App-Admin\client

Mario Niemand

MCSD

CTU-2018

Table of Contents

[Front End: 2](#_Toc5203625)

[Login Page: 2](#_Toc5203626)

[Client Page: 3](#_Toc5203627)

[Admin Page: 4](#_Toc5203628)

[Back End: 5](#_Toc5203629)

[Code for mainpage.xaml: 5](#_Toc5203630)

[Code for mainpage.xaml.cs: 5](#_Toc5203631)

[Code for LoginPage.xaml: 6](#_Toc5203632)

[Code for LoginPage.xaml.cs 6](#_Toc5203633)

[Code for MenuPg.xaml: 7](#_Toc5203634)

[Code for MenuPg.xaml.cs 8](#_Toc5203635)

[Code of a Class: 9](#_Toc5203636)

[Code of a ViewModel: 10](#_Toc5203637)

This project is back end targeted.

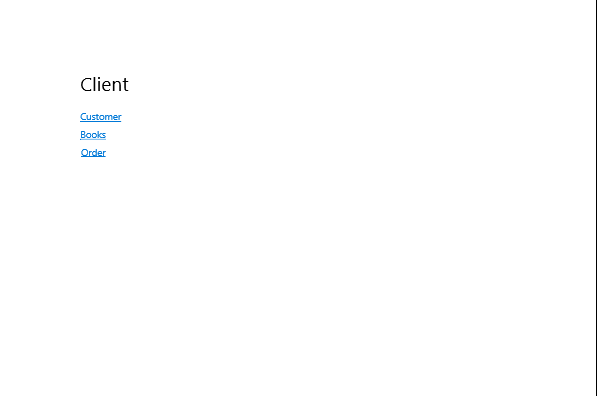
# Front End:

## Login Page:



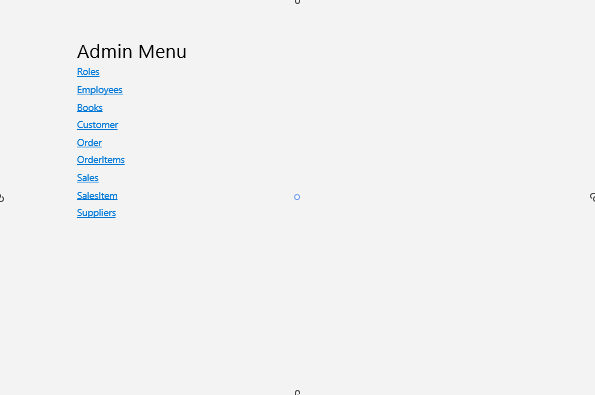
The Login page has a very simple interface. The user has to choose in the combo box what role he wants to access(Admin or Rep). Then he has to click the button to continue to his required destination.

## Client Page:



The Client page has hyperlinks to the only 3 pages clients are allowed to access.

## Admin Page:



The Admin has access to all pages and can change any fields he wishes.

# Back End:

## Code for mainpage.xaml:

<Page

x:Class="TruTextApp.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:TruTextApp"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d">

<StackPanel>

<StackPanel Orientation="Horizontal">

<Button Name="HomeButton" Content="Home" Click="HomeButton\_Click" Margin="0,0,20,0" />

<Button Name="BackButton" Content="Back" Click="BackButton\_Click" Margin="0,0,20,0" />

<Button Name="ForwardButton" Content="Forward" Click="ForwardButton\_Click" Margin="0,0,20,0" />

</StackPanel>

<Frame Name="MyFrame" Height="958">

</Frame>

</StackPanel>

</Page>

## Code for mainpage.xaml.cs:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace TruTextApp

{

/// <summary>

/// An empty page that can be used on its own or navigated to within a Frame.

