הסברים קצרים :

# Form close open

public partial class programForm : Form

{

private Rectangle[] controlerOriginalRectangle;

// save pointer to controls

private Control[] controls;

private Rectangle originalFormSize;

public programForm()

{

InitializeComponent();

}

private void programForm\_Load(object sender, EventArgs e)

{

Login login = new Login();

login.MdiParent = this;

login.Location = new Point(0, 0);

login.Dock = DockStyle.Fill;

login.Show();

HelpFunc.Form\_LoadCreateRectangles(ref originalFormSize, ref controls, ref controlerOriginalRectangle, this);

}

private void ProgramForm\_Resize(object sender, EventArgs e)

{

// loop over controls and updates values

HelpFunc.Form\_Resize(controls, controlerOriginalRectangle,originalFormSize,this);

}

}

public partial class Login : Form

{

// save values

private Rectangle[] controlerOriginalRectangle;

// save pointer to controls

private Control[] controls;

private Rectangle originalFormSize;

private UserLogic userLogic;

public Login()

{

InitializeComponent();

userLogic = new UserLogic();

}

private void Login\_Load(object sender, EventArgs e)

{

HelpFunc.addImgCursor("Move.png", new Size(80, 80), specialButton1);

HelpFunc.Form\_LoadCreateRectangles(ref originalFormSize, ref controls, ref controlerOriginalRectangle, this);

}

private void Form1\_Resize(object sender, EventArgs e)

{

HelpFunc.Form\_Resize(controls, controlerOriginalRectangle, originalFormSize, this);

}

private void specialButton1\_Click(object sender, EventArgs e)

{

string? checkValues;

checkValues = Validation\_CheckUser.checkId(id.Text);

HelpFunc.checkAndSetError(id,checkValues,errorId);

checkValues = Validation\_CheckUser.checkEmail(email.Text);

HelpFunc.checkAndSetError(email, checkValues, errorEmail);

checkValues = Validation\_CheckUser.checkPassword(password.Text);

HelpFunc.checkAndSetError(password, checkValues, errorPassword);

if (checkValues == null)

{

object resFun = userLogic.ShowFromUser\_UserFromSpecific\_Id\_Email\_Password(id.Text, email.Text, password.Text);

if (resFun.GetType() != typeof(DataTable))

{

MessageBox.Show(resFun.ToString());

}

else

{

DataTable dt = (DataTable)resFun;

User user = new User();

foreach (DataRow row in dt.Rows)

{

user = new User()

{

Id = row["id"].ToString()!,

Email = row["email"].ToString()!,

Password = row["password"].ToString()!,

FirstName = row["FirstName"].ToString()!,

LastName = row["LastName"].ToString()!,

Type = (bool)row["type"]

};

}

Main main = new Main(user);

main.Location = new Point((this.MdiParent.Width) / 2, (this.MdiParent.Height) / 2);

main.Activate();

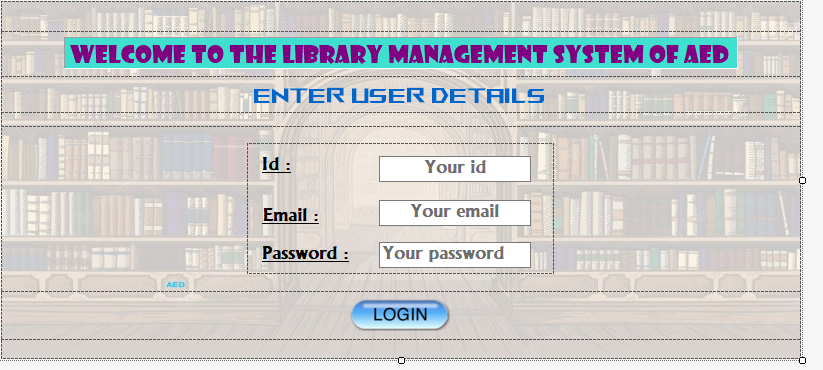
main.Show();

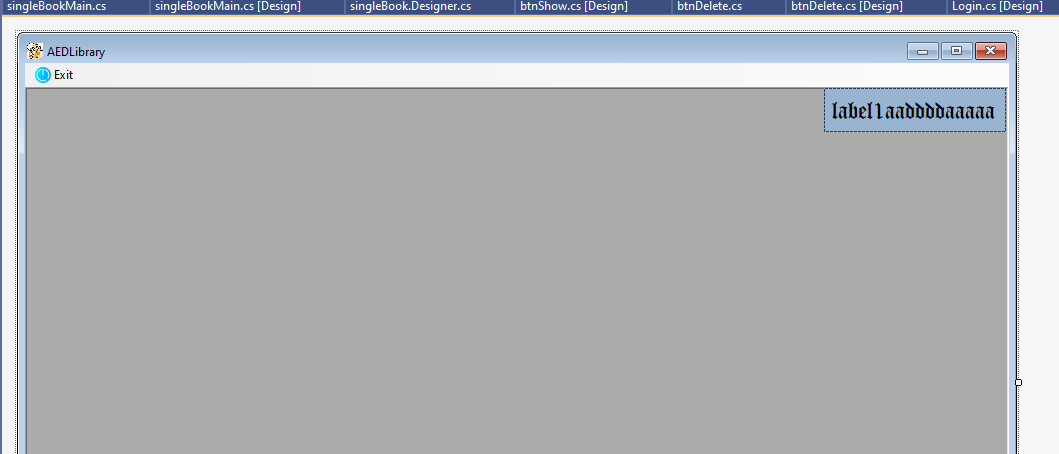
this.MdiParent.Hide();

}

}

}





# Tool strip

public Main(User \_user)

{

InitializeComponent();

HelpFunc.Form\_LoadCreateRectangles(ref originalFormSize, ref controls, ref controlerOriginalRectangle, this);

user = \_user;

}

private void Main\_Load(object sender, EventArgs e)

{

nameUser.Text = user.FirstName+ " " + user.LastName;

HelpFunc.playSound(@"appSong.wav");

// rules -

// Library employees can access everything

//// A library subscriber can access:

//Book : show

//Borrow : show only of him

#region createBasicMenu

#region createMenuApp

//

// menuApp

//

menuApp.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

showToolStripMenuItem,

showToolStripMenuItem1});

#endregion

#region ToolStripMenu - Book

//

// ToolStripMenu - Book

//

this.showToolStripMenuItem.Image = global::AppLayer.Properties.Resources.BookIcon;

this.showToolStripMenuItem.Name = "showToolStripMenuItem";

this.showToolStripMenuItem.Size = new System.Drawing.Size(67, 20);

this.showToolStripMenuItem.Text = "Books";

this.showToolStripMenuItem.DropDownItems.Add(this.showToolStripMenuItem5);

