(0);

switch (queryDataItem.DasType)

{

case DasTypes.Date:

{

if (queryModel.DasNamedParams)

{

accessDas.AddParam(queryDataItem.ParametrName, DateTime.Parse(defaultItem.Value));

}

else

{

accessDas.AddParam(DateTime.Parse(defaultItem.Value));

}

}

break;

case DasTypes.Integer:

{

if (queryModel.DasNamedParams)

{

accessDas.AddParam(queryDataItem.ParametrName, int.Parse(defaultItem.Value));

}

else

{

accessDas.AddParam(int.Parse(defaultItem.Value));

}

}

break;

default:

{

if (queryModel.DasNamedParams)

accessDas.AddParam(queryDataItem.ParametrName, defaultItem.Value);

else

{

accessDas.AddParam(int.Parse(defaultItem.Value));

}

}

break;

}

#endregion

}

}

}

}

}

## 

## Текст класу DataProvider

using System.Web.UI.WebControls;

using AskvpDeskTop.Model;

using System;

using System.Web;

using System.Collections.Generic;

using System.Linq;

using Microsoft.SharePoint;

using Newtonsoft.Json;

namespace AskvpDeskTop.Provider.Real

{

public class DataProvider : IDataProvider

{

private readonly IUserProvider \_userProvider;

private readonly SprFilter \_formFilter;

public DataProvider(IUserProvider userProvider, SprFilter formFilter)

{

\_userProvider = userProvider;

\_formFilter = formFilter;

}

public SprItem CreateForm(string sprCode)

{

SprItem sprItem = null;

if (!string.IsNullOrEmpty(sprCode))

{

SPSecurity.RunWithElevatedPrivileges(delegate

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var paramSprListItems =

askvpDesktopListsDataContext.GetList<ParamSprListЭлемент>(StaticConfiguration.ParamSprList);

var allParamsListItems =

askvpDesktopListsDataContext.GetList<AllParamsListЭлемент>(StaticConfiguration.AllParamsList);

var reportsLibraryItems =

askvpDesktopListsDataContext.GetList<ReportsLibraryЭлемент>(

StaticConfiguration.ReportsLibrary);

var reportLibraryItem = (from o in reportsLibraryItems

where o.Kod == sprCode

select new

{

Title = o.TitleSpr,

SprUrl = o.Spr\_url,

XSLTSpr = o.Xslt\_spr,

}).First();

var paramSprListCol = (from o in paramSprListItems

where o.Code.Kod == sprCode

select new

{

Ord = o.Ord ?? 0,

ParametrName = o.Name\_par.Name\_par,

KeyName = o.Name\_par.Key\_name,

Description = o.Other\_description,

DefaultValue =

o.Other\_default != null

? new List<DefaultItem> { new DefaultItem { Value = o.Other\_default } }

: null,

CharacterLength = o.Other\_length,

ControlWidth = o.Other\_width\_par,

MultiSelect = o.Multiselect ?? false,

ListSource =

o.List\_source != null

? ParseListSourceField(o.List\_source)

: new List<DefaultItem>(),

NextRow = o.Next\_row ?? false,

TextRowBefore = o.Text\_row\_before,

TextRowAfter = o.Text\_row\_after,

AffectDasMethod = o.Affect\_das\_method,

AffectSource =

o.Affect\_source != null

? ParseListSourceField(o.Affect\_source)

: new List<DefaultItem>(),

RadioGroup = ConvertToRadioGroups(o.Radio\_group),

DescriptionAlign = ConvertToAlign(o.Align\_description),

ParametrAlign = ConvertToAlign(o.Align\_par),

AffectSourceObject = ConvertToAffectSourceObjects(o.Affect\_source\_object),

DependAction = ConvertToDependActions(o.Depend\_action),

AffectNameParametr = o.Affect\_name\_par.Name\_par,

DescriptionGridMobile = o.Descr\_grid\_m ?? 12,

DescriptionGridMobileOffset = o.Descr\_grid\_m\_offset ?? 0,

DescriptionGridDesktop = o.Descr\_grid\_d ?? 2,

DescriptionGridDesktopOffset = o.Descr\_grid\_d\_offset ?? 0,

ParametrGridMobile = o.Param\_grid\_m ?? 12,

ParametrGridMobileOffset = o.Param\_grid\_m\_offset ?? 0,

ParametrGridDesktop = o.Param\_grid\_d ?? 3,

ParametrGridDesktopOffset = o.Param\_grid\_d\_offset ?? 0,

SprFilteringAction = ConvertToSprFilteringAction(o.Filtering),

}).ToList();

var allParamsListCol = (from o in allParamsListItems

where

paramSprListCol.Select(t => t.ParametrName).Union(

paramSprListCol.Select(t => t.AffectNameParametr))

.Contains(o.Name\_par)

select new

{

ParametrName = o.Name\_par,

o.Description,

DefaultValue =

o.Default != null

? new List<DefaultItem> { new DefaultItem { Value = o.Default } }

: null,

DasNamedParams = o.Das\_named\_params ?? false,

DependOn = o.DependOn != null ? ParseDependOnField(o.DependOn) : new List<string>(),

CharacterLength = o.Length != null ? Convert.ToDouble(o.Length) : 0,

ControlWidth = o.Width\_par != null ? Convert.ToDouble(o.Width\_par) : 0,

TextRows = o.Text\_rows != null ? o.Text\_rows.Value : 0,

SourceObject = ConvertToSourceObjects(o.Source\_object),

TypeParameter = ConvertToTypeParameters(o.Type\_par),

DasType = ConvertToDasTypes(o.Das\_type),

TextMode = ConvertToTextModes(o.Text\_mode),

ListSource =

o.Source != null ? ParseListSourceField(o.Source) : new List<DefaultItem>(),

FilteringAction = ConvertToFilteringAction(o.Filtering),

}).ToList();

var formItems = (from m in paramSprListCol

join k in allParamsListCol on m.ParametrName equals k.ParametrName

select new FormItem

{

Ord = m.Ord,

ParametrName = m.ParametrName,

KeyName = m.KeyName,

Description = m.Description ?? k.Description,

DefaultValue = m.DefaultValue ?? k.DefaultValue,

CharacterLength = m.CharacterLength ?? k.CharacterLength,

ControlWidth = m.ControlWidth ?? k.ControlWidth,

MultiSelect = m.MultiSelect,

Data = m.ListSource.Count > 0 ? m.ListSource : k.ListSource,

NextRow = m.NextRow,

TextRowBefore = m.TextRowBefore,

TextRowAfter = m.TextRowAfter,

AffectDasMethod = m.AffectDasMethod,

AffectSource = m.AffectSource,

RadioGroup = m.RadioGroup,

DescriptionAlign = m.DescriptionAlign,

ParametrAlign = m.ParametrAlign,

AffectSourceObject = m.AffectSourceObject,

SourceObject = k.SourceObject,

DependAction = m.DependAction,

DependOn = k.DependOn,

TextRows = k.TextRows,

DasType = k.DasType,

TextMode = k.TextMode,

TypeParameter = k.TypeParameter,

AffectNameParametr = m.AffectNameParametr,

DescriptionGridMobile = Convert.ToInt32(m.DescriptionGridMobile),

DescriptionGridMobileOffset = Convert.ToInt32(m.DescriptionGridMobileOffset),

DescriptionGridDesktop = Convert.ToInt32(m.DescriptionGridDesktop),

DescriptionGridDesktopOffset = Convert.ToInt32(m.DescriptionGridDesktopOffset),

ParametrGridMobile = Convert.ToInt32(m.ParametrGridMobile),

ParametrGridMobileOffset = Convert.ToInt32(m.ParametrGridMobileOffset),

ParametrGridDesktop = Convert.ToInt32(m.ParametrGridDesktop),

ParametrGridDesktopOffset = Convert.ToInt32(m.ParametrGridDesktopOffset),

FilteringAction = k.FilteringAction,

SprFilteringAction = m.SprFilteringAction,

}

).OrderBy(item => item.Ord).ToList();

\_formFilter.FilterFormItems(formItems);

sprItem = new SprItem

{

FormItems = formItems,

Title = reportLibraryItem.Title,

SprUrl = reportLibraryItem.SprUrl,

XSLTSpr = reportLibraryItem.XSLTSpr

};

}

});

}

return sprItem;

}

public List<SimpleFormItem> FindParamInfo(string sprCode, IEnumerable<string> paramCollection)

{

var simpleFormItems = new List<SimpleFormItem>();

if (string.IsNullOrEmpty(sprCode) || paramCollection == null) return simpleFormItems;

var collection = paramCollection as IList<string> ?? paramCollection.ToList();

SPSecurity.RunWithElevatedPrivileges(delegate

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var allParamsListItems =

askvpDesktopListsDataContext.GetList<AllParamsListЭлемент>(StaticConfiguration.AllParamsList);

var paramSprListItems =

askvpDesktopListsDataContext.GetList<ParamSprListЭлемент>(StaticConfiguration.ParamSprList);

var paramSprListSimpleFormItems = (from o in paramSprListItems

where o.Code.Kod == sprCode && collection.Contains(o.Name\_par.Name\_par)

select new SimpleFormItem

{

ParametrName = o.Name\_par.Name\_par,

Description = o.Other\_description

}).ToList();

var tempParamSprListSimpleFormItems =

collection.Except(paramSprListSimpleFormItems.Where(el => !string.IsNullOrEmpty(el.Description)).Select(el => el.ParametrName).ToList());

var formItems = tempParamSprListSimpleFormItems as IList<string> ?? tempParamSprListSimpleFormItems.ToList();

if (formItems.Any())

{

simpleFormItems = (from o in allParamsListItems

where formItems.Contains(o.Name\_par)

select new SimpleFormItem

{

ParametrName = o.Name\_par,

Description = o.Description,

}).ToList();

}

simpleFormItems.AddRange(paramSprListSimpleFormItems.Where(el => !string.IsNullOrEmpty(el.Description)));

}

});

return simpleFormItems;

}

public List<SimpleFormItem> GetUserConfig(string sprCode, int userId)

{

List<SimpleFormItem> simpleFormItems = null;

SPSecurity.RunWithElevatedPrivileges(delegate

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var askvpDesktopUserConfigListItems =

askvpDesktopListsDataContext.GetList<AskvpDesktopUserConfigListItem>(StaticConfiguration.AskvpDesktopUserConfig);

var userConfigCol = (from o in askvpDesktopUserConfigListItems

where o.Sprcode.Kod == sprCode && o.UserId == userId

select o.Config).Take(1).ToList();

simpleFormItems = userConfigCol.Count > 0

? JsonConvert.DeserializeObject<List<SimpleFormItem>>(userConfigCol[0])

: new List<SimpleFormItem>();

}

});

return simpleFormItems;

}

public void UpdateUserConfig(string sprCode, int userId, string simpleFormItems)

{

SharePointHelper.RunWithElevatedPrivilegesAndContextSwitch(delegate

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var askvpDesktopUserConfigListItems =

askvpDesktopListsDataContext.GetList<AskvpDesktopUserConfigListItem>(StaticConfiguration.AskvpDesktopUserConfig);

var userConfigCol = (from o in askvpDesktopUserConfigListItems

where o.Sprcode.Kod == sprCode && o.UserId == userId

select o).Take(1).ToList();

if (userConfigCol.Count > 0)

{

userConfigCol[0].Config = simpleFormItems;

}

else

{

var reportsLibraryItems =

askvpDesktopListsDataContext.GetList<ReportsLibraryЭлемент>(

StaticConfiguration.ReportsLibrary);

var reportsLibraryItem = (from o in reportsLibraryItems

where o.Kod == sprCode

select o);

var configUserItem = new AskvpDesktopUserConfigListItem

{

UserId = userId,

Config = simpleFormItems,

Sprcode = reportsLibraryItem.First()