/// </summary>

public sealed partial class MainPage : Page

{

public MainPage()

{

this.InitializeComponent();

MyFrame.Navigate(typeof(LoginPg));

}

private void HomeButton\_Click(object sender, RoutedEventArgs e)

{

MyFrame.Navigate(typeof(MenuPg));

}

private void BackButton\_Click(object sender, RoutedEventArgs e)

{

if (MyFrame.CanGoBack)

{

MyFrame.GoBack();

}

}

private void ForwardButton\_Click(object sender, RoutedEventArgs e)

{

if (MyFrame.CanGoForward)

{

MyFrame.GoForward();

}

}

}

}

## Code for LoginPage.xaml:

<Page

x:Class="TruTextApp.LoginPg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:TruTextApp"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d">

<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<ComboBox x:Name="Combo" Margin="112,152,0,0" Width="96">

<ComboBoxItem>Admin</ComboBoxItem>

<ComboBoxItem>Rep</ComboBoxItem>

</ComboBox>

<TextBlock HorizontalAlignment="Left" Height="81" Margin="112,64,0,0" Text="Login" FontSize="48" TextWrapping="Wrap" VerticalAlignment="Top" Width="142"/>

<Button Content="Proceed" HorizontalAlignment="Left" Height="34" Margin="112,212,0,0" VerticalAlignment="Top" Width="96" Click="Button\_Click"/>

</Grid>

</Page>

## Code for LoginPage.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace TruTextApp

{

/// <summary>

/// An empty page that can be used on its own or navigated to within a Frame.

/// </summary>

public sealed partial class LoginPg : Page

{

public LoginPg()

{

this.InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if(Combo.SelectedItem.ToString() == "Admin")

{

Frame.Navigate(typeof(MenuPg));

}

else

{

Frame.Navigate(typeof(ClientPg));

}

}

}

}

## Code for MenuPg.xaml:

<Page

x:Class="TruTextApp.MenuPg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:TruTextApp"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d">

<StackPanel>

<TextBlock FontSize="48" Text="Menu" Margin="200,100,20,0"/>

<HyperlinkButton Name ="RolesButton" FontSize="24" Content="Roles" Click="RolesButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name ="EmployeesButton" FontSize="24" Content="Employees" Click="EmployeesButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name="BooksButton" FontSize="24" Content="Books" Click="BooksButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name="CustomerButton" FontSize="24" Content="Customer" Click="CustomerButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name="OrderButton" FontSize="24" Content="Order" Click="OrderButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name="OrderItemsButton" FontSize="24" Content="OrderItems" Click="OrderItemsButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name="SalesButton" FontSize="24" Content="Sales" Click="SalesButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name="SalesItemButton" FontSize="24" Content="SalesItem" Click="SalesItemButton\_Click" Margin="200,0,20,0"/>

<HyperlinkButton Name="SuppliersButton" FontSize="24" Content="Suppliers" Click="SuppliersButton\_Click" Margin="200,0,20,0"/>

</StackPanel>

</Page>

## Code for MenuPg.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace TruTextApp

{

/// <summary>

/// An empty page that can be used on its own or navigated to within a Frame.

/// </summary>

public sealed partial class MenuPg : Page

{

public MenuPg()

{

this.InitializeComponent();

}

private void RolesButton\_Click(object sender, RoutedEventArgs e)

{

Frame.Navigate(typeof(RolePg));

}

private void EmployeesButton\_Click(object sender, RoutedEventArgs e)

{

Frame.Navigate(typeof(EmployeePg));

}

private void BooksButton\_Click(object sender, RoutedEventArgs e)

{

}

private void CustomerButton\_Click(object sender, RoutedEventArgs e)

{

}

private void OrderButton\_Click(object sender, RoutedEventArgs e)

{

}

private void OrderItemsButton\_Click(object sender, RoutedEventArgs e)

{

}

private void SalesButton\_Click(object sender, RoutedEventArgs e)

{

}

private void SalesItemButton\_Click(object sender, RoutedEventArgs e)

{

}

private void SuppliersButton\_Click(object sender, RoutedEventArgs e)

{

}

}

}

## Code of a Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

namespace TruTextApp

{

class EmployeeCl : INotifyPropertyChanged

{

private int \_empId;

public int EmpId

{

get { return this.\_empId; }

set

{

this.\_empId = value;

this.OnPropertyChanged(nameof(EmpId));