//

// Show

//

this.showToolStripMenuItem5.Name = "showToolStripMenuItem5";

this.showToolStripMenuItem5.Size = new System.Drawing.Size(180, 22);

this.showToolStripMenuItem5.Text = "Show";

this.showToolStripMenuItem5.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.showAllToolStripMenuItem1,

this.showSearchToolStripMenuItem1}); ;

//

if (true)

{

#region createAdvanceMenu

#region addToMenu

//

// menuApp

//

menuApp.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

showToolStripMenuItem2,

showToolStripMenuItem3});

#endregion

#region ToolStripMenu - Book

//

// ToolStripMenu - Book

//

this.showToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.addToolStripMenuItem,

this.showToolStripMenuItem4,

this.showToolStripMenuItem5,

this.updateToolStripMenuItem});

//

// Add

//

this.addToolStripMenuItem.Name = "addToolStripMenuItem";

this.addToolStripMenuItem.Size = new System.Drawing.Size(180, 22);

this.addToolStripMenuItem.Text = "Add";

this.addToolStripMenuItem.Click += new System.EventHandler(this.MenuItem\_Click);

//

// Delete

//

this.showToolStripMenuItem4.Name = "showToolStripMenuItem4";

this.showToolStripMenuItem4.Size = new System.Drawing.Size(180, 22);

this.showToolStripMenuItem4.Text = "Delete";

this.showToolStripMenuItem4.Click += new System.EventHandler(this.MenuItem\_Click);

//

// Update

//

this.updateToolStripMenuItem.Name = "updateToolStripMenuItem";

this.updateToolStripMenuItem.Size = new System.Drawing.Size(180, 22);

this.updateToolStripMenuItem.Text = "Update";

this.updateToolStripMenuItem.Click += new System.EventHandler(this.MenuItem\_Click);

#endregion

#region ToolStripMenu - Borrow

//

// ToolStripMenu - Borrow

//

this.showToolStripMenuItem1.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.addToolStripMenuItem1,

this.showToolStripMenuItem6,

this.showToolStripMenuItem7,

this.updateToolStripMenuItem1});

private void MenuItem\_Click(object sender, EventArgs e)

{

FormCollection FormsOpen = Application.OpenForms;

for (int i = 0; i < FormsOpen.Count; i++)

{

if (FormsOpen[i].Name != "Main")

FormsOpen[i].Close();

}

ToolStripMenuItem menuStrip = (ToolStripMenuItem)sender;

ToolStripItem parent = menuStrip.OwnerItem;

// show all example in show

if (parent.Text != "Books" && parent.Text != "Borrow" && parent.Text != "Categories" && parent.Text != "Users")

{

parent = parent.OwnerItem;

}

switch (parent.Text)

{

case "Books":

AreaBook book = new AreaBook(menuStrip.Text,null);

book.MdiParent = this;

book.Activate();

book.Show();

book.Size = new Size(this.Width - 100, this.Height - 150);

book.Location = new Point((this.Width - book.Width) / 2 - 10, (this.Height - book.Height) / 2 - 30);

break;

case "Borrow":

AreaBorrow borrow = new AreaBorrow(menuStrip.Text);

borrow.MdiParent = this;

borrow.Activate();

borrow.Show();

borrow.Size = new Size(this.Width - 100, this.Height - 150);

borrow.Location = new Point((this.Width - borrow.Width) / 2 - 10, (this.Height - borrow.Height) / 2 - 30);

break;

case "Categories":

AreaExistingCategories existingCategories = new AreaExistingCategories(menuStrip.Text);

existingCategories.MdiParent = this;

existingCategories.Activate();

existingCategories.Show();

existingCategories.Size = new Size(this.Width - 100, this.Height - 150);

existingCategories.Location = new Point((this.Width - existingCategories.Width) / 2 - 10, (this.Height - existingCategories.Height) / 2 - 30);

break;

case "Users":

AreaUser user = new AreaUser(menuStrip.Text);

user.MdiParent = this;

user.Activate();

user.Show();

user.Size = new Size(this.Width - 100, this.Height - 150);

user.Location = new Point((this.Width - user.Width) / 2 - 10, (this.Height - user.Height) / 2 - 30);

break;

}

}

private void Main\_Resize(object sender, EventArgs e)

{

HelpFunc.Form\_Resize(controls, controlerOriginalRectangle, originalFormSize, this);

}

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

# SpecialButton

using System.Windows.Forms;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.ComponentModel;

namespace AppLayer.SpecialComponents

{

public class SpecialButton : Button

{

//Fields

private int borderSize = 0;

private int borderRadius = 20;

private Color borderColor = Color.PaleVioletRed;

//Constructor

public SpecialButton()

{

this.FlatStyle = FlatStyle.Flat;

this.FlatAppearance.BorderSize = 0;

this.Size = new Size(150, 40);

this.BackColor = Color.MediumSlateBlue;

this.ForeColor = Color.White;

this.Resize += new EventHandler(Button\_Resize);

}

private void Button\_Resize(object sender, EventArgs e)

{

if (borderRadius > this.Height)

borderRadius = this.Height;

}

//Properties

[Category("SpecialButton Get\_Set\_Fun")]

public int BorderSize

{

get { return borderSize; }

set

{

borderSize = value;

// Invalidates the entire surface of the control and causes the control to be redrawn.

this.Invalidate();

}

}

[Category("SpecialButton Get\_Set\_Fun")]

public int BorderRadius

{

get { return borderRadius; }

set

{

borderRadius = value;

this.Invalidate();

}

}

[Category("SpecialButton Get\_Set\_Fun")]

public Color BorderColor

{

get { return borderColor; }

set

{

borderColor = value;

this.Invalidate();

}

}

[Category("SpecialButton Get\_Set\_Fun")]

public Color BackgroundColor

{

get { return this.BackColor; }

set { this.BackColor = value; }

}

[Category("SpecialButton Get\_Set\_Fun")]

public Color TextColor

{

get { return this.ForeColor; }

set { this.ForeColor = value; }

}

//Methods

private GraphicsPath GetFigurePath(Rectangle rect, float radius)

{

GraphicsPath path = new GraphicsPath();

float curveSize = radius \* 2F;

path.StartFigure();

path.AddArc(rect.X, rect.Y, curveSize, curveSize, 180, 90);

path.AddArc(rect.Right - curveSize, rect.Y, curveSize, curveSize, 270, 90);

path.AddArc(rect.Right - curveSize, rect.Bottom - curveSize, curveSize, curveSize, 0, 90);

path.AddArc(rect.X, rect.Bottom - curveSize, curveSize, curveSize, 90, 90);

path.CloseFigure();

return path;

}

protected override void OnPaint(PaintEventArgs pevent)