};

askvpDesktopListsDataContext.AskvpDesktopUserConfig.InsertOnSubmit(configUserItem);

}

askvpDesktopListsDataContext.SubmitChanges();

}

});

}

public string GetXSLT(string xsltName)

{

string xslt = null;

if (!string.IsNullOrEmpty(xsltName))

{

SPSecurity.RunWithElevatedPrivileges(delegate

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var styleSiteListItems =

askvpDesktopListsDataContext.GetList<StyleSiteListЭлемент>(StaticConfiguration.StyleSiteList);

var sourceStyleSiteItems =

styleSiteListItems.Where(el => el.NameStyle == xsltName).Select(el => el.SourceStyle);

if (sourceStyleSiteItems.Any())

{

xslt = sourceStyleSiteItems.ElementAt(0);

}

}

});

}

return xslt;

}

public List<LinkListModel> GetLinkListItems(string sprCode)

{

var linkSprListCol = new List<LinkListModel>();

if (!string.IsNullOrEmpty(sprCode))

{

SPSecurity.RunWithElevatedPrivileges(delegate

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var linkSprListItems =

askvpDesktopListsDataContext.GetList<LinkSprListЭлемент>(StaticConfiguration.LinkSprList);

linkSprListCol = (from o in linkSprListItems

where o.Code.Kod == sprCode

select new LinkListModel

{

LinkSpr = o.Link\_spr.Kod,

Description = o.Description,

KeyLink = o.Key\_link,

TCOL = o.TCOL != null ? Convert.ToInt32(o.TCOL) : 0,

TROW = o.TROW != null ? Convert.ToInt32(o.TROW) : 0,

OffsetTop = o.OffsetTop != null ? Convert.ToInt32(o.OffsetTop) : 0,

OffsetBottom = o.OffsetBottom != null ? Convert.ToInt32(o.OffsetBottom) : 0,

IncludeHead = o.IncludeHead != null && o.IncludeHead.Value

}).ToList();

}

});

}

return linkSprListCol;

}

private static List<string> ParseDependOnField(string dependOnField)

{

var dependOnFieldsList = new List<string>();

var dependSource = dependOnField.Replace(" ", "");

var dependTmp = "";

foreach (var t in dependSource)

{

if (t != '\r' && t != '\n')

dependTmp += t;

else

{

if (t == '\n') continue;

dependOnFieldsList.Add(dependTmp);

dependTmp = "";

}

}

if (dependTmp != "")

dependOnFieldsList.Add(dependTmp);

return dependOnFieldsList;

}

private static List<DefaultItem> ParseListSourceField(string listSourceField)

{

//разбираем в массивы имён и значений:

//@ - разделяет значение и имя, \n\r - разделяет элементы списка

//если @ нету - имя = значение

var listSourceFieldList = new List<DefaultItem>();

var numEl = 1 + listSourceField.Count(t => t == '\n'); //после последней строки перевода нету

var sName = new string[numEl];

var sValue = new string[numEl];

var flagNv = 0;

var j = 0;

foreach (var listSourceItem in listSourceField)

{

switch (listSourceItem)

{

case '@':

flagNv = 1;

break;

case '\n':

if (flagNv == 1)

{

/\*if (sValue[j] != "" && string.IsNullOrEmpty(sName[j]))

sName[j] = sValue[j];\*/

flagNv = 0;

j++;

}

break;

case '\r':

/\*if (sValue[j] != "" && string.IsNullOrEmpty(sName[j]))

sName[j] = sValue[j];\*/

flagNv = 0;

j++;

break;

default:

if (flagNv == 0)

sValue[j] += listSourceItem;

else

sName[j] += listSourceItem;

break;

}

}

for (int i = 0; i < sValue.Length; i++)

{

if (sValue[i] != null)

listSourceFieldList.Add(new DefaultItem { Value = sValue[i], NValue = sName[i] });

}

return listSourceFieldList;

}

private static SourceObjects ConvertToSourceObjects(Source\_object? sourceObject)

{

switch (sourceObject)

{

case Source\_object.None0:

return SourceObjects.None;

case Source\_object.DAS:

return SourceObjects.DAS;

case Source\_object.LIST:

return SourceObjects.LIST;

case Source\_object.DateTimeLIST:

return SourceObjects.DateTimeLIST;

default:

return SourceObjects.None;

}

}

private static TypeParameters ConvertToTypeParameters(Type\_par? typePar)

{

switch (typePar)

{

case Type\_par.DropDownList:

return TypeParameters.DropDownList;

case Type\_par.ListBox:

return TypeParameters.ListBox;

case Type\_par.TextBox:

return TypeParameters.TextBox;

case Type\_par.Calendar:

return TypeParameters.Calendar;

case Type\_par.CalendarWithoutTime:

return TypeParameters.CalendarWithoutTime;

case Type\_par.TimeBox:

return TypeParameters.TimeBox;

case Type\_par.RadioButton:

return TypeParameters.RadioButton;

case Type\_par.CheckBox:

return TypeParameters.CheckBox;

case Type\_par.AutoComplete:

return TypeParameters.AutoComplete;

default:

return TypeParameters.DropDownList;

}

}

private static DasTypes ConvertToDasTypes(Das\_type? dasType)

{

switch (dasType)

{

case Das\_type.Integer:

return DasTypes.Integer;

case Das\_type.String:

return DasTypes.String;

case Das\_type.Date:

return DasTypes.Date;

default:

return DasTypes.Integer;

}

}

private static FilteringAction ConvertToFilteringAction(Filtering? filteringAction)

{

switch (filteringAction)

{

case Filtering.RestrictDor:

return FilteringAction.RestrictDor;

case Filtering.CheckStanByDor:

return FilteringAction.CheckStanByDor;

case Filtering.CheckIndexStanByDor:

return FilteringAction.CheckIndexStanByDor;

default:

return FilteringAction.None;

}

}

private static SprFilteringAction ConvertToSprFilteringAction(Filtering0? filteringAction)

{

switch (filteringAction)

{

case Filtering0.CheckAtLeastOneDor:

return SprFilteringAction.CheckAtLeastOneDor;

default:

return SprFilteringAction.None;

}

}

private static TextModes ConvertToTextModes(Text\_mode? textMode)

{

switch (textMode)

{

case Text\_mode.SingleLine:

return TextModes.SingleLine;

case Text\_mode.MultiLine:

return TextModes.MultiLine;

default:

return TextModes.SingleLine;

}

}

private static string ConvertToRadioGroups(Radio\_group? radioGroup)

{

switch (radioGroup)

{

case Radio\_group.Нет:

return null;

case Radio\_group.\_1:

return "1";

case Radio\_group.\_2:

return "2";

case Radio\_group.\_3:

return "3";

case Radio\_group.\_4:

return "4";

case Radio\_group.\_5:

return "5";

case Radio\_group.\_6:

return "6";

case Radio\_group.\_7:

return "7";

case Radio\_group.\_8:

return "8";

case Radio\_group.\_9:

return "9";

case Radio\_group.\_10:

return "10";

default:

return null;

}

}

private static Align ConvertToAlign(Align\_par? alignPar)

{

switch (alignPar)

{

case Align\_par.Left:

return Align.Left;

case Align\_par.Right:

return Align.Right;

case Align\_par.Center:

return Align.Center;

default:

return Align.Left;

}

}

private static Align ConvertToAlign(Align\_description? alignPar)

{

switch (alignPar)

{

case Align\_description.Left:

return Align.Left;

case Align\_description.Right:

return Align.Right;

case Align\_description.Center:

return Align.Center;

default:

return Align.Left;

}

}

private static AffectSourceObjects ConvertToAffectSourceObjects(Affect\_source\_object? affectSourceObject)

{

switch (affectSourceObject)

{

case Affect\_source\_object.None0:

return AffectSourceObjects.None;

case Affect\_source\_object.DAS:

return AffectSourceObjects.DAS;

case Affect\_source\_object.DateTimeLIST:

return AffectSourceObjects.DateTimeList;

case Affect\_source\_object.LIST:

return AffectSourceObjects.List;

default:

return AffectSourceObjects.None;

}

}

private static DependActions ConvertToDependActions(Depend\_action? dependAction)

{

switch (dependAction)

{

case Depend\_action.None0:

return DependActions.None;

case Depend\_action.Disable:

return DependActions.Disable;

case Depend\_action.Enable:

return DependActions.Enable;

default:

return DependActions.None;

}

}

}

}

## 

## Текст класу QueryServiceDataProvider

using System.Collections.Generic;

using System.Linq;

using System.Security;

using System.Text;

using AskvpDeskTop.Exceptions;

using AskvpDeskTop.Model;

using GeneralLibrary;

using GetQueryServiceDB;

namespace AskvpDeskTop.Provider.Real

{

public class QueryServiceDataProvider: IQueryServiceDataProvider

{

public string GetData(string sprCode,string sprTitle, string serviceUrl, IEnumerable<SimpleFormItem> selectedFormItems,

DynamicConfiguration configuration)

{

var inputXml = CreateInputXml(sprCode, sprTitle, selectedFormItems);

var getQuery = new GetQueryServiceDBClass();

ErrorClass errorClass;

var queryResult = getQuery.GetData(inputXml, serviceUrl, configuration.DasUrl, out errorClass);

if(errorClass.IsException)

throw new QueryServiceDataException(sprCode, serviceUrl, inputXml,queryResult,null,null);

return queryResult;

}

private static string CreateInputXml(string sprCode, string sprTitle, IEnumerable<SimpleFormItem> simpleFormItems)

{

var strB = new StringBuilder();

strB.Append("<?xml version='1.0' encoding='windows-1251' standalone='yes'?>");

strB.Append("<UZ-XDOC schema=\"askvpuze\" doc\_version=\"2.0\" doc\_type=\"query\" >");

strB.Append("<HEAD messcode=\"" + sprCode + "\" from=\"change\" password=\"tiger\" title=\"" + SecurityElement.Escape(sprTitle) +

"\" />");

strB.Append("<BODY>");

strB.Append("<PARAMETERS>");

foreach (var simpleFormItem in simpleFormItems)

{

strB.Append("<PARAMETER");

strB.Append(" name=\"" + simpleFormItem.ParametrName + "\"");

strB.Append(" description=\"" + SecurityElement.Escape(simpleFormItem.Description) + "\"");

string nValue;

string value;

if (simpleFormItem.DefaultValue.Count > 1)

{

value = string.Join("@@", simpleFormItem.DefaultValue.Select(el => el.Value).ToArray());

nValue = string.Join(";", simpleFormItem.DefaultValue.Select(el => SecurityElement.Escape(el.NValue)).ToArray());

}

else

{

if (simpleFormItem.DefaultValue.Count != 0)

{

var defaultItem = simpleFormItem.DefaultValue[0];

value = defaultItem.Value;

nValue = !string.IsNullOrEmpty(defaultItem.NValue) ? SecurityElement.Escape(defaultItem.NValue) : defaultItem.Value;

}

else

{

value = nValue = "";

}

}

strB.Append(" value=\"" + value + "\"");

strB.Append(" nvalue=\"" + nValue + "\"");

strB.Append("/>");

}

strB.Append("</PARAMETERS>\n");

strB.Append("</BODY>\n");

strB.Append("</UZ-XDOC>");

return strB.ToString();

}

}

}

## 

## Текст класу UserProvider

using System;

using AskvpDeskTop.Exceptions;

using AskvpDeskTop.Model;

using Microsoft.Office.Server.UserProfiles;

using Microsoft.SharePoint;

using Microsoft.SharePoint.ApplicationPages.Calendar.Exchange;

using Microsoft.SharePoint.Linq;

using System.Linq;

using System.Web;

using Microsoft.Office.Server;

namespace AskvpDeskTop.Provider.Real

{

public class UserProvider : IUserProvider

{

public bool UserHasPermission(string sprCode)

