using BirKelimeBirIslem.Dal.Context;

using BirKelimeBirIslem.Model.Entities;

using System;

using System.Collections.Generic;

using System.Text;

using System.Linq;

namespace BirKelimeBirIslem.UI.Win.Classes.Kelime

{

public class KelimeAyarlari

{

#region Properties

Random rnd = new Random();

KelimeAra ara { get; set; }

List<Kelime\_Anlam> kelimelistesi { get; set; }

public List<string> Harfler { get; set; }

#endregion

#region Constructor

public KelimeAyarlari(string \_path)

{

var excel = ConnExcel.Conn();

excel.Yol(\_path);

this.kelimelistesi = new List<Kelime\_Anlam>();

kelimelistesi = Sorgu1(excel);

}

#endregion

#region Functions

public List<Kelime\_Anlam> Sorgu1(ConnExcel \_excel)

{

var list = from a in \_excel.url.Worksheet<Kelime\_Anlam>("Kelime\_Anlam2")

select a;

return list.ToList();

}

public KelimeAra Basla()

{

this.ara = new KelimeAra(kelimelistesi,Harfler);

return this.ara;

}

public void Harf()

{

if (Harfler != null) Harfler.Clear();

this.Harfler = HarflerOlustur();

}

private List<string> HarflerOlustur()

{

List<string> karakter = new List<string>() { "A", "B", "C","Ç", "D", "E", "F", "G", "Ğ", "H", "I", "İ", "J", "K", "L", "M", "N", "O", "Ö", "P", "R", "S", "Ş", "T", "U", "Ü", "V", "Y", "Z" };

List<string> harfler = new List<string>();

for (int i = 0; i < 8; i++)

{

harfler.Add(karakter[rnd.Next(1, karakter.Count)]);

}

return harfler;

}

#endregion

#region ToString

public override string ToString()

{

StringBuilder sb = new StringBuilder();

sb.Append(ara);

return sb.ToString();

}

#endregion

}

}

using BirKelimeBirIslem.Model.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace BirKelimeBirIslem.UI.Win.Classes.Kelime

{

public class KelimeAra

{

#region Properties

private List<int> harfmiktar { get; set; }

public int puan;

private int max { get; set; }

internal List<Kelime\_Anlam> bulunankelimeler { get; private set; }

#endregion

#region Constructor

public KelimeAra(List<Kelime\_Anlam> \_liste, List<string> \_harfler)

{

int adet = 0;

int sira = -1;

max = 0;

harfmiktar = new List<int>();

bulunankelimeler = new List<Kelime\_Anlam>();

List<Kelime\_Anlam> aramasonuc = new List<Kelime\_Anlam>();

foreach (var item in \_liste)

{

string kelime = item.Kelime.ToUpper().Trim().ToString();

adet = 0;

foreach (var item2 in \_harfler)

{

if (kelime.Count() == 0) break;

sira = kelime.IndexOf(item2);

if (sira != -1)

{

adet++;

kelime = kelime.Remove(sira, 1);

sira = -1;

}

}

if (kelime.Count() <= 1)

{

if (kelime.Count() == 1) adet++;

if (adet > max) max = adet;

harfmiktar.Add(adet);

aramasonuc.Add(new Kelime\_Anlam { Kelime\_ID = item.Kelime\_ID, Kelime = item.Kelime, Anlam = item.Anlam });

}

}

VerileriIsle(aramasonuc);

}

#endregion

#region Functions

private void VerileriIsle(List<Kelime\_Anlam> \_liste)

{

foreach (var item in \_liste.Select((value, i) => new { i, value }))

{

if (harfmiktar[item.i] == max)

{

bulunankelimeler.Add(new Kelime\_Anlam { Kelime\_ID = item.value.Kelime\_ID, Kelime = item.value.Kelime, Anlam = item.value.Anlam });

}

}

}

#endregion

#region ToString

public override string ToString()

{

StringBuilder sb = new StringBuilder();

foreach (var item in bulunankelimeler)

{

sb.AppendLine(item.Kelime + " --> " + item.Anlam);

}

string ifade;

switch (max)

{

case 9:

puan = 15;

ifade = "Alınan Puan: 15" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 8:

puan = 11;

ifade = "Alınan Puan: 11" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 7:

puan = 9;

ifade = "Alınan Puan: 9" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 6:

puan = 7;

ifade = "Alınan Puan: 7" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 5:

puan = 5;

ifade = "Alınan Puan: 5" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 4:

puan = 4;

ifade = "Alınan Puan: 4" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 3:

puan = 3;

ifade = "Alınan Puan: 3" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 2:

puan = 1;

ifade = "Alınan Puan: 1" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

default:

puan = 0;

ifade = "Alınan Puan: 0" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

}

sb.Append(ifade);

return sb.ToString();

}

#endregion

}

}

using BirKelimeBirIslem.UI.Win.Forms.Base;

using BirKelimeBirIslem.UI.Win.Classes.Kelime;

using DevExpress.XtraEditors;

using System;

using System.Windows.Forms;

namespace BirKelimeBirIslem.UI.Win.Forms.KelimeForm

{

public partial class KelimeKarti : BaseForm

{

public KelimeKarti()

{

InitializeComponent();

btnBasla.ItemClick += BtnBasla\_ItemClick;

btnOlustur.ItemClick += BtnOlustur\_ItemClick;

tglGiris.ItemClick += TglGiris\_ItemClick;

btnDosya.ItemClick += BtnDosya\_ItemClick;