{

base.OnPaint(pevent);

Rectangle rectSurface = this.ClientRectangle;

Rectangle rectBorder = Rectangle.Inflate(rectSurface, -borderSize, -borderSize);

int smoothSize = 2;

if (borderSize > 0)

smoothSize = borderSize;

if (borderRadius > 2) //Rounded button

{

using (GraphicsPath pathSurface = GetFigurePath(rectSurface, borderRadius))

using (GraphicsPath pathBorder = GetFigurePath(rectBorder, borderRadius - borderSize))

using (Pen penSurface = new Pen(this.Parent.BackColor, smoothSize))

using (Pen penBorder = new Pen(borderColor, borderSize))

{

pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;

//Button surface

this.Region = new Region(pathSurface);

//Draw surface border for HD result

pevent.Graphics.DrawPath(penSurface, pathSurface);

//Button border

if (borderSize >= 1)

//Draw control border

pevent.Graphics.DrawPath(penBorder, pathBorder);

}

}

else //Normal button

{

pevent.Graphics.SmoothingMode = SmoothingMode.None;

//Button surface

this.Region = new Region(rectSurface);

//Button border

if (borderSize >= 1)

{

using (Pen penBorder = new Pen(borderColor, borderSize))

{

penBorder.Alignment = PenAlignment.Inset;

pevent.Graphics.DrawRectangle(penBorder, 0, 0, this.Width - 1, this.Height - 1);

}

}

}

}

protected override void OnHandleCreated(EventArgs e)

{

base.OnHandleCreated(e);

this.Parent.BackColorChanged += new EventHandler(Container\_BackColorChanged);

}

private void Container\_BackColorChanged(object sender, EventArgs e)

{

this.Invalidate();

}

}

}

# CheckComboBox

using System.Windows.Forms.VisualStyles;

namespace AppLayer.SpecialComponents

{

/// <summary>

/// as ComboBox

/// Create functions to DrawItem and SelectedIndexChanged events

/// Creates the combo box drop-down

/// The contents of the dropdown are rendered using the

/// CheckBoxRenderer class.

/// The information of the combo box is updated according to the CheckComboBox\_DrawItem() in our class

/// </summary>

public partial class CheckComboBox : ComboBox

{

public CheckComboBox()

{

this.DrawMode = DrawMode.OwnerDrawFixed;

this.DrawItem += new DrawItemEventHandler(CheckComboBox\_DrawItem);

this.SelectedIndexChanged += new EventHandler(CheckComboBox\_SelectedIndexChanged);

}

void CheckComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

CheckComboBoxItem item = (CheckComboBoxItem)SelectedItem;

item.CheckState = !item.CheckState;

if (CheckStateChanged != null)

CheckStateChanged(item, e);

}

//Will fire when the list updates its content

void CheckComboBox\_DrawItem(object sender, DrawItemEventArgs e)

{

// make sure the index is valid (sanity check)

if (e.Index == -1)

{

return;

}

// test the item to see if its a CheckComboBoxItem

if (!(Items[e.Index] is CheckComboBoxItem))

{

// it's not, so just render it as a default string

e.Graphics.DrawString(

Items[e.Index].ToString(),

this.Font,

Brushes.Black,

new Point(e.Bounds.X, e.Bounds.Y));

return;

}

// get the CheckComboBoxItem from the collection

CheckComboBoxItem box = (CheckComboBoxItem)Items[e.Index];

// render it

CheckBoxRenderer.RenderMatchingApplicationState = true;

CheckBoxRenderer.DrawCheckBox(

e.Graphics,

new Point(e.Bounds.X, e.Bounds.Y),

e.Bounds,

box.Text,

this.Font,

(e.State & DrawItemState.Focus) == 0,

box.CheckState ? CheckBoxState.CheckedNormal : CheckBoxState.UncheckedNormal);

}

/// will run when we change the check box item in the drop-down list

public event EventHandler CheckStateChanged;

}

}

using System.ComponentModel;

namespace AppLayer.SpecialComponents

{

/// <summary>

/// from list items to combo box

/// </summary>

public class CheckComboBoxItem

{

public CheckComboBoxItem(string text, bool initialCheckState)

{

\_checkState = initialCheckState;

\_text = text;

}

#region Get and Set to Properties

//Properties - CheckState

[Category("CheckComboBoxItem Get\_Set\_Fun")]

private bool \_checkState = false;

public bool CheckState

{

get { return \_checkState; }

set { \_checkState = value; }

}

//Properties - Text

[Category("CheckComboBoxItem Get\_Set\_Fun")]

private string \_text = "";

public string Text

{

get { return \_text; }

set { \_text = value; }

}

//Properties - Tag

[Category("CheckComboBoxItem Get\_Set\_Fun")]

private object \_tag = null;

public object Tag

{

get { return \_tag; }

set { \_tag = value; }

}

// Happens after selecting CheckComboBoxItem from the list items

public override string ToString()

{

//return Text;

return "Select Search Options";

}

#endregion

}

}

# DB

public class UserFunc

{

private static UserLogic userLogic = new UserLogic();

public static void deleteSelectedUser(string id)

{

MessageBox.Show(userLogic.deleteSelectedUser(id));

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using BusinessLogicLayer.StoredProceduresLogic;

namespace AppLayer.SpecialComponents

{

public static class ExistingCategoriesFun

{

private static ExistingCategorylogic existingCategorylogic = new ExistingCategorylogic();

public static void createCategories(List<string> categories, ComboBox category)

{

object resFun = existingCategorylogic.getExistingCategories();

if (resFun.GetType() != typeof(DataTable))

{

MessageBox.Show(resFun.ToString());

}

else

{

DataTable dt = (DataTable)resFun;

foreach (DataRow row in dt.Rows)

{

categories.Add((string)row["Category"]);

}

}

category.DataSource = categories;

}

public static void category\_SelectedIndexChanged(List<string> secondaryCategorySelect, ComboBox secondaryCategory, string choose)

{

secondaryCategory.DataSource = null;

secondaryCategorySelect.Clear();

object resFun = existingCategorylogic.ShowFromExistingCategories\_SubcategoryFromCategory(choose);

if (resFun.GetType() != typeof(DataTable))

{

MessageBox.Show(resFun.ToString());

}

else

{

secondaryCategorySelect.Add("No secondary category");

DataTable dt = (DataTable)resFun;

foreach (DataRow row in dt.Rows)

{

secondaryCategorySelect.Add((string)row["secondaryCategory"]);

}

}

secondaryCategory.DataSource = secondaryCategorySelect;

}

public static void deleteSelectedExistingCategory(DataAccessLayer.Entities.ExistingCategory existingCategory)

{

MessageBox.Show(existingCategorylogic.deleteSelectedExistingCategory(existingCategory));