{

using (

var askvpDesktopListsDataContext =

new AskvDesktopListsDataContext(StaticConfiguration.AskvpDesktopSiteUrl))

{

var isExist = false;

EntityList<SprListSiteЭлемент> sprListSiteItems =

askvpDesktopListsDataContext.GetList<SprListSiteЭлемент>(StaticConfiguration.SprListSite);

SPSecurity.RunWithElevatedPrivileges(delegate()

{

if (!

askvpDesktopListsDataContext.GetList<ReportsLibraryЭлемент>(StaticConfiguration.ReportsLibrary)

.Any(el => el.Kod == sprCode))

throw new SprNotFoundException();

isExist = askvpDesktopListsDataContext.GetList<SprListSiteЭлемент>(StaticConfiguration.SprListSite)

.ScopeToFolder("Lists/" + StaticConfiguration.SprListSite, true)

.Where(t => t.Kod.Kod == sprCode).Select(m => m.Title).ToList().Any();

});

if (isExist)

return sprListSiteItems.ScopeToFolder("Lists/" + StaticConfiguration.SprListSite, true)

.Where(t => t.Kod.Kod == sprCode).Select(m => m.Kod.TitleSpr).Any();

else

{

return true;

}

}

}

private DefaultItem GetUserDor()

{

string kodDor = null, dorName = null;

var userIp = HttpContext.Current.Request.ServerVariables["REMOTE\_ADDR"].ToString();

SPSecurity.RunWithElevatedPrivileges(delegate

{

using (var site = new SPSite(SPContext.Current.Site.ID))

{

var manager = new UserProfileManager(SPServiceContext.GetContext(site));

var profile = manager.GetUserProfile(false);

kodDor = profile["road"].Value != null ? profile["road"].Value.ToString() : null;

}

});

if (String.IsNullOrEmpty(kodDor))

{

if (userIp.StartsWith("10.1"))

{

kodDor = "33";

}

if (userIp.StartsWith("10.2"))

{

kodDor = "32";

}

if (userIp.StartsWith("10.3"))

{

kodDor = "35";

}

if (userIp.StartsWith("10.4"))

{

kodDor = "40";

}

if (userIp.StartsWith("10.5"))

{

kodDor = "43";

}

if (userIp.StartsWith("10.6"))

{

kodDor = "45";

}

if (userIp.StartsWith("10.7"))

{

kodDor = "48";

}

}

switch (kodDor)

{

case "32":

dorName = "Південно-Західна";

break;

case "35":

dorName = "Львівська";

break;

case "40":

dorName = "Одеська";

break;

case "43":

dorName = "Південна";

break;

case "45":

dorName = "Придніпровська";

break;

case "48":

dorName = "Донецька";

break;

case null:

return null;

}

return new DefaultItem { Value = kodDor, NValue = dorName };

}

public UserModel GetUser()

{

var currentSPContext = SPContext.Current;

if (SPContext.Current == null || currentSPContext.Web.CurrentUser == null) return null;

var currentUser = currentSPContext.Web.CurrentUser;

return new UserModel { UserID = currentUser.ID, UserName = currentUser.Name, UserDor = GetUserDor() };

}

}

}

## 

## Текст класу SprItem

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

[Serializable]

public class SprItem

{

public string Title { get; set; }

public string SprUrl { get; set; }

public string XSLTSpr { get; set; }

public List<FormItem> FormItems = new List<FormItem>();

}

}

## 

## Текст класу FormItem

using Newtonsoft.Json;

using Newtonsoft.Json.Converters;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using NLog.Filters;

namespace AskvpDeskTop.Model

{

[Serializable]

public class FormItem : SimpleFormItem

{

/// <summary>

/// Порядковый номер поля

/// </summary>

///

public double Ord;

/// <summary>

/// Ключевое имя параметра (для DAS)

/// </summary>

public string KeyName;

/// <summary>

/// Для ДАСа: если да - все параметры-источники передаются в запрос как именованные, иначе - как неименованные

/// </summary>

public bool DasNamedParams;

/// <summary>

/// Максимальное количество символов

/// </summary>

public double CharacterLength;

/// <summary>

/// Ширина параметра

/// </summary>

public double ControlWidth;

/// <summary>

/// Список параметров, от которых зависит данный параметр

/// </summary>

public List<string> DependOn;

/// <summary>

/// Количество строк в текстовом поле с несколькими строками

/// </summary>

public double TextRows;

/// <summary>

/// Для списков: мультиселект параметра

/// </summary>

public bool MultiSelect;

public List<DefaultItem> Data;

/// <summary>

/// Если ДА, то Параметр переносится на следующую строку

/// </summary>

public bool NextRow;

/// <summary>

/// Текст перед параметром

/// </summary>

public string TextRowBefore;

/// <summary>

/// Текст после параметра

/// </summary>

public string TextRowAfter;

/// <summary>

/// Источник данных

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public SourceObjects SourceObject;

/// <summary>

/// Метод ДАС, формирующий набор значений параметра-родителя, влияющих на видимость текущего параметра

/// </summary>

public string AffectDasMethod;

/// <summary>

/// Список значений, при которых выполняется действие, выбранное в поле depend\_action.

/// </summary>

public List<DefaultItem> AffectSource;

/// <summary>

/// Тип параметра

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public TypeParameters TypeParameter;

/// <summary>

/// Тип параметра для ДАС

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public DasTypes DasType;

/// <summary>

/// Однострочный или многострочный текст

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public TextModes TextMode;

/// <summary>

/// Номер группы - только для радиобаттонов: радиобаттоны одной группы взаимосвязаны между собой

/// </summary>

public string RadioGroup;

/// <summary>

/// Выравнивание подписи параметра

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public Align DescriptionAlign;

/// <summary>

/// Выравнивание параметра

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public Align ParametrAlign;

/// <summary>

/// Источник данных списка значений параметра-родителя, при которых выполняется действие параметра-потомка, выбранное в поле depend\_action:

///ДАС, список значений, список временных выражений

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public AffectSourceObjects AffectSourceObject;

/// <summary>

/// Действие, выполняемое при попадании текущего значения параметра-родителя в список значений из depend\_dource или DAS

/// </summary>

[JsonConverter(typeof(StringEnumConverter))]

public DependActions DependAction;

/// <summary>

/// Имя параметра, от значения которого зависит видимость текущего параметра

/// </summary>

public string AffectNameParametr;

public int DescriptionGridMobile;

public int DescriptionGridMobileOffset;

public int DescriptionGridDesktop;

public int DescriptionGridDesktopOffset;

public int ParametrGridMobile;

public int ParametrGridMobileOffset;

public int ParametrGridDesktop;

public int ParametrGridDesktopOffset;

}

public enum TypeParameters

{

DropDownList,

ListBox,

TextBox,

Calendar,

CalendarWithoutTime,

TimeBox,

RadioButton,

CheckBox,

AutoComplete

}

public enum SourceObjects

{

None,

DAS,

LIST,

DateTimeLIST

}

public enum DasTypes

{

Integer,

String,

Date

}

public enum TextModes

{

SingleLine,

MultiLine

}

public enum Align

{

Left,

Right,

Center

}

public enum AffectSourceObjects

{

None,

DAS,

List,

DateTimeList

}

public enum DependActions

{

None,

Enable,

Disable,

}

}

## 

## Текст класу DefaultItem

using System;

namespace AskvpDeskTop.Model

{

[Serializable]

public class DefaultItem : IEquatable<DefaultItem>

{

public string Value { get; set; }

public string NValue { get; set; }

public bool Equals(DefaultItem y)

{

return Value.Equals(y.Value);

}

public override int GetHashCode()

{

return Value.GetHashCode() + NValue.GetHashCode();

}

public override string ToString()

{

return "Value:" + Value + ";NValue:" + NValue;

}

}

}

## 

## Текст класу QueryModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

public class QueryModel

{

public string QueryName { get; set; }

public string ParametrName { get; set; }

public bool DasNamedParams { get; set; }

public string AffectDasMethod { get; set; }

public string SearchPattern { get; set; }

public readonly List<QueryItem> QueryData = new List<QueryItem>();

}

}

## 

## Текст класу QueryItem

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

public class QueryItem

{

public DasTypes DasType { get; set; }

public string ParametrName { get; set; }

public List<DefaultItem> DefaultItems = new List<DefaultItem>();

}

}

## 

## Текст класу UserModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

public class UserModel

{

public string UserName { get; set; }

public int UserID { get; set; }

public DefaultItem UserDor { get; set; }

}

}

## 

## Текст класу FindItem

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

public class FindItem : IEquatable<FindItem>

{

public string ID { get; set; }

public string Title { get; set; }

public bool Personalized { get; set; }

public string RestrictedDors { get; set; }

public override int GetHashCode()

{

return ID.GetHashCode()\*Title.GetHashCode();

}

public bool Equals(FindItem y)

{

return this.ID.Equals(y.ID);

}

public int GetHashCode(FindItem findItem)

{

return findItem.GetHashCode();

}

}

}

## 

## Текст класу MenuItem

using Newtonsoft.Json;

using Newtonsoft.Json.Converters;

using System;

namespace AskvpDeskTop.Model

{

public class MenuItem : FindItem , IEquatable<MenuItem>

{

[JsonConverter(typeof(StringEnumConverter))]

public MenuItemType ItemType { get; set; }

public override int GetHashCode()

{

return ID.GetHashCode() \* Title.GetHashCode();

}

public bool Equals(MenuItem y)

{

return this.ID.Equals(y.ID);

}

public int GetHashCode(MenuItem menuItem)

{

return menuItem.GetHashCode();

}

}

}

## 

## Текст класу SearchModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

public class SearchModel

{

public string Pattern { get; set; }

public int PageNum { get; set; }

public int ItemsPerPage { get; set; }

public bool Personalized { get; set; }

}

}

## 

## Текст класу StaticConfiguration

using NLog;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

public static class StaticConfiguration

{

/// <summary>

/// URL расположения списков

/// </summary>

public const string AskvpDesktopSiteUrl = "http://localhost/solutions";

/// <summary>

/// Название списка иерархии справок

/// </summary>

public const string SprListSite = "SprListSite";

/// <summary>

/// Название списка справок

/// </summary>

public const string ReportsLibrary = "ReportsLibrary";

/// <summary>

/// Название списка всех параметров

/// </summary>

public const string AllParamsList = "AllParamsList";

/// <summary>

/// Название списка переходов в справке

/// </summary>

public const string LinkSprList = "LinkSprList";

/// <summary>

/// Список, содержащий XSLT

/// </summary>

public const string StyleSiteList = "StyleSiteList";

/// <summary>

/// Название списка параметров справок

/// </summary>

public const string ParamSprList = "ParamSprList";

/// <summary>

/// Название списка для сохранения персональных настроек пользователя

/// </summary>

public const string AskvpDesktopUserConfig = "AskvpDesktopUserConfig";

/// <summary>

/// Название списка, содержащий конфигурационные настройки

/// </summary>

public const string ConfigPortal = "ConfigPortal";

public const string ExcelSessionToken = "excel";

public const string ReportSessionToken = "report";

public const string FormSessionToken = "form";

public static string CreateSessionKey(params string[] inputParams)

{

string sessionKey = null;

for (var i=0; i<inputParams.Length; i++)

{

if (i==inputParams.Length-1)

{

sessionKey += inputParams[i];

}

else

{

sessionKey += inputParams[i] + ":";

}

}

return sessionKey;

}

}

public static class Errors

{

public const string Security = "У Вас відсутній доступ до цієї довідки";

public const string SprNotFound = "Довідка не знайдена";

public const string Default = "Вибачаємоcя за тимчасові незручності.";