}

}

private string \_empName;

public string EmpName

{

get { return this.\_empName; }

set

{

this.\_empName = value;

this.OnPropertyChanged(nameof(EmpName));

}

}

private string \_empSurname;

public string EmpSurname

{

get { return this.\_empSurname; }

set

{

this.\_empSurname = value;

this.OnPropertyChanged(nameof(EmpSurname));

}

}

private int \_roleId;

public int RoleId

{

get { return this.\_roleId; }

set

{

this.\_roleId = value;

this.OnPropertyChanged(nameof(RoleId));

}

}

private string \_email;

public string Email

{

get { return this.\_email; }

set

{

this.\_email = value;

this.OnPropertyChanged(nameof(Email));

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this,

new PropertyChangedEventArgs(propertyName));

}

}

}

}

## Code of a ViewModel:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

using System.Net.Http;

using System.Net.Http.Headers;

using Newtonsoft.Json;

using System.Text.RegularExpressions;

namespace TruTextApp

{

class EmployeeVm : INotifyPropertyChanged /\*\*\*\*\*/

{

private List<EmployeeCl> recordList; /\*\*\*\*\*/

private int recordIndex;

public Command NextRecord { get; private set; }

public Command PreviousRecord { get; private set; }

public Command FirstRecord { get; private set; }

public Command LastRecord { get; private set; }

public Command AddRecord { get; private set; }

public Command EditRecord { get; private set; }

public Command DiscardChanges { get; private set; }

public Command SaveChanges { get; private set; }

private const string ServerUrl = "http://localhost:50000/";

private HttpClient client = null;

public EmployeeVm() /\*\*\*\*\*/

{

this.recordIndex = 0;

this.IsAtStart = true;

this.IsAtEnd = false;

this.NextRecord = new Command(this.Next,

() => {

return this.CanBrowse &&

this.recordList != null && !this.IsAtEnd;

});

this.PreviousRecord = new Command(this.Previous,

() => {

return this.CanBrowse &&

this.recordList != null && !this.IsAtStart;

});

this.FirstRecord = new Command(this.First,

() => {

return this.CanBrowse &&

this.recordList != null && !this.IsAtStart;

});

this.LastRecord = new Command(this.Last,

() => {

return this.CanBrowse &&

this.recordList != null && !this.IsAtEnd;

});

this.AddRecord = new Command(this.Add,

() => { return this.CanBrowse; });

this.EditRecord = new Command(this.Edit,

() => { return this.CanBrowse; });

this.DiscardChanges = new Command(this.Discard,

() => { return this.CanSaveOrDiscardChanges; });

this.SaveChanges = new Command(this.SaveAsync,

() => { return this.CanSaveOrDiscardChanges; });

this.recordList = null;

this.client = new HttpClient();

this.client.BaseAddress = new Uri(ServerUrl);

this.client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

}

#region Methods for fetching and updating data

// Create a new (empty) record

// and put the form into Adding mode

private void Add()

{

EmployeeCl newRecord = new EmployeeCl { EmpId = 0 }; /\*\*\*\*\*2\*/

this.recordList.Insert(recordIndex, newRecord);

this.IsAdding = true;

this.OnPropertyChanged(nameof(Current));

}

// Edit the current record

// - save the existing details of the record

// and put the form into Editing mode

private EmployeeCl oldRecord; /\*\*\*\*\*/

private void Edit()

{

this.oldRecord = new EmployeeCl(); /\*\*\*\*\*/

this.CopyRecord(this.Current, this.oldRecord);

this.IsEditing = true;

}

// Discard changes made while in Adding or Editing mode

// and return the form to Browsing mode

private void Discard()

{

// If the user was adding a new record, then remove it

if (this.IsAdding)

{

this.recordList.Remove(this.Current);

this.OnPropertyChanged(nameof(Current));

}

// If the user was editing an existing record,

// then restore the saved details

if (this.IsEditing)

{

this.CopyRecord(this.oldRecord, this.Current);

}

this.IsBrowsing = true;

this.LastError = String.Empty;