}

}

}

# Display

public static void playSound(string urlSound)

{

SoundPlayer simpleSound = new SoundPlayer(urlSound);

simpleSound.Stop();

simpleSound.PlayLooping();

}

private static void resizeControl(Rectangle r, Control c, Rectangle originalFormSize, object thisObj)

{

float xRatio;

float yRatio;

if(thisObj == null)

{

return;

}

else if (thisObj.GetType().BaseType.Name == "Form")

{

Form thisForm = (Form)thisObj;

xRatio = (float)(thisForm.Width) / (float)(originalFormSize.Width);

yRatio = (float)(thisForm.Height) / (float)(originalFormSize.Height);

}

else if (thisObj.GetType().BaseType.Name == "UserControl")

{

UserControl thisUC = (UserControl)thisObj;

xRatio = (float)(thisUC.Width) / (float)(originalFormSize.Width);

yRatio = (float)(thisUC.Height) / (float)(originalFormSize.Height);

}

else

{

return;

}

int newX = (int)(r.Location.X \* xRatio);

int newY = (int)(r.Location.Y \* yRatio);

int newWidth = (int)(r.Width \* xRatio);

int newHeight = (int)(r.Height \* yRatio);

c.Location = new Point(newX, newY);

c.Size = new Size(newWidth, newHeight);

}

public static void Form\_Resize(Control[] controls , Rectangle [] controlerOriginalRectangle,Rectangle originalFormSize,object thisObj)

{

if(controls != null)

{

// loop over controls and updates values

foreach (var (control, index) in controls.Select((value, i) => (value, i)))

{

resizeControl(controlerOriginalRectangle[index], control, originalFormSize, thisObj);

}

}

}

public static void addImgCursor(string url, Size size, Control control)

{

Bitmap bitmap = new Bitmap(new Bitmap(url), size);

control.Cursor = new Cursor(bitmap.GetHicon());

}

public static void Form\_LoadCreateRectangles(ref Rectangle originalFormSize, ref Control[] controls, ref Rectangle[] controlerOriginalRectangle, object thisObj)

{

if (thisObj.GetType().BaseType.Name == "Form")

{

Form thisForm = (Form)thisObj;

originalFormSize = new Rectangle(thisForm.Location.X, thisForm.Location.Y, thisForm.Size.Width, thisForm.Size.Height);

controlerOriginalRectangle = new Rectangle[thisForm.Controls.Count];

controls = new Control[thisForm.Controls.Count];

// copy all collection to array from 0

thisForm.Controls.CopyTo(controls, 0);

}

else if(thisObj.GetType().BaseType.Name == "UserControl")

{

UserControl thisForm = (UserControl)thisObj;

originalFormSize = new Rectangle(thisForm.Location.X, thisForm.Location.Y, thisForm.Size.Width, thisForm.Size.Height);

controlerOriginalRectangle = new Rectangle[thisForm.Controls.Count];

controls = new Control[thisForm.Controls.Count];

// copy all collection to array from 0

thisForm.Controls.CopyTo(controls, 0);

}

else

{

return ;

}

//// Loop over tuples with the item and its index

foreach (var (control, index) in controls.Select((value, i) => (value, i)))

{

controlerOriginalRectangle[index] = new Rectangle(control.Location.X, control.Location.Y, control.Width, control.Height);

}

}

public static void cbxDesign\_DrawItem(ref object sender, ref DrawItemEventArgs e)

{

// By using Sender, one method could handle multiple ComboBoxes

ComboBox cbx = sender as ComboBox;

if (cbx != null)

{

// Always draw the background

e.DrawBackground();

// Drawing one of the items?

if (e.Index >= 0)

{

// Set the string alignment. Choices are Center, Near and Far

StringFormat sf = new StringFormat();

sf.LineAlignment = StringAlignment.Center;

sf.Alignment = StringAlignment.Center;

// Set the Brush to ComboBox ForeColor to maintain any ComboBox color settings

// Assumes Brush is solid

Brush brush = new SolidBrush(cbx.ForeColor);

// If drawing highlighted selection, change brush

if ((e.State & DrawItemState.Selected) == DrawItemState.Selected)

brush = SystemBrushes.HighlightText;

// Draw the string

e.Graphics.DrawString(cbx.Items[e.Index].ToString(), cbx.Font, brush, e.Bounds, sf);

}

}

}

# UCS

public static void hideAndShowUC(UserControl[] ucs, string kindAction,Form form)

{

if(ucs.Length != 5)

{

MessageBox.Show("The array must contain 4 UC (add, delete, showSearch, showAll, update)");

return;

}

foreach (UserControl uc in ucs)

{

uc.Size = new Size(uc.Parent.Width - 50, uc.Height);

uc.Location = new Point((form.Width - uc.Width) / 2 - 10, (form.Height - uc.Height) / 2 - 30);

uc.Hide();

}

switch (kindAction)

{

case "Add":

ucs[0].Show();

break;

case "Delete":

ucs[1].Show();

break;

case "Show All":

ucs[2].Show();

break;

case "Show Search":

ucs[3].Show();

break;

case "Update":

ucs[4].Show();

break;

}

}

public static void createCheckComboBoxList(string[] fieldsName,CheckComboBox checkComboBox1,Control[] controls, Control panelShow)

{

CheckComboBoxItem[] checkComboBoxItems = new CheckComboBoxItem[fieldsName.Length];

foreach (var (field, index) in fieldsName.Select((field, index) => (field, index)))

{

checkComboBoxItems[index] = new CheckComboBoxItem(field, false);

}

checkComboBox1.Items.AddRange(checkComboBoxItems);

//// wire up the check state changed event

checkComboBox1.CheckStateChanged += new System.EventHandler((sender, e) => checkComboBox1\_CheckStateChanged(sender,e,controls,checkComboBox1.Items.Cast<CheckComboBoxItem>().ToArray(),panelShow));

}

private static void showAll(CheckComboBoxItem[] checkComboBox, Boolean showAll)

{

foreach (CheckComboBoxItem item in checkComboBox)

{

if (item.Text.ToLower() != "all")

{

switch (showAll)

{

case true:

{

item.CheckState = true;

}

break;

case false:

item.CheckState = false;

break;

}

}

}

}

private static void checkComboBox1\_CheckStateChanged(object sender, EventArgs e,Control[] controls, CheckComboBoxItem[] checkComboBoxItems, Control panelShow)

{

if (sender is CheckComboBoxItem)

{

CheckComboBoxItem item = (CheckComboBoxItem)sender;

MessageBox.Show(item.Text);

MessageBox.Show(item.CheckState.ToString());

if(item.Text.ToLower() == "all")

{

showAll(checkComboBoxItems,item.CheckState);

}

foreach(Control control in controls)