}

}

## 

## Текст класу DynamicConfiguration

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace AskvpDeskTop.Model

{

public class DynamicConfiguration

{

public string DasUrl { get; set; }

public string PortalUrl { get; set; }

public string XSLUrl { get; set; }

public string ServerServiceUrl { get; set; }

public string EmailAddress { get; set; }

public string SMTPserver { get; set; }

}

}

## 

## Текст файлу index.html

<!DOCTYPE html>

<!--[if lt IE 7]>

<html lang="en" xmlns:ng="http://angularjs.org" id="ng-app" ng-app="aDesktopApp"

class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->

<!--[if IE 7]>

<html lang="en" xmlns:ng="http://angularjs.org" id="ng-app" ng-app="aDesktopApp"

class="no-js lt-ie9 lt-ie8"> <![endif]-->

<!--[if lt IE 9]>

<html lang="en" xmlns:ng="http://angularjs.org" id="ng-app" ng-app="aDesktopApp" class="no-js lt-ie9"><![endif]-->

<!--[if gt IE 8]><!-->

<html lang="en" id="ng-app" xmlns:ng="http://angularjs.org" ng-app="aDesktopApp" class="no-js"> <!--<![endif]-->

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="description" content="">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="./css/main.css?v=@@version">

<title ng-bind="$stateParams['id\_spr'] ? $stateParams['id\_spr'] : 'Робочий стіл'">Робочий стіл</title>

<script src="./scripts/bowser.min.js"></script>

<script type="text/javascript">

if (

!(bowser.msie && bowser.version > 7 )

&& !(bowser.chrome && bowser.version > 38)

&& !(bowser.firefox && bowser.version > 32)

&& !(bowser.ios && bowser.version > 6)

) {

window.location = "/\_layouts/Browser.html";

}

if (bowser.msie && bowser.version == 8) {

window.location = "./ie8.html"

}

</script>

<!--[if lt IE 9]>

<script type="text/javascript" src="./scripts/es5-shim.js"></script>

<script type="text/javascript" src="./scripts/html5shiv.min.js"></script>

<script type="text/javascript" src="./scripts/respond.min.js"></script>

<script>

document.createElement('ui-select');

document.createElement('ui-select-match');

document.createElement('ui-select-choices');

document.createElement('ng-include');

document.createElement('toaster-container');

document.createElement('ng-pluralize');

document.createElement('ng-view');

// Optionally these for CSS

document.createElement('ng:include');

document.createElement('ng:pluralize');

document.createElement('ng:view');

</script>

<![endif]-->

</head>

<body>

<div class="main-container" ui-view></div>

<footer class="container">

<div>

<div style="padding-top: 10px; text-align: right;">

Версія клієнта: @@version

</div>

</div>

</footer>

<toaster-container toaster-options="{'close-button': true, 'position-class': 'toast-bottom-left'}"></toaster-container>

<script type="text/javascript" src="./scripts/main.js?v=@@version"></script>

</body>

</html>

## 

## Текст файлу authPartial.html

<ul class="nav navbar-nav navbar-right" ng-controller="authPartialCtrl">

<li ng-if="username !== null" class="dropdown"><a href="" class="dropdown-toggle" data-toggle="dropdown">{{username}}<span

class="caret"></span></a>

<ul class="dropdown-menu" role="menu">

<li><a href="/\_layouts/AccessDenied.aspx?loginasanotheruser=true">Вхід під іншим обліковим записом</a></li>

</ul>

</li>

<li ng-if="username === null"><a ng-href="{{loginPath}}">Увійти</a></li>

</ul>

## 

## Текст файлу mainView.html

<nav class="navbar navbar-default navbar-inverse">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header pull-left">

<ul class="nav navbar-nav navbar-left">

<li ng-if="$state.is('menu.search')" ui-sref-active="active"><a ng-click="toPreviousState()"><span

class="glyphicon glyphicon-menu-left"></span>&nbsp;&nbsp;НАЗАД</a></li>

</ul>

</div>

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<!--<a class="navbar-brand" href="#">Портал УЗ</a>-->

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1" ng-controller="mainViewCtrl">

<form class="navbar-form navbar-left" role="search" >

<div class="form-group">

<input type="text" ng-model="searchVal" class="form-control aDesktop-search" placeholder="Для початку пошуку введіть {{minSearchLength}} символи">

</div>

<div class="form-group">

<div class="btn-group">

<label style="width:120px; display:none; text-align:center;" class="btn btn-default" data-toggle="tooltip" data-placement="bottom" title="Пошук по усім справкам" ng-model="$stateParams['personalized']" btn-radio="'global'"><span class="glyphicon glyphicon-globe" aria-hidden="true"></span>&nbsp;Усі</label>

<label style="width:120px; display:none; text-align:center;" class="btn btn-default" data-toggle="tooltip" data-placement="bottom" title="Пошук по збереженим справкам" ng-model="$stateParams['personalized']" btn-radio="'personal'"><span class="glyphicon glyphicon-user" aria-hidden="true"></span>&nbsp;Збережені</label>

</div>

</div>

</form>

<div ng-include src="'partials/authPartial.html'"></div>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<ui-view>

## 

## Текст файлу menuView.html

<ol class="breadcrumb">

<li ng-repeat="item in storage.breadcrumbs" ui-sref-active="active" ng-if="$stateParams['personalized'] == 'global'">

<a ng-if="!$last" ui-sref="menu.view({id: item.id})">{{item.title}}</a>

<span ng-if="$last">{{item.title}}</span>

</li>

</ol>

<div class="container">

<div class="col-lg-offset-1 col-lg-10">

<div class="list-group">

<div ng-repeat="menuItem in MenuElems" ng-class="menuItem.ItemType == 'Link' ? 'list-group-item' : ''">

<a ng-click="callMenuDetail(menuItem.ID, menuItem.Title)"

ng-if="menuItem.ItemType == 'Folder' " class="list-group-item">

{{menuItem.Title}}

</a>

<div ng-click="callFormView(menuItem.ID)" ng-if="menuItem.ItemType == 'Link'">

<span class="label label-primary">{{menuItem.ID}}</span>

{{menuItem.Title}}

<!--<label ng-if="menuItem.Personalized" ng-click="callReportView(menuItem.ID); $event.preventDefault(); $event.stopPropagation();"-->

<!--title="Виконати запит із збереженими параметрами" class="btn btn-default pull-right"> <span class="glyphicon glyphicon-user pull-right"-->

<!--aria-hidden="true"></span></label>-->

</div>

</div>

</div>

</div>

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

<h3 class="text-center text-primary" ng-bind="TextInfo" ></h3>

</div>

</div>

## 

## Текст файлу searchView.html

<div class="container">

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

<h3 class="text-center text-primary" ng-bind="TextInfo" ></h3>

</div>

<div class="col-lg-offset-1 col-lg-10">

<div class="list-group">

<div class="list-group-item" ng-repeat="resultItem in searchResult">

<div ng-click="callFormView(resultItem.ID)">

<span class="label label-primary">{{resultItem.ID}}</span>

<span ng-bind-html="resultItem.Title | searchfilter: $stateParams['pattern']"></span>

<!--<label ng-if="resultItem.Personalized" ng-click=" callReportView(resultItem.ID);$event.preventDefault(); $event.stopPropagation();"-->

<!--class="btn btn-default pull-right"> <span class="glyphicon glyphicon-user pull-right"-->

<!--aria-hidden="true"></span></label>-->

</div>

</div>

</div>

</div>

<div class="text-center">

<!-- previous-text="&lsaquo;" next-text="&rsaquo;" first-text="&laquo;" last-text="&raquo;"-->

<pagination boundary-links="false" direction-links="false" ng-change="pageChanged()" max-size="maxSize" rotate="false" total-items="totalItems" ng-model="currentPage" num-pages="smallnumPages"></pagination>

</div>

</div>

## 

## Текст файлу formView.html

<nav class="navbar navbar-default navbar-inverse">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<!--<a class="navbar-brand" href="#">Портал УЗ</a>-->

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1" >

<ul class="nav navbar-nav navbar-left">

<li ui-sref-active="active"><a ui-sref="menu.view({id: $storage.lastMenuState.params.id, personalized: $storage.lastMenuState.params.personalized, tsid: $stateParams['tsid'] })">Меню</a></li>

</ul>

<div ng-include src="'partials/authPartial.html'"></div>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<div class="container">

<div class="panel panel-primary" block-ui="test">

<div class="panel-heading"><span class="label label-default">{{schemaForm.ID}}</span>&nbsp;&nbsp;{{schemaForm.Title }}</div>

<div class="panel-body">

<form>

<div class="row" ng-repeat="row in schemaForm.form.rows">

<div ng-repeat="form in row">

<div ng-include src="'formView/partials/formViewPartial.html?v=@@version'"/>

</div>

</div>

<div class="text-center">

<button type="button" class="btn btn-primary" ng-click="saveSelectedValToSession()">Запит</button>

<!--<button type="button" class="btn btn-success" ng-click="savePersonalData()">Зберегти&nbsp;&nbsp;<span class="glyphicon glyphicon-user" aria-hidden="true"></span></button>-->

</div>

</form>

</div>

</div>

</div>

@@comstart <div class="container">{{selectedVal | filter: json}}</div> @@comend

## 

## Текст файлу formViewPartial.html

<div class="form-group">

<label class="col-xs-{{form.DescrGridMobile}} col-xs-offset-{{form.DescrGridMobileOffset}} col-md-{{form.DescrGridDesktop}} col-md-offset-{{form.DescrGridDesktopOffset}} control-label"

ng-if="form.TypeParameter !='RadioButton' && form.TypeParameter!='Label'"

ng-class="form.DescriptionAlign == 'Left' ? 'text-left' : 'text-right'">{{ form.Description }}</label>

<div ng-switch on="form.TypeParameter"

ng-class="form.TypeParameter == 'Label' ?'col-md-12' :'col-xs-'+form.ParamGridMobile +' col-xs-offset-'+ form.ParamGridMobileOffset+ ' col-md-'+ form.ParamGridDesktop +' col-md-offset-'+ form.ParamGridDesktopOffset ">

<label ng-switch-when="Label" class="control-label text-center">

{{ form.Description }}

</label>

<div ng-switch-when="CalendarWithoutTime">

<p class="input-group">

<input type="text" ng-disabled="schemaForm.schema[form.ParametrName].Disable"

class="form-control" datetime-picker="dd.MM.yyyy"

ng-model="selectedVal[form.ParametrName].DefaultValue.Value"

is-open="schemaForm.schema[form.ParametrName].Opened"

enable-time="false"

clear-text="Очистити"

today-text="Сьогодні"

time-text="Час"

date-text="Дата"

close-text="Закрити"

/>

<span class="input-group-btn">

<button type="button" class="btn btn-default"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-click="openDatePicker($event,schemaForm.schema[form.ParametrName])"><i

class="glyphicon glyphicon-calendar"></i></button>

</span>

</div>

<div ng-switch-when="Calendar">

<p class="input-group">

<input type="text" ng-disabled="schemaForm.schema[form.ParametrName].Disable"

class="form-control" datetime-picker="dd.MM.yyyy HH:mm"

ng-model="selectedVal[form.ParametrName].DefaultValue.Value"

is-open="schemaForm.schema[form.ParametrName].Opened"

clear-text="Очистити"

now-text="Зараз"

today-text="Сьогодні"

time-text="Час"

date-text="Дата"

close-text="Закрити"

timepicker-options="{showMeridian: false}"

/>

<span class="input-group-btn">

<button type="button" class="btn btn-default"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-click="openDatePicker($event,schemaForm.schema[form.ParametrName])"><i

class="glyphicon glyphicon-calendar"></i></button>

</span>

</p>

</div>

<div ng-switch-when="TextBox">

<input type="text"

ng-if="!(form.TextModes == 'MultiLine')"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-model="selectedVal[form.ParametrName].DefaultValue.Value"

maxlength="{{form.CharacterLength ? form.CharacterLength : -1}}"