}

// Save the new or updated record back to the web service

// and return the form to Browsing mode

private async void SaveAsync()

{

// Validate the details of the record

if (this.ValidateRecord(this.Current))

{

// Only continue if the record details are valid

this.IsBusy = true;

try

{

// Convert the current record into HTTP request format with a JSON payload

var serializedData = JsonConvert.SerializeObject(this.Current);

StringContent content =

new StringContent(serializedData, Encoding.UTF8, "text/json");

// If the user is adding a new record,

// send an HTTP POST request to the web service with the details

if (this.IsAdding)

{

var response =

await client.PostAsync("api/employees", content); /\*\*\*\*\*/

if (response.IsSuccessStatusCode)

{

// Get the ID of the newly created record and display it

Uri recordUri = response.Headers.Location;

var newRecord = await this.client.GetAsync(recordUri);

if (newRecord.IsSuccessStatusCode)

{

var recordData = await newRecord.Content.ReadAsStringAsync();

this.CopyRecord(

JsonConvert.DeserializeObject<EmployeeCl>(recordData), this.Current); /\*\*\*\*\*/

this.OnPropertyChanged(nameof(Current));

this.IsAdding = false;

this.IsBrowsing = true;

this.LastError = String.Empty;

}

else

{

// Handle GET failure

this.LastError = response.ReasonPhrase;

}

}

// Handle POST failure

else

{

this.LastError = response.ReasonPhrase;

}

}

// The user must be editing an existing record,

// so send the details by using a PUT request

else

{

string path = $"api/employees/{this.Current.EmpId}"; /\*\*\*\*\*2\*/

var response = await client.PutAsync(path, content);

if (response.IsSuccessStatusCode)

{

this.IsEditing = false;

this.IsBrowsing = true;

this.LastError = String.Empty;

}

// Handle PUT failure

else

{

this.LastError = response.ReasonPhrase;

}

}

}

catch (Exception e)

{

// Handle exceptions

this.LastError = e.Message;

}

finally

{

this.IsBusy = false;

}

}

}

// Helper method to validate record details

private bool ValidateRecord(EmployeeCl record) /\*\*\*\*\*M\*/

{

string validationErrors = string.Empty;

bool hasErrors = false;

if (string.IsNullOrWhiteSpace(record.EmpName))

{

hasErrors = true;

validationErrors = "Name must not be empty\n";

}

if (string.IsNullOrWhiteSpace(record.EmpSurname))

{

hasErrors = true;

validationErrors = "Surname must not be empty\n";

}

// Email address is a series of characters that do not include a space or @

// followed by @

// followed by a series of characters that do not include a space or @

// followed by .

// followed by a series of characters that do not include a space or @

Regex emailRegex = new Regex(@"^[^@ ]+@[^@ ]+\.[^@ ]+$");

if (string.IsNullOrWhiteSpace(record.Email) ||

!emailRegex.IsMatch(record.Email))

{

hasErrors = true;

validationErrors += "Invalid Email Address\n";

}

this.LastError = validationErrors;

return !hasErrors;

}

// Utility method for copying the details of a record

private void CopyRecord(EmployeeCl source, EmployeeCl destination) /\*\*\*\*\*M\*/

{

destination.EmpId = source.EmpId;

destination.EmpName = source.EmpName;

destination.EmpSurname = source.EmpSurname;

destination.RoleId = source.RoleId;

destination.Email = source.Email;

}

public async Task GetDataAsync()

{

try

{

this.IsBusy = true;

//await Task.Delay(5000);

var response = await this.client.GetAsync("api/employees"); /\*\*\*\*\*/

if (response.IsSuccessStatusCode)

{

var recordData = await response.Content.ReadAsStringAsync();

this.recordList = JsonConvert.DeserializeObject<List<EmployeeCl>>(recordData); /\*\*\*\*\*/

this.recordIndex = 0;

this.OnPropertyChanged(nameof(Current));

this.IsAtStart = true;

this.IsAtEnd = (this.recordList.Count == 0);

this.LastError = String.Empty;

}

else

{

this.LastError = response.ReasonPhrase;