}

KelimeAyarlari kelimeayarlari;

XtraOpenFileDialog dosya = new XtraOpenFileDialog();

private DateTime \_startTime = DateTime.Now;

private TimeSpan \_timeSpan = new TimeSpan(0, 1, 0); // 1 minute 10 seconds as total time

System.Diagnostics.Stopwatch st = new System.Diagnostics.Stopwatch();

private bool VeriKontrol()

{

if (kelimeayarlari == null)

{

XtraMessageBox.Show("Veri Seçimi Yapınız!", "Bilgilendirme", MessageBoxButtons.OK);

return false;

}

else

{

if (string.IsNullOrEmpty(txtHarf1.Text)) txtHarf1.Text = "A";

if (string.IsNullOrEmpty(txtHarf2.Text)) txtHarf2.Text = "A";

if (string.IsNullOrEmpty(txtHarf3.Text)) txtHarf3.Text = "A";

if (string.IsNullOrEmpty(txtHarf4.Text)) txtHarf4.Text = "A";

if (string.IsNullOrEmpty(txtHarf5.Text)) txtHarf5.Text = "A";

if (string.IsNullOrEmpty(txtHarf6.Text)) txtHarf6.Text = "A";

if (string.IsNullOrEmpty(txtHarf7.Text)) txtHarf7.Text = "A";

if (string.IsNullOrEmpty(txtHarf8.Text)) txtHarf8.Text = "A";

return true;

}

}

private void KelimeKarti\_Load(object sender, System.EventArgs e)

{

lblSure.Caption = "00:00";

}

private void TglGiris\_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)

{

try

{

if (tglGiris.Checked)

{

txtHarf1.ReadOnly = false;

txtHarf2.ReadOnly = false;

txtHarf3.ReadOnly = false;

txtHarf4.ReadOnly = false;

txtHarf5.ReadOnly = false;

txtHarf6.ReadOnly = false;

txtHarf7.ReadOnly = false;

txtHarf8.ReadOnly = false;

}

else

{

txtHarf1.ReadOnly = true;

txtHarf2.ReadOnly = true;

txtHarf3.ReadOnly = true;

txtHarf4.ReadOnly = true;

txtHarf5.ReadOnly = true;

txtHarf6.ReadOnly = true;

txtHarf7.ReadOnly = true;

txtHarf8.ReadOnly = true;

}

}

catch (Exception)

{

throw;

}

}

private void BtnBasla\_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)

{

try

{

if (VeriKontrol())

{

lstKelime.Items.Clear();

kelimeayarlari.Harfler[0] = txtHarf1.Text;

kelimeayarlari.Harfler[1] = txtHarf2.Text;

kelimeayarlari.Harfler[2] = txtHarf3.Text;

kelimeayarlari.Harfler[3] = txtHarf4.Text;

kelimeayarlari.Harfler[4] = txtHarf5.Text;

kelimeayarlari.Harfler[5] = txtHarf6.Text;

kelimeayarlari.Harfler[6] = txtHarf7.Text;

kelimeayarlari.Harfler[7] = txtHarf8.Text;

st.Start();

var ifd = kelimeayarlari.Basla();

st.Stop();

lblSure.Caption = st.Elapsed.Seconds + ":" + st.Elapsed.Milliseconds;

lstKelime.Items.Add(ifd.ToString());

lblPuan.Caption = ifd.puan.ToString();

//lstKelime.MakeItemVisible(1);

st.Reset();

}

}

catch (Exception)

{

throw;

}

}

private void BtnOlustur\_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)

{

try

{

if (VeriKontrol())

{

kelimeayarlari.Harf();

lstKelime.Items.Clear();

txtHarf1.Text = kelimeayarlari.Harfler[0].ToString();

txtHarf2.Text = kelimeayarlari.Harfler[1].ToString();

txtHarf3.Text = kelimeayarlari.Harfler[2].ToString();

txtHarf4.Text = kelimeayarlari.Harfler[3].ToString();

txtHarf5.Text = kelimeayarlari.Harfler[4].ToString();

txtHarf6.Text = kelimeayarlari.Harfler[5].ToString();

txtHarf7.Text = kelimeayarlari.Harfler[6].ToString();

txtHarf8.Text = kelimeayarlari.Harfler[7].ToString();

}

}

catch (Exception)

{

throw;

}

}

private void BtnDosya\_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)

{

try

{

dosya.Filter = "Excel Dosyası |\*.xls;\*.xlsx";

dosya.Title = "Excel Dosyası Seçiniz.";

dosya.FilterIndex = 2;

dosya.RestoreDirectory = true;

dosya.ShowDialog();

if (dosya.FileName != "")

{

kelimeayarlari = new KelimeAyarlari(dosya.FileName);

if (kelimeayarlari != null) XtraMessageBox.Show("Veriler Başarıyla Yüklendi", "Bilgilendirme", MessageBoxButtons.OK);

else XtraMessageBox.Show("Veri Yükleme Hatası!", "Bilgilendirme", MessageBoxButtons.OK);

}

else XtraMessageBox.Show("Dosya Seçilmedi!", "Bilgilendirme", MessageBoxButtons.OK);

}

catch (Exception)

{

throw;

}

}

}

}