/>

<textarea ng-if="form.TextModes == 'MultiLine'"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-model="selectedVal[form.ParametrName].DefaultValue.Value"

maxlength="{{form.CharacterLength ? form.CharacterLength : -1}}"

rows="form.TextRows"

></textarea>

</div>

<div ng-switch-when="CheckBox">

<input type="checkbox" ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-model="selectedVal[form.ParametrName].DefaultValue.Value" ng-true-value="1" ng-false-value="0"

ng-change="setCheckBoxChanges(form.ParametrName,schemaForm.schema[form.ParametrName])"

/>

</div>

<div ng-switch-when="RadioButton">

<label>

<input ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-click="setRadioChanges(form.ParametrName,schemaForm.schema[form.ParametrName])" type="radio"

ng-checked="selectedVal[form.ParametrName].DefaultValue.Value \* 1"

name="{{form.RadioGroup}}"> {{ form.Description }}

</label>

</div>

<div ng-switch-when="DropDownList">

<ui-select ng-if="!form.Multiselect"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-model="selectedVal[form.ParametrName].DefaultValue" theme="selectize"

on-select="setChanges(form.ParametrName,schemaForm.schema[form.ParametrName])">

<ui-select-match >{{$select.selected.NValue}}</ui-select-match>

<ui-select-choices ng-mouseenter="$event.stopPropagation();"

repeat="item in schemaForm.schema[form.ParametrName].Data | filter: $select.search track by item.Value">

<div ng-mouseenter="$event.stopPropagation();" ng-click="selItem($select, $event, item)">

<div ng-bind-html="item.NValue | highlight: $select.search"></div>

</div>

</ui-select-choices>

</ui-select>

<!--<select multiple

ng-if="form.Multiselect"

ng-model="selectedVal[form.ParametrName]"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-options="item.NValue for item in schemaForm.schema[form.ParametrName].Data track by item.Value">

</select>-->

<ui-select ng-if="form.Multiselect"

multiple

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

style="width: 100%"

ng-model="selectedVal[form.ParametrName].DefaultValue" theme="select2">

<ui-select-match >{{$item.NValue}}</ui-select-match>

<ui-select-choices ng-mouseenter="$event.stopPropagation();"

repeat="dataItem in schemaForm.schema[form.ParametrName].Data | propsFilter: {Value: $select.search, NValue: $select.search}">

<div>{{dataItem.NValue}}</div>

</ui-select-choices>

</ui-select>

</div>

<div ng-switch-when="ListBox">

<ui-select ng-if="!form.Multiselect"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-model="selectedVal[form.ParametrName].DefaultValue" theme="selectize"

on-select="setChanges(form.ParametrName,schemaForm.schema[form.ParametrName])">

<ui-select-match >{{$select.selected.NValue}}</ui-select-match>

<ui-select-choices ng-mouseenter="$event.stopPropagation();"

repeat="item in schemaForm.schema[form.ParametrName].Data | filter: $select.search track by item.Value">

<div ng-mouseenter="$event.stopPropagation();" ng-click="selItem($select, $event, item)">

<div ng-bind-html="item.NValue | highlight: $select.search"></div>

</div>

</ui-select-choices>

</ui-select>

<!--<select multiple

ng-if="form.Multiselect"

ng-model="selectedVal[form.ParametrName]"

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-options="item.NValue for item in schemaForm.schema[form.ParametrName].Data track by item.Value">

</select>-->

<ui-select ng-if="form.Multiselect"

multiple

ng-disabled="schemaForm.schema[form.ParametrName].Disable"

style="width: 100%"

ng-model="selectedVal[form.ParametrName].DefaultValue" theme="select2">

<ui-select-match >{{$item.NValue}}</ui-select-match>

<ui-select-choices ng-mouseenter="$event.stopPropagation();"

repeat="dataItem in schemaForm.schema[form.ParametrName].Data | propsFilter: {Value: $select.search, NValue: $select.search}">

<div>{{dataItem.NValue}}</div>

</ui-select-choices>

</ui-select>

</div>

<div ng-switch-when="AutoComplete">

<ui-select ng-disabled="schemaForm.schema[form.ParametrName].Disable"

ng-model="selectedVal[form.ParametrName].DefaultValue"

theme="selectize"

reset-search-input="false"

on-select="setChanges(form.ParametrName,schemaForm.schema[form.ParametrName])">

<ui-select-match placeholder="Для початку пошуку введіть {{minSearchLength}} символи">{{$select.selected.NValue}}</ui-select-match>

<ui-select-choices

repeat="item in schemaForm.schema[form.ParametrName].Data | filter: $select.search track by item.Value"

refresh="refreshData(form.ParametrName,schemaForm.schema[form.ParametrName],$select.search)"

refresh-delay="0">

<div ng-mouseenter="$event.stopPropagation();" ng-click="selItem($select, $event, item)">

<div ng-bind-html="item.NValue | highlight: $select.search"></div>

</div>

</ui-select-choices>

</ui-select>

</div>

</div>

</div>

## 

## Текст файлу reportView.html

<nav class="navbar navbar-default navbar-inverse">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse"

data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<!--<a class="navbar-brand" href="#">Портал УЗ</a>-->

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav navbar-left">

<li>

<a ui-sref="menu.view({id: $storage.lastMenuState.params.id, personalized: $storage.lastMenuState.params.personalized, tsid: $stateParams['tsid'] })">Меню</a>

</li>

<li><a ui-sref="formView({id\_spr: $storage.id\_spr, tsid: $stateParams['tsid'] })">Параметри</a></li>

</ul>

<div ng-include src="'partials/authPartial.html'"></div>

<div class="navbar-right navbar-form">

<div class="form-group">

<button type="button" class="btn btn-success" ng-click="getExcel()">

<span class="glyphicon glyphicon-file "></span>&nbsp;Excel

</button>

<button class="btn btn-primary" style="display:none;">

<span class="glyphicon glyphicon-envelope"></span>&nbsp;e-mail

</button>

<button type="button" class="btn btn-default" ng-click="Print()">

<span class="glyphicon glyphicon-print"></span> Друк

</button>

</div>

</div>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container-fluid -->

</nav>

<!--<div class="container">

<div class="pull-right">

<button type="button" class="btn btn-success" ng-click="getExcel()">

<span class="glyphicon glyphicon-file "></span>&nbsp;Excel

</button>

<button class="btn btn-primary" style="display:none;">

<span class="glyphicon glyphicon-envelope"></span>&nbsp;e-mail

</button>

<button type="button" class="btn btn-default" ng-click="Print()">

<span class="glyphicon glyphicon-print"></span> Друк

</button>

</div>

</div>-->

<div ng-bind-html="report" print stick-header></div>

## 

## Текст файлу app.ts

/// <reference path="assets/types/angularjs/angular.d.ts"/>

/// <reference path="assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="components/blockuiDirective/blockuiDirective.ts"/>

/// <reference path="components/UUIDService.ts"/>

// Declare app level module which depends on views, and components

var desktopApp = angular.module('aDesktopApp', [

'ngAnimate',

'ui.router',

'ui.bootstrap',

'toaster',

'blockUI',

'ngStorage',

'ui.bootstrap.datetimepicker',

'aDesktopApp.mainView',

'aDesktopApp.formView',

'aDesktopApp.reportView',

'aDesktopApp.authPartial',

/\*'aDesktopApp.searchView',\*/

'ngSanitize'

])

.run(['$rootScope', '$state', '$stateParams', '$location', 'uuid', '$sessionStorage', '$http', '$window', function ($rootScope, $state, $stateParams, $location, UUID, $sessionStorage, $http, $window) {

$rootScope.$state = $state;

$rootScope.$stateParams = $stateParams;

$rootScope.$storage = $sessionStorage;

$rootScope.$window = $window;

if (!$rootScope.$storage.breadcrumbs)

$rootScope.$storage.breadcrumbs = [{id: 0, title: "Головне"}];

$rootScope.$on('$stateChangeStart', function (e, toState, toParams, fromState, fromParams) {

if (toParams['tsid'] == "0" || toParams['tsid'] == "") {

e.preventDefault();

toParams['tsid'] = UUID.gNew();

$state.go(toState.name, toParams);

}

if (toState != fromState) {

if (toState.name == 'menu.search') {

var previousState:any = new Object();

previousState.name = fromState.name;

previousState.params = fromParams;

$rootScope.$storage.previousState = previousState;

}

}

});

}]).

config(['$urlRouterProvider', 'blockUIConfig', function ($urlRouteProvider:ng.ui.IUrlRouterProvider, blockUIConfig:any) {

$urlRouteProvider.when('/menu', '/0/menu/global/0').otherwise('/0/menu/global/0');

blockUIConfig.message = 'Завантаження';

blockUIConfig.autoBlock = false;

}])/\*.

directive('uiBlock', BlockUIDirective.Factory)\*/.

service('uuid', UUID);

## 

## Текст файлу authPartial.ts

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="./authService.ts"/>

interface IAuthPartialScope extends ng.IScope

{

username: any;

loginPath: any;

}

class AuthPartialController

{

scope: IAuthPartialScope;

authService : IAuthService;

static $inject = ['AuthService','$scope'];

constructor(AuthService : IAuthService, $scope: IAuthPartialScope)

{

this.scope = $scope;

var current = this;

this.authService = AuthService;

this.authService.getData().then((result) => current.setResult(result));

}

setResult(data : any)

{

this.scope.username = data.username;

this.scope.loginPath = data.loginPath;

}

}

var authPartial = angular.module('aDesktopApp.authPartial', ['ui.router']);

authPartial.controller('authPartialCtrl', AuthPartialController);

## 

## Текст файлу authService.ts

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="./authPartial.ts"/>

interface IAuthService

{

getData() : ng.IPromise<any>;

}

class AuthService implements IAuthService

{

$http: ng.IHttpService;

$q: ng.IQService;

static $inject = ['$http','$q'];

constructor($http: ng.IHttpService, $q: ng.IQService)

{

this.$http = $http;

this.$q = $q;

}

getData():ng.IPromise<any>

{

var deferred = this.$q.defer();

this.$http.get('aDesktopApi/auth',{params:{"usermenu": true}})

.success(function(data, status, headers, config){

deferred.resolve(data);

})

.error(function(data, status, headers, config){

deferred.reject(status);

});

return deferred.promise;

}

}

authPartial.service('AuthService', AuthService);

## 

## Текст файлу mainView.ts

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="../Model/SearchOptEnum.ts" />

/// <reference path="../menuView/menuView.ts" />

/// <reference path="../searchView/searchView.ts" />

/// <reference path="../searchView/searchService.ts" />

interface IMainRootScope extends ng.IRootScopeService

{

$stateParams: any;

$storage: any;

}

interface IMainViewScope extends ng.IScope

{

searchVal : any;

searchOpt: any;

minSearchLength: number;

toPreviousState();

}

class MainViewController

{

scope: IMainViewScope;

rootScope : IMainRootScope;

stateService : ng.ui.IStateService;

timeoutService: ng.ITimeoutService;

static $inject = ['$scope', '$state','$rootScope','$timeout'];

constructor($scope : IMainViewScope, $stateService : ng.ui.IStateService, $rootScope : IMainRootScope,

$timeout: ng.ITimeoutService)

{

var current = this;

this.scope = $scope;

this.stateService = $stateService;

this.rootScope = $rootScope;

this.timeoutService = $timeout;

this.scope.minSearchLength = 3;

this.rootScope.$watch('$stateParams["personalized"]',(newvalue) =>

{

var nObject:any = { personalized : newvalue,tsid: current.rootScope.$stateParams['tsid'] };

if(current.stateService.includes('menu.view'))

{ current.stateService.go('menu.view',nObject);}

else

{

nObject.pattern = current.rootScope.$stateParams['pattern'];

current.stateService.go('menu.search',nObject);