{

if (item.Text.ToLower() == "all")

{

switch (item.CheckState)

{

case true:

control.Show();

control.Tag = "show";

break;

case false:

control.Hide();

control.Tag = "hide";

break;

}

}

// name control must start with panel

// name CheckComboBoxItem maybe have space

else if (control.Name.ToLower().Split("panel")[1] == item.Text.Replace(" ", "").ToLower())

{

if (item.CheckState)

{

control.Show();

control.Tag = "show";

}

if (!item.CheckState)

{

control.Hide();

control.Tag = "hide";

}

foreach (CheckComboBoxItem removeMark in checkComboBoxItems)

{

if (removeMark.Text.ToLower() == "all" && removeMark.CheckState)

{

removeMark.CheckState = false;

}

}

}

}

//foreach

//switch (item.Text)

//{

// case "One":

// //checkBox1.Checked = item.CheckState;

// break;

// case "Two":

// //checkBox2.Checked = item.CheckState;

// break;

// case "Three":

// //checkBox3.Checked = item.CheckState;

// break;

//}

int countControlShow = 0;

foreach(Control control in controls)

{

if(control.Name.ToLower() != "panelaction" && control.Tag.ToString() == "show")

{

countControlShow++;

}

}

if(countControlShow > 0)

{

panelShow.Show();

}

else

{

panelShow.Hide();

}

}

}

public static void checkAndSetError(Control insertErrNext, string? checkRes,ErrorProvider err)

{

if (checkRes != null)

{

err.SetError(insertErrNext, checkRes);

}

else

{

err.SetError(insertErrNext, String.Empty);

}

}

# DGet Data

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using DataAccessLayer.Entities;

namespace AppLayer.Components.Single

{

public partial class getData : Form

{

string conStrin = @"Data Source=.;Initial Catalog=Library;Integrated Security=True";

SqlDataAdapter sda;

SqlCommandBuilder scb;

DataTable dt;

public getData()

{

InitializeComponent();

sda = new SqlDataAdapter();

using (SqlConnection sqlConnection = new SqlConnection(conStrin))

{

SqlCommand cmd = new SqlCommand("getUsers", sqlConnection);

try

{

if (sqlConnection.State != ConnectionState.Open)

sqlConnection.Open();

// call to procedure that get books

//cmd.Parameters.Add(new SqlParameter("@newDay\_Date", dayAdd.date));

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@ERROR", SqlDbType.NVarChar, 500);

cmd.Parameters["@ERROR"].Direction = ParameterDirection.Output;

SqlDataReader dr = cmd.ExecuteReader();

//cmd.ExecuteNonQuery();

//sqlConnection.Close();

// Check if was problem with the command

if (cmd.Parameters["@ERROR"].Value != null && cmd.Parameters["@ERROR"].Value.ToString().Length > 0)

{

string message = (string)cmd.Parameters["@ERROR"].Value;

MessageBox.Show(message);

}

else if (dr.HasRows)

{

DataTable dataTable = new DataTable();

dataTable.Load(dr);

dt = dataTable;

sqlConnection.Close();

DataAccessLayer.Entities.User user = new DataAccessLayer.Entities.User();

foreach (DataRow row in dataTable.Rows)

{

user = new DataAccessLayer.Entities.User()

{

Id = row["Id"].ToString()!,

Email = row["Email"].ToString()!,

Password = row["Password"].ToString()!,

FirstName = row["FirstName"].ToString()!,

LastName = row["LastName"].ToString()!,

Type = (bool)row["Type"]

};

MessageBox.Show(user.FirstName.ToString());

}

dataGridView1.AutoGenerateColumns = false;

dataGridView1.DataSource = dataTable;

dataGridView2.DataSource = dataTable;

}

}

catch (SqlException e)

{

sqlConnection.Close();

MessageBox.Show(e.Message);

}

}

}

private void getData\_Load(object sender, EventArgs e)

{

}

private void dataGridView1\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

MessageBox.Show(dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString());

MessageBox.Show(dataGridView1.Rows[e.RowIndex].Cells.Count.ToString());

}

}

}

# Action Btns

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AppLayer.SpecialComponents;

namespace AppLayer.Components.Single

{

//point to function

public delegate void sqlAction();

public partial class btnDelete : UserControl

{

//public sqlAction deleteItem;

public btnDelete()

{

InitializeComponent();

try

{

HelpFunc.addImgCursor("Delete.png", new Size(50, 50), pictureBox1);

}

catch

{

return;

}

}

public void addEventTopictureBox1Click(object key, string nameTableOfItem)

{

pictureBox1.Click += new EventHandler((sender,e)=>deleteItem(key, nameTableOfItem));

}

public void deleteItem(object key, string nameTableOfItem)

{

if (MessageBox.Show("Are you sure you want to delete this information?", "ConfirmationSoniccccc", MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) == System.Windows.Forms.DialogResult.Yes)

{

switch (nameTableOfItem)

{

case "Book":

if (key != null)

{

BookFunc.deleteSelectedBook(key as string);

}

break;

case "Borrow":

if (key != null)

{

BorrowFunc.deleteSelectedBorrow(key as string);

}

break;

case "ExistingCategories":

if (key != null)

{

ExistingCategoriesFun.deleteSelectedExistingCategory(key as DataAccessLayer.Entities.ExistingCategory);

}

break;

case "User":

if (key != null)

{

UserFunc.deleteSelectedUser(key as string);

}

break;

}

}

else

{

MessageBox.Show("Yaaay we stay ! ");

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AppLayer.Components.Single.Show;

using AppLayer.SpecialComponents;

namespace AppLayer.Components.Single

{

public partial class btnShow : UserControl

{

public btnShow()

{

InitializeComponent();

try

{

HelpFunc.addImgCursor("Move.png", new Size(50, 50), pictureBox1);

}

catch

{

return;

}

}

public void addEventTopictureBox1Click(object objShow,string nameTableOfItem)

{

pictureBox1.Click += new EventHandler((sender, e) => seeItem(objShow,nameTableOfItem));

}

public void seeItem(object objShow, string nameTableOfItem)

{

switch (nameTableOfItem)

{

case "Book":

if(objShow != null)

{

showFormBook showBook = new showFormBook(objShow as DataAccessLayer.Entities.Book);

openPopForm(showBook);

}

else

{

MessageBox.Show("no null !");

}

break;

case "Borrow":

if (objShow != null)

{

showFormBorrow showBorrow = new showFormBorrow(objShow as DataAccessLayer.Entities.Borrow);

openPopForm(showBorrow);

}

else

{

MessageBox.Show("no null !");

}

break;

case "ExistingCategories":

if (objShow != null)

