

**I came!**

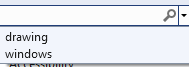
**מדריך C#**

|  |  |
| --- | --- |
| [התחלה](#_התחלה_:) | **3 - 5** |
| [יצירת חיבור למסד נתונים SQL](#_יצירת_חיבור_למסד) | **5 - 28** |
| [פעולות SQL :](#_פעולות_SQL_:) | **29 - 41** |
| [Program Stored + Fun](#_Program_Stored_+) | **41 - 53** |
| [פיתוח App Form](#_פיתוח_App_Form) |  |
|  |  |
|  |  |
|  |  |

# **התחלה :**

## **יצירת אפליקציה דרך פרויקט קונסול :**

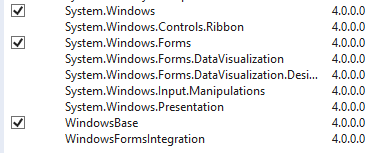
 



**drawing**

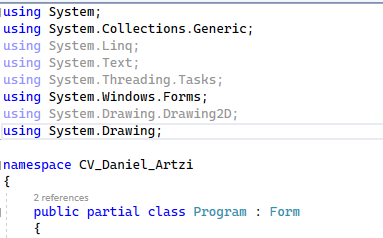


**windows**



**Program**

****

****

using System;

using System.Collections.Generic;

using System.Linq;

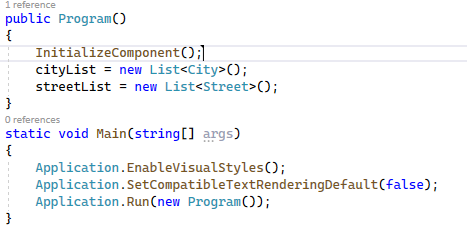
using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing.Drawing2D;

using System.Drawing;

****

public Program()

{

InitializeComponent();

cityList = new List<City>();

streetList = new List<Street>();

}

static void Main(string[] args)

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Program());

}

****

private void InitializeComponent()

{

this.ResumeLayout(false);

this.PerformLayout();

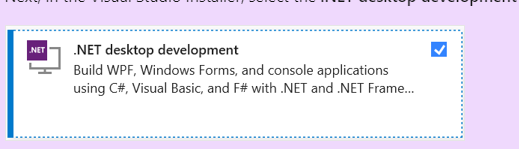
}

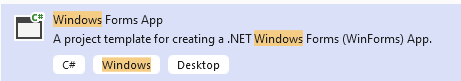
****

**מילה** **מילה** **מילה** **מילה**

## **יצירת אפליקציה דרך אתחול ראשי :**

**תחילה יש לבדוק שהתוסף מותקן**





**windows**

# **יצירת חיבור למסד נתונים SQL :**

## **דרך קצרה :**

 **sqlDbConnect**

public class sqlDbConnect

{

private SqlConnection \_con;

private SqlCommand \_command;

private SqlDataAdapter \_adapter;

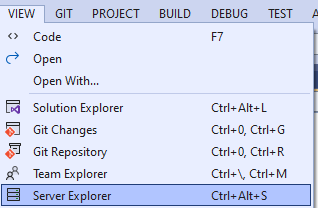
private DataTable \_data;

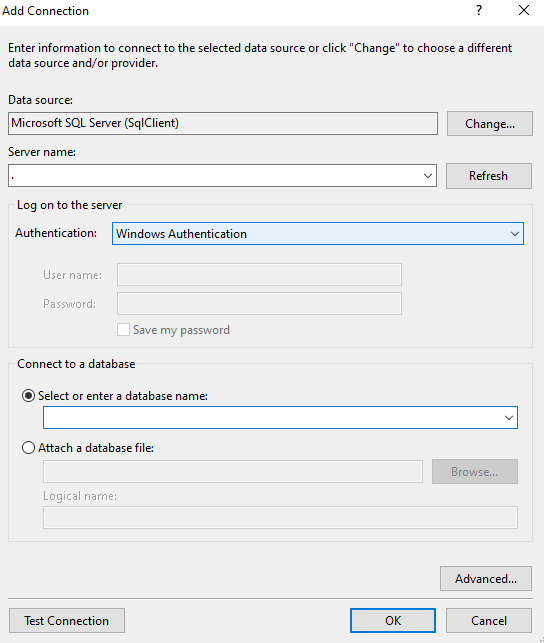
}

using System.Data;

using System.Data.Sql;

using System.Data.SqlClient;