}

});

var timeout = null;

this.scope.$watch('searchVal', (newvalue) =>

{

if(newvalue && newvalue.length >=3) {

if (timeout) current.timeoutService.cancel(timeout);

timeout = current.timeoutService(()=> {

current.stateService.go('menu.search', {

personalized: current.rootScope.$stateParams['personalized'],

pattern: newvalue,

tsid: current.rootScope.$stateParams['tsid'],

pagenum: 1

})

}, 400);

}

});

this.scope.toPreviousState=()=>

{

var previousState = current.rootScope.$storage.previousState;

current.stateService.go(previousState.name, previousState.params);

};

}

}

var mainView = angular.module('aDesktopApp.mainView', ['ui.router', 'ngSanitize'])

.config(['$stateProvider', function ($stateProvider:ng.ui.IStateProvider) {

$stateProvider.state('menu', {

abstract: true,

url: '',

templateUrl: './mainView/mainView.html?v=@@version',

controller: 'mainViewCtrl'

}).state('menu.view',

{

url: '/{tsid}/menu/{personalized}/{id}',

templateUrl: './menuView/menuView.html?v=@@version',

controller: 'menuViewCtrl'

})

.state('menu.search',

{

url: '/{tsid}/search/{personalized}/{pagenum:int}/{pattern}',

templateUrl: './searchView/searchView.html?v=@@version',

controller: 'searchViewCtrl'

})

}]);

mainView.controller('mainViewCtrl', MainViewController);

mainView.controller('menuViewCtrl', MenuViewController);

mainView.service('MenuService', MenuService);

mainView.controller('searchViewCtrl',SearchViewController);

mainView.service('SearchService', SearchService);

mainView.filter('searchfilter',['$sce', function($sce) {

function escapeRegexp(queryToEscape) {

return queryToEscape.replace(/([.?\*+^$[\]\\(){}|-])/g, '\\$1');

}

return function(matchItem, query) {

return $sce.trustAsHtml(query && matchItem ? matchItem.replace(new RegExp(escapeRegexp(query), 'gi'), '<span class="highlight-aDesktop">$&</span>') : matchItem);

};

}]);

## 

## Текст файлу menuView.ts

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="./menuService.ts"/>

/// <reference path="../Model/MenuItem.ts" />

/// <reference path="../Model/SearchOptEnum.ts" />

interface IMenuViewScope extends ng.IScope

{

MenuElems : MenuItem[];

callMenuDetail(id : any, title : any);

callReportView(sprCode : any);

callFormView(sprCode : any);

storage : any;

TextInfo : string;

}

class MenuViewController

{

menuService : IMenuService;

scope: IMenuViewScope;

stateParams : ng.ui.IStateParamsService;

stateService: ng.ui.IStateService;

sessionStorageService : any;

static $inject = ['MenuService', '$scope', '$stateParams','$state','$sessionStorage'];

constructor(MenuService : IMenuService,$scope : IMenuViewScope, $stateParams : ng.ui.IStateParamsService,$state: ng.ui.IStateService,$sessionStorage : any)

{

var current = this;

this.menuService = MenuService;

this.scope = $scope;

this.stateParams = $stateParams;

this.stateService = $state;

this.sessionStorageService = $sessionStorage;

this.scope.storage = this.sessionStorageService;

/\*save last menu state\*/

var lastMenuState: any = new Object();

lastMenuState.name = this.stateService.current.name;

lastMenuState.params = this.stateService.params;

this.sessionStorageService.lastMenuState = lastMenuState;

for (var i = this.sessionStorageService.breadcrumbs.length - 1; i > 0; i--) {

if (this.sessionStorageService.breadcrumbs[i].id == this.stateParams['id'] || this.sessionStorageService.breadcrumbs[i].id == 0)

break;

this.sessionStorageService.breadcrumbs.pop();

}

this.scope.callMenuDetail = (id: any, title: any)=>

{

current.sessionStorageService.breadcrumbs.push(

{

id: id,

title : title

}

);

current.stateService.go('menu.view',{ id: id });

};

this.scope.TextInfo = 'Завантаження...';

if (this.stateParams["personalized"] == SearchOpt[SearchOpt.global]) {

this.menuService.getMenu(this.stateParams["id"]).then((result) => current.setResult(result));

} else {

this.menuService.getPersonalMenu().then((result) => current.setResult(result));

}

this.scope.callReportView = (sprCode: any) =>

{

current.stateService.go('reportView',{ option: 'id\_spr=' + sprCode , tsid: current.stateParams['tsid'], direction : 'direct' });

};

this.scope.callFormView = (sprCode : any) =>

{

current.stateService.go('formView', {id\_spr: sprCode, tsid: current.stateParams['tsid']});

};

}

private setResult(result)

{

this.scope.TextInfo = result.length == 0 ? "Довідки відсутні" : "";

this.scope.MenuElems = result;

}

}

var menuView = angular.module('aDesktopApp.menuView', ['ui.router', 'ngSanitize']);

menuView.controller('menuViewCtrl', MenuViewController);

## 

## Текст файлу menuService.ts

/// <reference path="./menuView.ts"/>

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../Model/MenuItem.ts" />

interface IMenuService

{

getMenu(folderID: string) : ng.IPromise<MenuItem[]>;

getPersonalMenu() : ng.IPromise<MenuItem[]>;

}

class MenuService implements IMenuService

{

$http: ng.IHttpService;

$q: ng.IQService;

static $inject = ['$http','$q'];

constructor($http: ng.IHttpService, $q: ng.IQService)

{

this.$http = $http;

this.$q = $q;

}

getMenu(folderID : string):ng.IPromise<MenuItem[]>

{

var deferred = this.$q.defer();

this.$http.get('aDesktopApi/menu',{params:{"folderid": folderID}})

.success(function(data, status, headers, config){

deferred.resolve(data.menu);

})

.error(function(data, status, headers, config){

deferred.reject(status);

});

return deferred.promise;

}

getPersonalMenu() : ng.IPromise<MenuItem[]>

{

var deferred = this.$q.defer();

this.$http.get('/aDesktopApi/menu',{params:{"option": true}})

.success(function(data, status, headers, config){

deferred.resolve(data.menu);

})

.error(function(data, status, headers, config){

deferred.reject(status);

});

return deferred.promise;

}

}

menuView.service('MenuService', MenuService);

## 

## Текст файлу searchView.ts

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="./searchService.ts"/>

/// <reference path="../Model/SearchModel.ts" />

interface ISearchViewScope extends ng.IScope

{

searchResult : FindItem[];

callReportView(sprCode : any);

callFormView(sprCode : any);

pageChanged();

TextInfo : string;

currentPage : any;

totalItems : number;

maxSize : number;

}

class SearchViewController

{

scope: ISearchViewScope;

searchService : ISearchService;

stateParams : ng.ui.IStateParamsService;

stateService : ng.ui.IStateService;

toasterService : any;

itemsPerPage: number = 10;

static $inject = ['SearchService', '$scope','$stateParams','$state','toaster'];

constructor($SearchService: ISearchService, $scope: ISearchViewScope, $stateParams : ng.ui.IStateParamsService, $state : ng.ui.IStateService, toaster : any)

{

var current = this;

this.searchService = $SearchService;

this.scope = $scope;

this.stateParams = $stateParams;

this.stateService = $state;

this.toasterService = toaster;

this.scope.callReportView = (sprCode: any) =>

{

current.stateService.go('reportView',{ option: 'id\_spr=' + sprCode , tsid: current.stateParams['tsid'], direction : 'direct' });

};

this.scope.callFormView = (sprCode : any) =>

{

current.stateService.go('formView', {id\_spr: sprCode, tsid: current.stateParams['tsid']});

};

this.scope.pageChanged=()=>

{

current.stateService.go("menu.search", {pagenum:current.scope.currentPage}, {inherit:true});

};

this.scope.maxSize = 0;

this.scope.TextInfo = 'Завантаження...';

var searchModel:SearchModel = new SearchModel();

searchModel.ItemsPerPage = this.itemsPerPage;

searchModel.PageNum = this.stateParams["pagenum"];

searchModel.Pattern = this.stateParams["pattern"];

searchModel.Personalized = this.stateParams["personalized"] == SearchOpt[SearchOpt.personal];

this.searchService.getData(searchModel).then(function (data) {

current.scope.currentPage = current.stateParams["pagenum"];

var isEmpty: boolean = data.suggestions.length == 0;

current.scope.TextInfo = isEmpty ? "Інформація відсутня" : "";

current.scope.maxSize = isEmpty || data.countOfItems<= current.itemsPerPage ? 0 : 10;

current.scope.totalItems = data.countOfItems;

current.scope.searchResult = data.suggestions;

}).catch(function(data)

{

current.toasterService.pop('error','Помилка',data);

});

}

}

/\*var searchView = angular.module('aDesktopApp.searchView', ['ui.router', 'ngSanitize'])

.config(['$stateProvider', function ($stateProvider: ng.ui.IStateProvider) {

$stateProvider.state('searchView', {

url: '/{tsid}/searchView/{personalized}/{pattern}/',

templateUrl: './searchView/searchView.html',

controller: 'searchViewCtrl'

});

}]);\*/

/\*

searchView.controller('searchViewCtrl',SearchViewController);\*/

## 

## Текст файлу searchService.ts

/// <reference path="./searchView.ts"/>

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../Model/FindItem.ts" />

/// <reference path="../Model/SearchModel.ts" />

/// <reference path="../Model/SearchOptEnum.ts" />

interface ISearchService

{

getData(searchdata : SearchModel) : ng.IPromise<any>

}

class SearchService implements ISearchService

{

$http: ng.IHttpService;

$q: ng.IQService;

static $inject = ['$http','$q'];

constructor($http: ng.IHttpService, $q: ng.IQService)

{

this.$http = $http;

this.$q = $q;

}

getData(searchdata : SearchModel):ng.IPromise<any>

{

var deferred = this.$q.defer();

this.$http({ method: 'POST',url:'/aDesktopApi/search', data: searchdata})

.success(function(data){

deferred.resolve(data);

})

.error(function(data){

deferred.reject(data);

});

return deferred.promise;

}

}

/\*searchView.service('SearchService', SearchService);\*/

## 

## Текст файлу formView.ts

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="../assets/types/moment/moment.d.ts"/>

/// <reference path="../Model/SchemaFormModel.ts" />

/// <reference path="../Model/SimpleFormItem.ts" />

/// <reference path="./formService.ts"/>

interface IFormViewScope extends ng.IScope {

schemaForm: any;

selectedVal : any;

minSearchLength: number;

setChanges(parametrName:string, selectedObject:any);

setRadioChanges(parametrName:string, selectedObject:any);

setCheckBoxChanges(parametrName:string, selectedObject:any);

refreshData(parametrName:string, selectedObject:any, val:any);

saveSelectedValToSession();

savePersonalData();

openDatePicker(event:any, value:any);

/\* fix ui-select ie8\*/

selItem(select:any, event:any, item:any);

}

class FormViewController {

formService:IFormService;

scope:IFormViewScope;

stateParams:ng.ui.IStateParamsService;

stateService:ng.ui.IStateService;

toastService:any;

blockUIService:any;

timeoutService:ng.ITimeoutService;

sessionStorageService:any;

static $inject = ['FormService', '$scope', '$stateParams', '$state', 'toaster', 'blockUI', '$timeout', '$sessionStorage'];

constructor($FormService:IFormService, $scope:IFormViewScope, $stateParams:ng.ui.IStateParamsService, $stateService:ng.ui.IStateService, toaster:any, blockUI:any, $timeout:ng.ITimeoutService, $sessionStorage:any) {

var current = this;

this.formService = $FormService;

this.scope = $scope;

this.stateParams = $stateParams;

this.stateService = $stateService;

this.toastService = toaster;

this.blockUIService = blockUI;

this.timeoutService = $timeout;

this.sessionStorageService = $sessionStorage;

this.scope.minSearchLength = 3;

this.sessionStorageService.id\_spr = this.stateParams["id\_spr"];

this.blockUIService.start();

this.formService.getForm(this.stateParams["tsid"], this.stateParams["id\_spr"]).then(function (data) {

console.log(data);

current.scope.schemaForm = data;

current.blockUIService.stop();

current.init();