{

showFormExistingCategory showExistingCategory = new showFormExistingCategory(objShow as DataAccessLayer.Entities.ExistingCategory);

openPopForm(showExistingCategory);

}

else

{

MessageBox.Show("no null !");

}

break;

case "User":

if (objShow != null)

{

showFormUser showUser = new showFormUser(objShow as DataAccessLayer.Entities.User);

openPopForm(showUser);

}

else

{

MessageBox.Show("no null !");

}

break;

}

}

private void openPopForm(Form popForm)

{

popForm.Activate();

popForm.Show();

//popForm.Location = new Point((this.Width - popForm.Width) / 2, (this.Height - popForm.Height) / 2);

// button in table in user in form

Form mainForm = this.Parent.Parent.Parent as Form;

// we want center so we divide width by 2

int XplusWidth = mainForm!.Location.X + mainForm!.Width / 2;

int resXWithPopWidth = XplusWidth - popForm.Width / 2;

// we want center so we divide width by 2

int YplusHeight = mainForm.Location.Y + mainForm.Height / 2;

int resYWithPopHeight = YplusHeight - popForm.Height / 2;

//popForm.Location = new Point((this.Parent.Parent.Parent.Width - popForm.Width) + popForm.Location.X, resHeightWithLocation);

//popForm.StartPosition = mainForm.CenterScreen;

popForm.Location = new Point(resXWithPopWidth,resYWithPopHeight);

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AppLayer.SpecialComponents;

using AppLayer.Components.comBook;

using AppLayer.Components.comBorrow;

using AppLayer.Components.comExistingCategories;

using AppLayer.Components.comUser ;

namespace AppLayer.Components.Single.btnAction

{

public partial class btnUpDate : UserControl

{

public btnUpDate()

{

InitializeComponent();

try

{

HelpFunc.addImgCursor("Click.png", new Size(50, 50), pictureBox1);

}

catch

{

return;

}

}

public void addEventTopictureBox1Click(object objUpdate, string nameTableOfItem)

{

pictureBox1.Click += new EventHandler((sender, e) => updateItem(objUpdate, nameTableOfItem));

}

public void updateItem(object objUpdate, string nameTableOfItem)

{

switch (nameTableOfItem)

{

case "Book":

if (objUpdate != null)

{

//UpdateBook upBook = new UpdateBook(objShow as DataAccessLayer.Entities.Book);

updateFromTable up1 = new updateFromTable(objUpdate, "Book");

up1.Show();

//showUpUc(upBook);

}

else

{

MessageBox.Show("no null !");

}

break;

//case "Borrow":

// if (objShow != null)

// {

// showBorrow showBorrow = new showBorrow(objShow as DataAccessLayer.Entities.Borrow);

// openPopForm(showBorrow);

// }

// else

// {

// MessageBox.Show("no null !");

// }

// break;

//case "ExistingCategories":

// if (objShow != null)

// {

// showExistingCategory showExistingCategory = new showExistingCategory(objShow as DataAccessLayer.Entities.ExistingCategory);

// openPopForm(showExistingCategory);

// }

// else

// {

// MessageBox.Show("no null !");

// }

// break;

//case "User":

// if (objShow != null)

// {

// showUser showUser = new showUser(objShow as DataAccessLayer.Entities.User);

// openPopForm(showUser);

// }

// else

// {

// MessageBox.Show("no null !");

// }

// break;

}

}

//public void showUpUc(UserControl uc)

//{

// uc.Show();

// this.Parent.Parent.Parent.Controls.Add(uc);

//}

}

}

# Com

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AppLayer.SpecialComponents;

using BusinessLogicLayer.StoredProceduresLogic;

using AppLayer.SpecialComponents;

using DataAccessLayer.Entities;

namespace AppLayer.Components.comBook

{

public partial class AreaBook : Form

{

string kindActionNow;

Book book;

public string KindActionNow

{

get { return kindActionNow; }

set { kindActionNow = value; }

}

public Book Book

{

get { return book; }

set { book = value; }

}

public AreaBook(string kindAction,Book sentBook)

{

InitializeComponent();

Book book = new Book()

{

Code = "a",

FirstName\_Author = "b",

LastName\_Author = "c",

Title = "d",

PublicationDate = DateTime.Now,

Category = "A",

SecondaryCategory = "B"

};

if(sentBook == null)

{

Book = book;

}

kindActionNow = kindAction;

if (Book != null)

{

switch (KindActionNow)

{

case "Add":

addBook2.Book = book;

break;

}

}

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

//this.Close();

}

private void updateBook1\_Load(object sender, EventArgs e)

{

}

private void addBook1\_Load(object sender, EventArgs e)

{

}

private void AreaBook\_Load(object sender, EventArgs e)

{

if (Book != null)

{

switch (KindActionNow)

{

case "Add":

addBook2.Book = book;

break;

}

}

UserControl[] UCsBook = { addBook2, deleteBook1, showAllBooks1, showSearchBook1, updateBook1 };

HelpFunc.hideAndShowUC(UCsBook, KindActionNow, this.MdiParent);

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using BusinessLogicLayer.StoredProceduresLogic;

using AppLayer.SpecialComponents;

using DataAccessLayer.Entities;

namespace AppLayer.Components.comBook

{

public partial class AddBook : UserControl

{

List<string> categories;

List<string> secondaryCategorySelect;

Book book;

public Book Book

{

get { return book; }

set { book = value; }

}

public AddBook()

{

InitializeComponent();

}

private void AddBook\_Load(object sender, EventArgs e)

{

publicationDate.MaxDate = DateTime.Now;

categories = new List<string>();

secondaryCategorySelect = new List<string>();

ExistingCategoriesFun.createCategories(categories, category);

if (Book != null)

{

code.Text = book.Code;

publicationDate.Value = book.PublicationDate;

title.Text = book.Title;

firstNameAuthor.Text = book.FirstName\_Author;

lastNameAuthor.Text = book.LastName\_Author;

category.Text = book.Category;

secondaryCategory.Text = book.SecondaryCategory;

}

}

private void category\_SelectedIndexChanged(object sender, EventArgs e)

{

ExistingCategoriesFun.category\_SelectedIndexChanged(secondaryCategorySelect, secondaryCategory, category.Text);

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AppLayer.SpecialComponents;

namespace AppLayer.Components.comBook

{

public partial class ShowSearchBook : UserControl

{

public ShowSearchBook()

{

InitializeComponent();

// add three check box items to the combo box and set their checked states to true

string[] fieldsName = { "Code", "Title",

"First Name Author", "Last Name Author",

"Publication Date",

"Category", "Secondary Category","All"};

Control[] fieldsControls = { panelCode, panelTitle, panelFirstNameAuthor, panelLastNameAuthor, panelPublicationDate, panelSecondaryCategory, panelCategory };

foreach (Control field in fieldsControls)