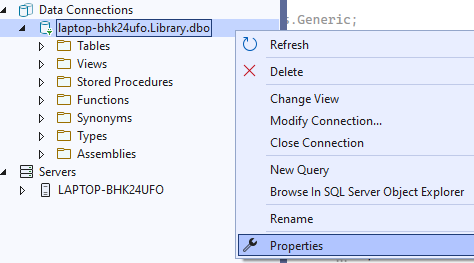




לאחר שהוספנו נחפש אחר **מחרוזת ההתחברות**

נבחר את **שם המסד נתונים** להתחבר אליו

אם **השרת לוקאלי** נוסיף רק ( **.** )



public sqlDbConnect()

{

\_con = new SqlConnection(@"Data Source=.;Initial Catalog=Library;Integrated Security=True");

\_con.Open();

}

נחזור למחלקה ונוסיף את **פונקציית בנאי ההתחברות**

public DataTable QueryEx()

{

\_adapter = new SqlDataAdapter(\_command);

\_data = new DataTable();

\_adapter.Fill(\_data);

return \_data;

}

public void NonQueryEx()

{

if (\_command != null)

{

\_command.ExecuteNonQuery();

}

}

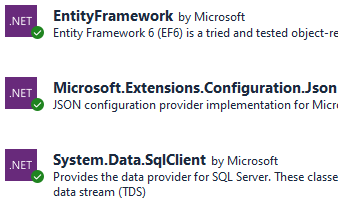
נוסיף פונקציות **להפעלת שאילתות**

## **אם נרצה בדרך יותר מסודרת ומתוחכמת - DataAcessLayer :**





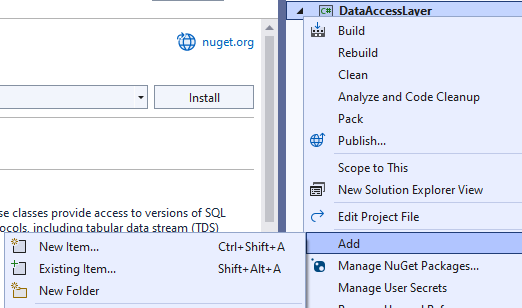
**DataAccessLayer**



microsoft extensions json

data sql data

EntityFramework



**DataContext** - **AppConfiguration.cs**

using Microsoft.Extensions.Configuration;

public class AppConfiguration

{

// We will create a class that receives a connection path to the database dynamically,

//according to the main folder path

public AppConfiguration()

{

// There can be a situation where the library will be used and the path will not be to windows app, i.e. build

try

{

ConfigurationBuilder configBuildr = new ConfigurationBuilder();

//the url end at bin\Debug\net6.0-windows\ -> want parent

DirectoryInfo pathToApp = Directory.GetParent(Directory.GetCurrentDirectory())!.Parent!.Parent!;

string path = Path.Combine(pathToApp!.FullName, "appsettings.json");

configBuildr.AddJsonFile(path, false); // Not Optional Mast Be There

IConfigurationRoot root = configBuildr.Build();

IConfigurationSection appSeting = root.GetSection("ConnectionStrings:DefaultConnestion");

sqlConectionString = appSeting.Value;

}

catch (Exception ex)

{

sqlConectionString = "";

Console.WriteLine(ex.Message);

}

}

public string sqlConectionString { get; set; }

}

public class AppConfiguration // web api

{

// We will create a class that receives a connection path to the database dynamically,

//according to the main folder path

public AppConfiguration()