}).catch(function (data) {

current.blockUIService.stop();

current.toastService.pop('error', 'Помилка під час завантаження форми ' + data);

});

this.scope.selectedVal = {};

this.scope.setChanges = (parametrName, selectedObject) => this.setChanges(parametrName, selectedObject);

this.scope.setRadioChanges = (parametrName, selectedObject) => this.setRadioChanges(parametrName, selectedObject);

this.scope.setCheckBoxChanges= (parametrName, selectedObject) => this.setCheckBoxChanges(parametrName, selectedObject);

this.scope.savePersonalData = () => {

current.toastService.pop('info', "Збереження даних");

current.formService.savePersonalData(current.scope.schemaForm.ID, this.prepareDataToSave()).then(()=> {

current.toastService.pop('success', "Дані успішно збережно");

}).catch(function () {

current.toastService.pop('error', "Помилка під час збереження даних");

});

};

this.scope.saveSelectedValToSession = () => {

var dataArray = this.prepareDataToSave();

current.formService.saveSelectedDataToSession(this.stateParams['tsid'], current.scope.schemaForm.ID, dataArray).then(()=> {

current.blockUIService.start();

this.formService.validateData(this.stateParams['tsid'], current.scope.schemaForm.ID).then(

function (data) {

current.blockUIService.stop();

if (data.validationErrorMessages !== null) {

current.toastService.pop('error', "Помилка: " + data.validationErrorMessages);

}

else {

current.stateService.go('reportView', {

option: 'id\_spr=' + current.stateParams["id\_spr"],

tsid: current.stateParams["tsid"],

direction: 'transition'

});

}

}).catch(function (data) {

current.blockUIService.stop();

});

});

};

/\*this.scope.saveSelectedValToSession = () => {

var dataArray = this.prepareDataToSave();

current.formService.saveSelectedDataToSession(this.stateParams['tsid'], current.scope.schemaForm.ID, dataArray).then(()=> {

current.stateService.go('reportView', {

option: 'id\_spr=' + this.stateParams["id\_spr"],

tsid: this.stateParams["tsid"],

direction : 'transition'

});

});

};\*/

this.scope.openDatePicker = (event, value) => {

event.preventDefault();

event.stopPropagation();

value.Opened = true;

};

var timeout = null;

//только для автокомплита

this.scope.refreshData = (parametrName, selectedObject, val) => {

/\*Clear old Data\*/

current.scope.schemaForm.schema[parametrName].Data = [];

if (val.length <= 2) {

if (timeout) {

current.timeoutService.cancel(timeout);

}

return;

}

var Data = [];

if (timeout)

current.timeoutService.cancel(timeout);

timeout = current.timeoutService(()=> {

if (selectedObject.DependOn) {

for (var i = 0; i < selectedObject.DependOn.length; i++) {

var selVal = current.scope.selectedVal[selectedObject.DependOn[i]].DefaultValue;

if (!selVal) break;

var queryItem = new QueryItem();

queryItem.DasType = current.scope.schemaForm.schema[selectedObject.DependOn[i]].DasType;

queryItem.ParametrName = selectedObject.DependOn[i];

queryItem.DefaultItems.push(selVal);

Data.push(queryItem);

}

}

var queryModel = new QueryModel();

queryModel.ParametrName = parametrName;

queryModel.QueryName = selectedObject.KeyName;

queryModel.DasNamedParams = selectedObject.DasNamedParams;

queryModel.QueryData = Data;

queryModel.SearchPattern = val;

/\* block elem\*/

//parentValue.isBusy = true;

this.formService.getData(queryModel).then(

function (result) {

var formItem = current.scope.schemaForm.schema[result.ParametrName];

formItem.Data = result.DefaultItems;

});

}, 400);

};

/\* fix ui-select ie 8 \*/

this.scope.selItem = (select, event, item) => {

event.stopPropagation();

select.select(item);

}

}

//Установка зависимостей

private setDependentAndAffectPropAndGrouped():void {

var current = this;

angular.forEach(current.scope.schemaForm.schema, function (parentValue, parentKey) {

angular.forEach(current.scope.schemaForm.schema, function (value, key) {

/\* Set dependent prop \*/

if (value.DependOn && value.DependOn.indexOf(parentKey) > -1) {

if (!("Dependent" in parentValue))

parentValue.Dependent = [];

parentValue.Dependent.push(key);

}

/\* Set affected prop \*/

if (value.AffectNameParametr && value.AffectNameParametr == parentKey) {

if (!("Affected" in parentValue))

parentValue.Affected = [];

var affectObj:any = {};

affectObj.ParametrName = key;

affectObj.AffectSource = value.AffectSource;

affectObj.AffectSourceObject = value.AffectSourceObject;

affectObj.DependAction = value.DependAction;

parentValue.Affected.push(affectObj);

}

/\* Set Grouped \*/

if (parentKey != key && value.RadioGroup && value.RadioGroup == parentValue.RadioGroup) {

if (!("Grouped" in parentValue))

parentValue.Grouped = [];

parentValue.Grouped.push(key);

}

}

);

});

}

private prepareDataToSave():any {

var dataArray = new Object();

var current = this;

angular.forEach(this.scope.selectedVal, function (value, key) {

console.log(value);

console.log("key " + key);

if (!current.scope.schemaForm.schema[key].Disable) {

dataArray[key] = {};

var typeParameter = current.scope.schemaForm.schema[key].TypeParameter;

switch (typeParameter) {

case TypeParameters[TypeParameters.Calendar]:

case TypeParameters[TypeParameters.CalendarWithoutTime]:

{

var emptyChar:string = " ";

var dateFormat:string = typeParameter == TypeParameters[TypeParameters.Calendar] ? "DD.MM.YYYY HH:mm:ss" : "DD.MM.YYYY";

var calendarItem:any = new Object();

calendarItem.NValue = '';

calendarItem.Value = value.DefaultValue.Value && !isNaN(value.DefaultValue.Value.getTime()) ? moment(value.DefaultValue.Value).format(dateFormat) : emptyChar;

dataArray[key].DefaultValue = calendarItem;

}

break;

default:

{

dataArray[key] = value;

}

}

dataArray[key].FilteringAction = value.FilteringAction;

dataArray[key].SprFilteringAction = value.SprFilteringAction;

}

});

console.log(dataArray);

return dataArray;

}

private init() {

var current = this;

current.setDependentAndAffectPropAndGrouped();

angular.forEach(current.scope.schemaForm.schema, function (parentValue, parentKey) {

current.scope.selectedVal[parentKey] = {};

switch (parentValue.TypeParameter) {

case TypeParameters[TypeParameters.ListBox]:

case TypeParameters[TypeParameters.DropDownList]:

{

current.scope.selectedVal[parentKey].DefaultValue = [];

if (parentValue.SourceObject == SourceObject[SourceObject.DAS]) {

if (!parentValue.DependOn) {

current.callGetData(parentKey, parentValue, null, null);

}

}

else {

current.setDefaultVal(parentKey, parentValue);

current.setChanges(parentKey, parentValue);

}

}

break;

case TypeParameters[TypeParameters.AutoComplete]:

{

if (!parentValue.DependOn) {

if (parentValue.DefaultValue)

current.callGetData(parentKey, parentValue, null, parentValue.DefaultValue[0].Value);

}

}

break;

case TypeParameters[TypeParameters.Calendar]:

case TypeParameters[TypeParameters.CalendarWithoutTime]:

{

var dateFormat:string = parentValue.TypeParameter == TypeParameters[TypeParameters.Calendar] ? "DD.MM.YYYY HH:mm:ss" : "DD.MM.YYYY";

current.scope.selectedVal[parentKey].DefaultValue = parentValue.DefaultValue[0];

if (parentValue.DefaultValue[0])

current.scope.selectedVal[parentKey].DefaultValue.Value = moment(parentValue.DefaultValue[0].Value, dateFormat).toDate();

}

break;

default:

{

current.scope.selectedVal[parentKey].DefaultValue = parentValue.DefaultValue[0];

current.scope.selectedVal[parentKey].DefaultValue.Value = parentValue.DefaultValue[0].Value;

}

break;

}

current.scope.selectedVal[parentKey].FilteringAction = parentValue.FilteringAction;

current.scope.selectedVal[parentKey].SprFilteringAction = parentValue.SprFilteringAction;

});

/\* initial affect on start \*/

angular.forEach(current.scope.schemaForm.schema, function (parentValue, parentKey) {

current.setAffect(parentKey, parentValue);

});

}

//Подгрузка данных в первый раз и подгрузка данных для зависимого элемента (DAS)

private callGetData(parentKey, parentValue, QueryData, searchPattern) {

var current = this;

var queryModel = new QueryModel();

queryModel.ParametrName = parentKey;

queryModel.QueryName = parentValue.KeyName;

queryModel.DasNamedParams = parentValue.DasNamedParams;

queryModel.QueryData = QueryData;

queryModel.SearchPattern = searchPattern;

/\* block elem\*/

//parentValue.isBusy = true;

var myBlockUI = this.blockUIService.instances.get('test');

myBlockUI.start('Завантаження даних...');

this.formService.getData(queryModel).then(

function (result) {

var formItem = current.scope.schemaForm.schema[result.ParametrName];

// formItem.isBusy = false;

myBlockUI.stop();

if (formItem.TypeParameter == TypeParameters[TypeParameters.AutoComplete]) {

if (!formItem.DefaultValue) {

formItem.Data = [result.DefaultItems[0]];

current.scope.selectedVal[result.ParametrName].DefaultValue = formItem.Data[0];

}

else {

var defaultVal = formItem.DefaultValue;

var selectedIndex = 0;

if (defaultVal) {

for (var i = 0; i < result.DefaultItems.length; i++) {

if (result.DefaultItems[i].Value == defaultVal[0].Value) {

selectedIndex = i;

break;

}

}

}

formItem.Data = [result.DefaultItems[selectedIndex]];

current.scope.selectedVal[result.ParametrName].DefaultValue = formItem.Data[0];

}

current.setChanges(result.ParametrName, formItem);

}

else {

formItem.Data = result.DefaultItems;

current.setDefaultVal(result.ParametrName, formItem);

current.setChanges(result.ParametrName, formItem);

}

}).catch(function (data) {

myBlockUI.stop();

current.toastService.pop('error', 'Помилка під час завантаження даних ' + data);

});

}

private setDefaultVal(keyItem, valueItem) {

var defaultVal = valueItem.DefaultValue;

var selectedIndex = 0;

if (defaultVal) {

for (var i = 0; i < valueItem.Data.length; i++) {

if (valueItem.Data[i].Value == defaultVal[0].Value) {

selectedIndex = i;

break;

}

}

}

this.scope.selectedVal[keyItem].DefaultValue = valueItem.Data[selectedIndex];

}

private setAffect(selectedKey:string, selectedObject:any) {

var current = this;

if (selectedObject.Affected) {

for (var i = 0; i < selectedObject.Affected.length; i++) {

var selectedVal:any;

selectedVal = this.scope.selectedVal[selectedKey];

var isInverse:boolean;

isInverse = !(selectedObject.Affected[i].AffectSource.indexOf(selectedVal.DefaultValue.Value) > -1)

this.setAffectEvent(this.scope.schemaForm.schema[selectedObject.Affected[i].ParametrName], isInverse);