{

field.Hide();

field.Tag = "hide";

}

panel1.Hide();

try

{

HelpFunc.createCheckComboBoxList(fieldsName, checkComboBox1, fieldsControls, panel1);

Bitmap bitmap = new Bitmap(new Bitmap("Search.png"), new Size(30, 30));

specialButton1.Cursor = new Cursor(bitmap.GetHicon());

}

catch

{

return;

}

}

private void secondaryCategory\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void ShowBook\_Load(object sender, EventArgs e)

{

}

private void panelTitle\_Paint(object sender, PaintEventArgs e)

{

}

private void panelLastNameAuthor\_Paint(object sender, PaintEventArgs e)

{

}

private void label9\_Click(object sender, EventArgs e)

{

}

private void panel12\_Paint(object sender, PaintEventArgs e)

{

}

private void panelFirstNameAuthor\_Paint(object sender, PaintEventArgs e)

{

}

// this message handler gets called when the user checks/unchecks an item the combo box

//private void checkComboBox1\_CheckStateChanged(object sender, EventArgs e)

//{

// if (sender is CheckComboBoxItem)

// {

// CheckComboBoxItem item = (CheckComboBoxItem)sender;

// MessageBox.Show(item.Text);

// MessageBox.Show(item.CheckState.ToString());

// switch (item.Text)

// {

// case "One":

// //checkBox1.Checked = item.CheckState;

// break;

// case "Two":

// //checkBox2.Checked = item.CheckState;

// break;

// case "Three":

// //checkBox3.Checked = item.CheckState;

// break;

// }

// }

//}

}

}



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AppLayer.SpecialComponents;

namespace AppLayer.Components.comBook

{

public partial class UpdateBook : UserControl

{

DataAccessLayer.Entities.Book book;

public DataAccessLayer.Entities.Book Book

{

get { return book; }

set { book = value; }

}

List<string> categories;

List<string> secondaryCategorySelect;

public UpdateBook()

{

InitializeComponent();

publicationDate.MaxDate = DateTime.Now;

}

public UpdateBook(DataAccessLayer.Entities.Book upBook)

{

InitializeComponent();

publicationDate.MaxDate = DateTime.Now;

Book = upBook;

code.Text = upBook.Code;

title.Text = upBook.Title;

firstNameAuthor.Text = upBook.FirstName\_Author;

lastNameAuthor.Text = upBook.LastName\_Author;

publicationDate.Value = upBook.PublicationDate;

category.Text = upBook.Category;

secondaryCategory.Text = upBook.SecondaryCategory;

}

public void setBook(DataAccessLayer.Entities.Book upBook)

{

book = upBook;

}

private void UpdateBook\_Load(object sender, EventArgs e)

{

categories = new List<string>();

secondaryCategorySelect = new List<string>();

ExistingCategoriesFun.createCategories(categories, category);

if (book != null)

{

code.Text = book.Code;

title.Text = book.Title;

firstNameAuthor.Text = book.FirstName\_Author;

lastNameAuthor.Text = book.LastName\_Author;

publicationDate.MaxDate = book.PublicationDate;

category.Text = book.Category;

secondaryCategory.Text = book.SecondaryCategory;

}

else

{

publicationDate.MaxDate = DateTime.Now;

}

}

private void category\_SelectedIndexChanged\_1(object sender, EventArgs e)

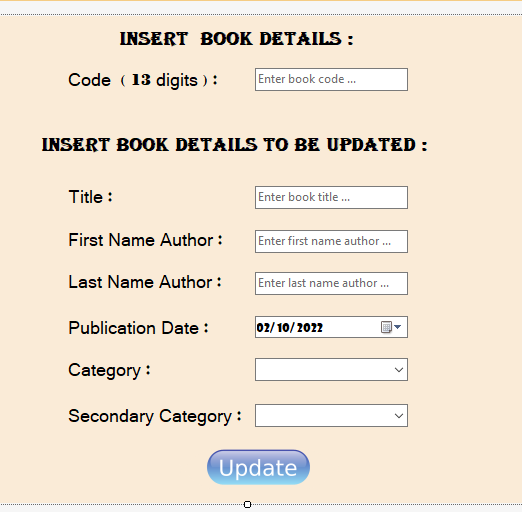
{

ExistingCategoriesFun.category\_SelectedIndexChanged(secondaryCategorySelect, secondaryCategory, category.Text);

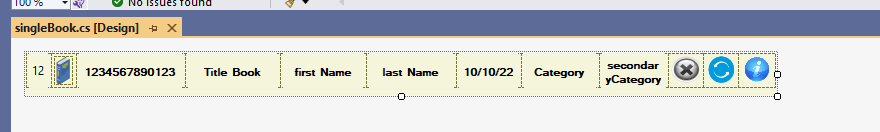
}

}

}



# Single



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using DataAccessLayer.Entities;

namespace AppLayer.Components.Single

{

public partial class singleBook : UserControl

{

#region Set Values to UC

private DataAccessLayer.Entities.Book book;

public DataAccessLayer.Entities.Book Book

{

get { return book; }

set {

book = value;

if(book != null)

{

code.Text = book.Code;

Title.Text = book.Title;

firstName\_Author.Text = book.FirstName\_Author;

lastName\_Author.Text = book.LastName\_Author;

publicationDate.Text = book.PublicationDate.ToString("MM/dd/yy");

category.Text = book.Category;

secondaryCategory.Text = book.SecondaryCategory;

num.Text = CountLine + "";

}

}

}

#endregion

static int countline = 0;

public static int CountLine

{

get { return countline; }

set { countline = value; }

}

public singleBook()

{

InitializeComponent();

}

public singleBook(DataAccessLayer.Entities.Book showBook)

{

InitializeComponent();

countline++;

Book = showBook;

btnDelete1.addEventTopictureBox1Click(showBook.Code,"Book");

btnShow1.addEventTopictureBox1Click(showBook, "Book");

btnUpDate1.addEventTopictureBox1Click(showBook, "Book");

}

private void category\_Click(object sender, EventArgs e)

{

MessageBox.Show("aaaa");

}

int hover;

Color temp;

private void tableLayoutPanel1\_MouseEnter(object sender, EventArgs e)

{

if (hover == 0)

{

temp = BackColor;

BackColor = Color.AliceBlue;

hover++;

}

}

private void singleBook\_MouseLeave(object sender, EventArgs e)

{

if (hover == 1)

{

hover--;