{

var configBuildr = new ConfigurationBuilder();

var path = Path.Combine(Directory.GetCurrentDirectory(), "appsettings.json");

configBuildr.AddJsonFile(path, false); // Not Optional Mast Be There

var root = configBuildr.Build();

var appSeting = root.GetSection("ConnectionStrings:DefaultConnestion");

sqlConectionString = appSeting.Value;

}

public string sqlConectionString { get; set; }

}

**GeneralSettingsForSQL** - **ParamData.cs** ,

using System.Data;

using System.Runtime.InteropServices;

public struct ParamData

{

public string pName;

public SqlDbType pDataType;

public object? pValue; // can be number , string , date

public ParameterDirection pDirection;

public int? pSize;

public ParamData(string pName, SqlDbType pDataType, object? pValue, ParameterDirection pDirection, [OptionalAttribute] int? size)

{

this.pName = pName;

this.pDataType = pDataType;

this.pValue = pValue;

this.pDirection = pDirection;

this.pSize = size;

}

}

**GeneralSettingsForSQL** - **StoredProcedure.cs** ,

using System.Data;

using System.Runtime.InteropServices;

public class StoredProcedure

{

List<ParamData> sParams;

public string ProcName;

public StoredProcedure()

{

sParams = new List<ParamData>();

ProcName = "";

}

public void SetParam(string pName, SqlDbType pDataType, object? pValue, ParameterDirection pDirection, [OptionalAttribute] int? pSize)

{

ParamData pData = new ParamData(pName, pDataType, pValue, pDirection, pSize);

sParams.Add(pData);

}

//We will add a function to get the

//list of parameters, parameter by name

public List<ParamData>? GetParams()

{

if (sParams.Count != 0)

{

return sParams;

}

else

{

return null;

}

}

public ParamData? GetParamByName(string pNameGet)

{

if (sParams.Count != 0)

{

foreach (ParamData pData in sParams)

{

if (pData.pName == pNameGet)

{

return pData;

}

}

return null;

}

else

{

return null;

}

}

}

**GeneralSettingsForSQL** - **StoredProcedureCollection.cs**

//We will create a class that will be a collection

//of procedures that we will define,

//with add and remove functions as needed

public class StoredProcedureCollection

{

public List<StoredProcedure> listStoredProcedures;

public StoredProcedureCollection()

{

listStoredProcedures = new List<StoredProcedure>();

}

public void add(StoredProcedure value)

{

listStoredProcedures.Add(value);

}

public void remove(int index)

{

if (index > listStoredProcedures.Count - 1 || index < 0)

{

Console.WriteLine("No data to remove");

}

else

{

listStoredProcedures.RemoveAt(index);

}

}

//In addition there will be a function to receive a specific procedure

public StoredProcedure getProcedureByIndex(int Index)

{

//return (StoredProcedure)listStoredProcedures[Index];

return listStoredProcedures[Index];

}

}

**GeneralSettingsForSQL** - **Execute.cs**

using System.Collections;

using System.Data;

using System.Data.SqlClient;

public class Execute

{

// return -> error message / boolean ( true )

public static object ExecuteSps(StoredProcedureCollection spCollection, SqlConnection Connection)

{

try

{

// Go over the procedures to be performed

foreach (StoredProcedure spData in spCollection.listStoredProcedures)

{

SqlCommand cmd = new SqlCommand();

if (Connection.State != ConnectionState.Open)

Connection.Open();

cmd.Connection = Connection;

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = spData.ProcName;

//Go over the parameters of the procedure

IEnumerator myEnumerator = spData.GetParams()!.GetEnumerator();

int i = 0;

while (myEnumerator.MoveNext())

{

ParamData pData = (ParamData)myEnumerator.Current;

cmd.Parameters.Add(pData.pName, pData.pDataType);

cmd.Parameters[i].Value = pData.pValue;

cmd.Parameters[i].Direction = pData.pDirection;

if (pData.pSize.HasValue)

cmd.Parameters[i].Size = (int)pData.pSize;

i++;