}

}

}

private setRadioChanges(parametrName:string, selectedObject:any) {

this.scope.selectedVal[parametrName].DefaultValue.Value = '1';

if (selectedObject.Grouped) {

for (var i = 0; i < selectedObject.Grouped.length; i++) {

this.scope.selectedVal[selectedObject.Grouped[i]].DefaultValue.Value = 0;

this.setAffect(selectedObject.Grouped[i], this.scope.schemaForm.schema[selectedObject.Grouped[i]]);

}

}

this.setAffect(parametrName, selectedObject);

}

private setCheckBoxChanges(parametrName:string, selectedObject:any) {

this.setAffect(parametrName, selectedObject);

}

private setAffectEvent(affectedObject:any, isInverse:boolean) {

switch (affectedObject.DependAction) {

case DependActions[DependActions.Disable]:

{

affectedObject.Disable = !isInverse;

}

break;

case DependActions[DependActions.Enable]:

{

affectedObject.Disable = !!isInverse;

}

break;

}

}

private setChanges(parametrName:string, selectedObject:any) {

var current = this;

if (selectedObject.Dependent) {

angular.forEach(selectedObject.Dependent, function (dependentVal) {

var dependentObject = current.scope.schemaForm.schema[dependentVal];

var Data = [];

for (var i = 0; i < dependentObject.DependOn.length; i++) {

if(!current.scope.selectedVal[dependentObject.DependOn[i]])

return;

var selVal = current.scope.selectedVal[dependentObject.DependOn[i]].DefaultValue;

if (!selVal || selVal.length == 0) return;

var queryItem = new QueryItem();

queryItem.DasType = current.scope.schemaForm.schema[dependentObject.DependOn[i]].DasType;

queryItem.ParametrName = dependentObject.DependOn[i];

queryItem.DefaultItems.push(selVal);

Data.push(queryItem);

}

if (Data.length > 0) {

current.callGetData(dependentVal, dependentObject, Data, null);

}

});

}

if (selectedObject.Affected) {

this.setAffect(parametrName, selectedObject);

}

}

}

var formView = angular.module('aDesktopApp.formView', ['ui.router', 'ngSanitize', 'ui.select'])

.config(['$stateProvider', function ($stateProvider:ng.ui.IStateProvider) {

$stateProvider.state('formView', {

url: '/{tsid}/form/{id\_spr}',

templateUrl: './formView/formView.html?v=@@version',

controller: 'formViewCtrl'

});

}]);

formView.controller('formViewCtrl', FormViewController);

formView.filter('propsFilter', function () {

return function (items, props) {

var out = [];

if (angular.isArray(items)) {

items.forEach(function (item) {

var itemMatches = false;

var keys = Object.keys(props);

for (var i = 0; i < keys.length; i++) {

var prop = keys[i];

var text = props[prop].toLowerCase();

if (item[prop].toString().toLowerCase().indexOf(text) !== -1) {

itemMatches = true;

break;

}

}

if (itemMatches) {

out.push(item);

}

});

} else {

// Let the output be the input untouched

out = items;

}

return out;

};

});

## 

## Текст файлу formService.ts

/// <reference path="./formView.ts"/>

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../Model/QueryItem.ts" />

/// <reference path="../Model/QueryModel.ts" />

/// <reference path="../Model/DefaultItem.ts" />

/// <reference path="../Model/SimpleFormItem.ts" />

interface IFormService {

getForm(tsid:string, pattern:string) : ng.IPromise<DefaultItem[]>;

getData(queryObject:QueryModel) : ng.IPromise<QueryItem>;

savePersonalData(sprCode:string, personalData:any) : ng.IPromise<boolean>;

saveSelectedDataToSession(tsid:string, sprCode:string, Data:any) : ng.IPromise<boolean>;

validateData(tsid:string, sprCode:string) : ng.IPromise<any>;

}

class FormService implements IFormService {

$http:ng.IHttpService;

$q:ng.IQService;

static $inject = ['$http', '$q'];

constructor($http:ng.IHttpService, $q:ng.IQService) {

this.$http = $http;

this.$q = $q;

}

getForm(tsid:string, pattern:string):ng.IPromise<DefaultItem[]> {

var deferred = this.$q.defer();

this.$http.get('/aDesktopApi/form', {

params: {"tsid": tsid, "id\_spr": pattern},

headers: {

'Cache-Control': 'no-cache',

'If-Modified-Since': 'Mon, 26 Jul 1997 05:00:00 GMT',

'Pragma': 'no-cache'

}

})

.success(function (data, status, headers, config) {

deferred.resolve(data);

})

.error(function (data, status, headers, config) {

deferred.reject(data);

});

return deferred.promise;

}

getData(queryObj:QueryModel):ng.IPromise<QueryItem> {

var deferred = this.$q.defer();

this.$http.post('/aDesktopApi/query', queryObj)

.success(function (data, status, headers, config) {

var result = new QueryItem();

result.DefaultItems = data.data;

result.ParametrName = config.data.ParametrName;

deferred.resolve(result);

})

.error(function (data, status, headers, config) {

deferred.reject(data);

});

return deferred.promise;

}

savePersonalData(sprCode:string, personalData:any):ng.IPromise<boolean> {

var deferred = this.$q.defer();

var simpleFormItemCol:Array<SimpleFormItem> = this.prepareItemsToSave(personalData);

this.$http({method: 'POST', url: '/aDesktopApi/form', params: {"savepd": sprCode}, data: simpleFormItemCol})

.success(function (data, status, headers, config) {

deferred.resolve(true);

})

.error(function (data, status, headers, config) {

deferred.resolve(false);

});

return deferred.promise;

}

saveSelectedDataToSession(tsid:string, sprCode:string, Data:any):ng.IPromise<boolean> {

var deferred = this.$q.defer();

var simpleFormItemCol:Array<SimpleFormItem> = this.prepareItemsToSave(Data);

this.$http({

method: 'POST',

url: '/aDesktopApi/form',

params: {"save": sprCode, "tsid": tsid},

data: simpleFormItemCol

})

.success(function (data, status, headers, config) {

deferred.resolve(true);

})

.error(function (data, status, headers, config) {

deferred.resolve(false);

});

return deferred.promise;

}

validateData(tsid:string, sprCode:string):ng.IPromise<any> {

var deferred = this.$q.defer();

this.$http({

method: 'GET',

url: '/aDesktopApi/report',

params: {"validate": true, "id\_spr": sprCode, "tsid": tsid}

})

.success(function (data, status, headers, config) {

deferred.resolve(data);

})

.error(function (data, status, headers, config) {

deferred.resolve(data);

});

return deferred.promise;

}

private prepareItemsToSave(Data:any):SimpleFormItem[] {

var simpleFormItemCol:Array<SimpleFormItem> = new Array<SimpleFormItem>();

angular.forEach(Data, (value, key)=> {

var simpleFormItem:SimpleFormItem = new SimpleFormItem();

simpleFormItem.DefaultValue = angular.isArray(value.DefaultValue) ? value.DefaultValue : [value.DefaultValue];

simpleFormItem.Description = "";

simpleFormItem.ParametrName = key;

simpleFormItem.FilteringAction = value.FilteringAction;

simpleFormItem.SprFilteringAction = value.SprFilteringAction;

simpleFormItemCol.push(simpleFormItem);

});

return simpleFormItemCol;

}

}

formView.service('FormService', FormService);

## 

## Текст файлу reportView.ts

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

/// <reference path="../assets/types/angularjs/angular-ui-router.d.ts"/>

/// <reference path="./reportService.ts"/>

/// <reference path="../components/printDirective/printDirective.ts"/>

/// <reference path="../components/stickHeaderDirective/stickHeaderDirective.ts"/>

interface IMainRootScope extends ng.IRootScopeService {

$stateParams: any;

}

interface IPrint {

Print()

}

interface IReportViewScope extends ng.IScope,IPrint {

report : string;

getExcel(): void;

}

class ReportViewController {

scope:IReportViewScope;

reportService:IReportService;

stateParams:ng.ui.IStateParamsService;

windowService:ng.IWindowService;

sceService:ng.ISCEService;

rootScopeService:IMainRootScope;

locationService:ng.ILocationService;

blockUIService:any;

toastService:any;

static $inject = ['ReportService', '$scope', '$stateParams', '$window', '$sce', 'blockUI', '$rootScope', '$location', 'toaster'];

constructor($ReportService:IReportService, $scope:IReportViewScope, $stateParams:ng.ui.IStateParamsService, $window:ng.IWindowService, $sce:ng.ISCEService, $blockUI:any, $rootScope:IMainRootScope, $location:ng.ILocationService, toaster:any) {

var current = this;

this.reportService = $ReportService;

this.scope = $scope;

this.stateParams = $stateParams;

this.windowService = $window;

this.sceService = $sce;

this.blockUIService = $blockUI;

this.rootScopeService = $rootScope;

this.locationService = $location;

this.toastService = toaster;

$blockUI.start();

this.rootScopeService.$stateParams["id\_spr"] = this.getQueryVariables($stateParams['option'])['id\_spr'];

this.reportService.getData($stateParams['option'], $stateParams["tsid"], $stateParams['option'], $stateParams["direction"] == "direct").then(function (data) {

current.scope.report = current.sceService.trustAsHtml(data);

$blockUI.stop();

},

(data)=> {

current.toastService.pop('error', 'Помилка', data);

$blockUI.stop()

});

this.scope.getExcel = () => {

current.windowService.location.href = "/aDesktopApi/excel?id\_spr=" + this.getQueryVariables($stateParams['option'])['id\_spr'] + "&tsid=" + current.stateParams["tsid"];

};

}

getQueryVariables(query) {

query = decodeURIComponent(query);

var vars = query.split('&');

var arr = {};

for (var i = 0; i < vars.length; i++) {

var pair = vars[i].split('=');

arr[pair[0]] = pair[1];

}

return arr;

}

}

var reportView = angular.module('aDesktopApp.reportView', ['ui.router', 'ngSanitize'])

.config(['$stateProvider', function ($stateProvider:ng.ui.IStateProvider) {

$stateProvider.state('reportView', {

url: '/{tsid}/report/{direction}/?{option}',

templateUrl: './reportView/reportView.html?v=@@version',

controller: 'reportViewCtrl'

});

}]).

directive('print', PrintDirective.Factory()).

directive('stickHeader', StickHeaderDirective.Factory());

reportView.controller('reportViewCtrl', ReportViewController);

## 

## Текст файлу reportService.ts

/// <reference path="./reportView.ts"/>

/// <reference path="../assets/types/angularjs/angular.d.ts"/>

interface IReportService

{

getData(sprCode: string, tsid: string, param: any, isDirect : boolean) : ng.IPromise<string>

}

class ReportService implements IReportService

{

$http: ng.IHttpService;

$q: ng.IQService;

static $inject = ['$http','$q'];

constructor($http: ng.IHttpService, $q: ng.IQService)

{

this.$http = $http;

this.$q = $q;

}

getQueryVariables(query) {

query = decodeURIComponent(query);

var vars = query.split('&amp;');

var arr = {};

for (var i = 0; i < vars.length; i++) {

var pair = vars[i].split('=');

arr[pair[0]] = pair[1];

}

return arr;

}

getData(sprCode : string, tsid: string, param: any, isDirect : boolean ):ng.IPromise<string>

{

//param = this.getQueryVariables(param);

var deferred = this.$q.defer();

//param.id\_spr = param.sprCode || sprCode;

//param.tsid = tsid;

if(isDirect) param.option = isDirect;

this.$http.get('/aDesktopApi/report?'+ param.replace(/&amp;/g,"&") + "&tsid=" + tsid ,{

headers: {

'Cache-Control': 'no-cache',

'If-Modified-Since': 'Mon, 26 Jul 1997 05:00:00 GMT',

'Pragma': 'no-cache'

}

})

.success(function(data, status, headers, config){

deferred.resolve(data);

})

.error(function(data, status, headers, config){

deferred.reject(data);

});

return deferred.promise;

}

}

reportView.service('ReportService', ReportService);