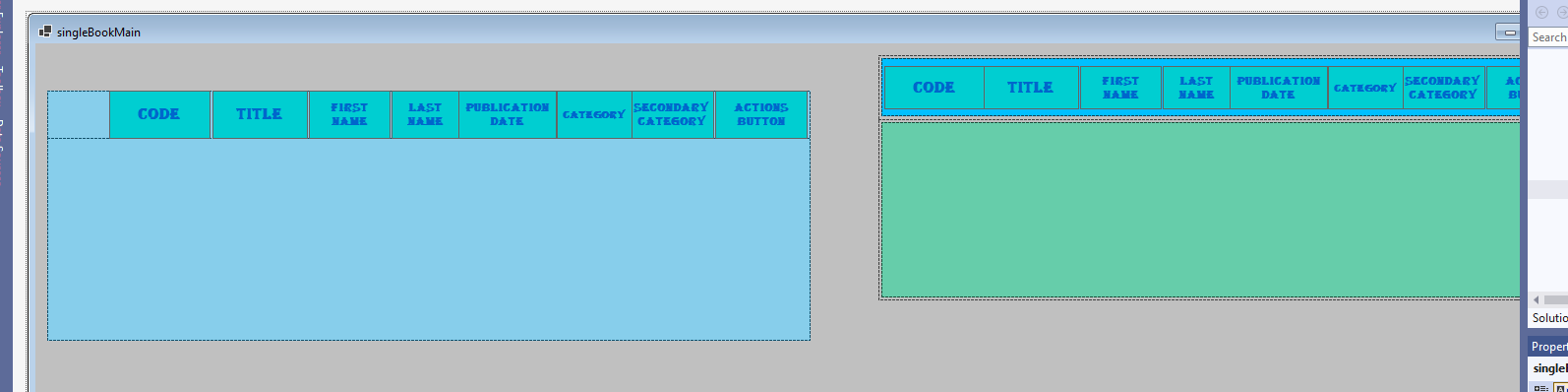
BackColor = temp;

}

}

}

}



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AppLayer.Components.Single

{

public partial class singleBookMain : Form

{

public singleBookMain()

{

InitializeComponent();

DataAccessLayer.Entities.Book book = new DataAccessLayer.Entities.Book()

{

Code = "1",

Category = "2",

FirstName\_Author = "3",

LastName\_Author = "4",

PublicationDate = DateTime.Now,

SecondaryCategory = "5",

Title = "6"

};

//AppLayer.Components.Single.singleBook singleBook = new AppLayer.Components.Single.singleBook(book);

//this.Controls.Add(singleBook);

//singleBook.Show();

//singleBook.Location = new System.Drawing.Point(10, 250);

////singleBook.TabIndex = 7;

//AppLayer.Components.Single.singleBook.CountLine = 0 ;

//AppLayer.Components.Single.singleBook singleBook2 = new AppLayer.Components.Single.singleBook(book);

//this.Controls.Add(singleBook2);

//singleBook2.Show();

//singleBook2.Location = new System.Drawing.Point(10, 400);

//singleBook.Size = new System.Drawing.Size(758, 110);

}

private void singleBookMain\_Load(object sender, EventArgs e)

{

BusinessLogicLayer.StoredProceduresLogic.BookLogic bookLogic = new BusinessLogicLayer.StoredProceduresLogic.BookLogic();

object resFun = bookLogic.getBooks();

if (resFun.GetType() != typeof(DataTable))

{

MessageBox.Show(resFun.ToString());

}

else

{

DataTable dt = (DataTable)resFun;

DataAccessLayer.Entities.Book book;

//dt = dt.AsEnumerable().Reverse().CopyToDataTable();

foreach (DataRow row in dt.Rows)

{

book = new DataAccessLayer.Entities.Book()

{

Code = row["Code"].ToString()!,

Title = row["Title"].ToString()!,

FirstName\_Author = row["FirstName\_Author"].ToString()!,

LastName\_Author = row["LastName\_Author"].ToString()!,

PublicationDate = DateTime.Parse(row["PublicationDate"].ToString()!),

Category = row["Category"].ToString()!,

SecondaryCategory = row["SecondaryCategory"].ToString()!

};

singleBook single = new singleBook(book);

single.Dock = DockStyle.Top;

single.BackColor = getTheme();

panelData.Controls.Add(single);

}

panelData.AutoScrollPosition = new Point(0, 0);

//Main main = new Main(user);

//main.Location = new Point((this.MdiParent.Width) / 2, (this.MdiParent.Height) / 2);

//main.Activate();

//main.Show();

//this.MdiParent.Hide();

}

}

int counter = 0;

public Color getTheme()

{

if(counter %2 == 0)

{

counter++;

return Color.Salmon;

}

else

{

counter++;

return Color.Red;

}

}

private async void LoadBooks()

{

await Task.Run(() =>

{

BusinessLogicLayer.StoredProceduresLogic.BookLogic bookLogic = new BusinessLogicLayer.StoredProceduresLogic.BookLogic();

object resFun = bookLogic.getBooks();

if (resFun.GetType() != typeof(DataTable))

{

MessageBox.Show(resFun.ToString());

}

else

{

DataTable dt = (DataTable)resFun;

DataAccessLayer.Entities.Book book;

//dt = dt.AsEnumerable().Reverse().CopyToDataTable();

foreach (DataRow row in dt.Rows)

{

book = new DataAccessLayer.Entities.Book()

{

Code = row["Code"].ToString()!,

Title = row["Title"].ToString()!,

FirstName\_Author = row["FirstName\_Author"].ToString()!,

LastName\_Author = row["LastName\_Author"].ToString()!,

PublicationDate = DateTime.Parse(row["PublicationDate"].ToString()!),

Category = row["Category"].ToString()!,

SecondaryCategory = row["SecondaryCategory"].ToString()!

};

singleBook single = new singleBook(book);

single.Dock = DockStyle.Top;

single.BackColor = getTheme();

panelData.Controls.Add(single);

}

panelData.AutoScrollPosition = new Point(0, 0);

//Main main = new Main(user);

//main.Location = new Point((this.MdiParent.Width) / 2, (this.MdiParent.Height) / 2);

//main.Activate();

//main.Show();

//this.MdiParent.Hide();

}

});

}

public async Task<List<string>> getStrings()

{

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

await Task.Run(() =>

{

strings.Add("ssss");

});

return strings;

}

public async void getget()

{

List<string> a = await getStrings();

MessageBox.Show(a.Count+"");

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AppLayer.Components.Single.Show

{

public partial class showFormBook : Form

{

public showFormBook(DataAccessLayer.Entities.Book showBook)

{

InitializeComponent();

code.Text = showBook.Code;

firstNameAuthor.Text = showBook.FirstName\_Author;

lastNameAuthor.Text = showBook.LastName\_Author;

title.Text = showBook.Title;

publicationDate.Text = showBook.PublicationDate.ToString("MM/dd/yy");

category.Text = showBook.Category;

secondaryCategory.Text = showBook.SecondaryCategory;

}

}

}