}

}

catch (Exception e)

{

this.LastError = e.Message;

}

finally

{

this.IsBusy = false;

}

}

#endregion

#region Properties for "busy" and error message handling

private bool \_isBusy;

public bool IsBusy

{

get { return this.\_isBusy; }

set

{

this.\_isBusy = value;

this.OnPropertyChanged(nameof(IsBusy));

}

}

private string \_lastError = null;

public string LastError

{

get { return this.\_lastError; }

private set

{

this.\_lastError = value;

this.OnPropertyChanged(nameof(LastError));

}

}

#endregion

#region Properties for managing the edit mode

// Manage the edit mode of the form - is the user browsing, adding a record, or editing a record

private enum EditMode { Browsing, Adding, Editing };

private EditMode editMode;

public bool IsBrowsing

{

get { return this.editMode == EditMode.Browsing; }

private set

{

if (value)

{

this.editMode = EditMode.Browsing;

}

this.OnPropertyChanged(nameof(IsBrowsing));

this.OnPropertyChanged(nameof(IsAdding));

this.OnPropertyChanged(nameof(IsEditing));

this.OnPropertyChanged(nameof(IsAddingOrEditing));

}

}

public bool IsAdding

{

get { return this.editMode == EditMode.Adding; }

private set

{

if (value)

{

this.editMode = EditMode.Adding;

}

this.OnPropertyChanged(nameof(IsBrowsing));

this.OnPropertyChanged(nameof(IsAdding));

this.OnPropertyChanged(nameof(IsEditing));

this.OnPropertyChanged(nameof(IsAddingOrEditing));

}

}

public bool IsEditing

{

get { return this.editMode == EditMode.Editing; }

private set

{

if (value)

{

this.editMode = EditMode.Editing;

}

this.OnPropertyChanged(nameof(IsBrowsing));

this.OnPropertyChanged(nameof(IsAdding));

this.OnPropertyChanged(nameof(IsEditing));

this.OnPropertyChanged(nameof(IsAddingOrEditing));

}

}

public bool IsAddingOrEditing

{

get { return this.IsAdding || this.IsEditing; }

}

private bool CanBrowse

{

get

{

return this.IsBrowsing &&

this.client != null;

}

}

private bool CanSaveOrDiscardChanges

{

get

{

return this.IsAddingOrEditing &&

this.client != null;

}

}

#endregion

#region Methods and properties for navigation commands

private bool \_isAtStart;

public bool IsAtStart

{

get { return this.\_isAtStart; }

set

{

this.\_isAtStart = value;

this.OnPropertyChanged(nameof(IsAtStart));

}

}

private bool \_isAtEnd;

public bool IsAtEnd

{

get { return this.\_isAtEnd; }

set

{

this.\_isAtEnd = value;

this.OnPropertyChanged(nameof(IsAtEnd));

}

}

public EmployeeCl Current /\*\*\*\*\*/

{

get

{

if (this.recordList != null)

{

return this.recordList[recordIndex];

}

else

{

return null;

}

}

}

private void Next()

{

if (this.recordList.Count - 1 > this.recordIndex)

{

this.recordIndex++;

this.OnPropertyChanged(nameof(Current));

this.IsAtStart = false;

this.IsAtEnd = (this.recordList.Count - 1 == this.recordIndex);

}

}

private void Previous()

{

if (this.recordIndex > 0)

{

this.recordIndex--;

this.OnPropertyChanged(nameof(Current));

this.IsAtEnd = false;

this.IsAtStart = (this.recordIndex == 0);

}

}

private void First()

{

this.recordIndex = 0;

this.OnPropertyChanged(nameof(Current));

this.IsAtStart = true;

this.IsAtEnd = (this.recordList.Count == 0);

}

private void Last()

{

this.recordIndex = this.recordList.Count - 1;

this.OnPropertyChanged(nameof(Current));

this.IsAtEnd = true;

this.IsAtStart = (this.recordList.Count == 0);

}

#endregion

#region INotifyPropertyChanged interface

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

#endregion

}

}