}

//Carrying out the procedure and checking

//whether there was an error during the execution

SqlDataReader dr = cmd.ExecuteReader();

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

{

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

// We'll close the connection path so you can read more procedures

Connection.Close();

return message;

}

//Checking if there is data

else if (dr.HasRows)

{

SqlDataReader sqlDataReader = (SqlDataReader)dr;

var dataTable = new DataTable();

dataTable.Load(sqlDataReader);

Connection.Close();

return dataTable;

}

}

//Closing the database connection

Connection.Close();

return true;

}

catch (Exception exc)

{

return exc.Message;

}

}

}

**Entities** – **tables names –**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

public class Book

{

[Key]

[Column(Order = 1)]

[ForeignKey("Code")]

[StringLength(13)]

public string Code { get; set; }

[Key]

[Column(Order = 2)]

[ForeignKey("Id")]

[StringLength(9)]

public string Id { get; set; }

[Required]

[Column(TypeName = "nvarchar(20)")]

[MaxLength(20)]

public string Title { get; set; }

[Required]

[Column(TypeName = "nvarchar(20)")]

public string FirstName\_Author { get; set; }

public ICollection<Borrow> borrows { get; set; }

[Required]

public DateTime PublicationDate { get; set; }

[Required]

[Column(TypeName = "nvarchar(25)")]

public string Category { get; set; }

[Column(TypeName = "nvarchar(35)")]

public string SecondaryCategory { get; set; }

}

**Entities** – **files –** Server\_Error**.cs**

public struct Server\_Error

{

public string? typeRequest { get; set; }

public string? description { get; set; }

public string? errorTime { get; set; }

}

**Interfaces** – **IBook.cs–**

using DataAccessLayer.Entities;

public interface IBook

{

// which functions must be used at sql

public object addNewBook(Book newBook);

public object deleteSelectedBook(string selectedCode);

public object getBooks();

public object ShowFromBook\_BookFromSpecificCode(string code);

public object ShowFromBook\_BooksFromCategory(string category);

public object updateSelectedBook(Book updateBook);

}

**StoredProcedures** – **BookStoredProcedures.cs**

using DataAccessLayer.DataContext;

using DataAccessLayer.Interfaces;

using System.Data.SqlClient;

using DataAccessLayer.Entities;

using System.Data;

using DataAccessLayer.GeneralSettingsForSQL;

public class BookStoredProcedures : IBook

{

//We will connect to the database and run procedures

private AppConfiguration settings { get; set; }

private SqlConnection connection { get; set; }

public BookStoredProcedures()

{

settings = new AppConfiguration();

connection = new SqlConnection();

connection.ConnectionString = settings.sqlConectionString;

}

public object addNewBook(Book newBook)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "addNewBook";

spData.SetParam("@newBook\_Code", SqlDbType.NChar, newBook.Code, ParameterDirection.Input);

spData.SetParam("@newBook\_Title", SqlDbType.NVarChar, newBook.Title, ParameterDirection.Input);

spData.SetParam("@newBook\_FirstName\_Author", SqlDbType.NVarChar, newBook.FirstName\_Author, ParameterDirection.Input);

spData.SetParam("@newBook\_LastName\_Author", SqlDbType.NVarChar, newBook.LastName\_Author, ParameterDirection.Input);

spData.SetParam("@newBook\_PublicationDate", SqlDbType.DateTime, newBook.PublicationDate, ParameterDirection.Input);

spData.SetParam("@newBook\_Category", SqlDbType.NVarChar, newBook.Category, ParameterDirection.Input);

spData.SetParam("@newBook\_SecondaryCategory", SqlDbType.NVarChar, newBook.SecondaryCategory, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object deleteSelectedBook(string selectedCode)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "deleteSelectedBook";

spData.SetParam("@selectedBook\_Code", SqlDbType.NChar, selectedCode, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object getBooks()

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "getBooks";

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object ShowFromBook\_BookFromSpecificCode(string code)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "ShowFromBook\_BookFromSpecificCode";

spData.SetParam("@code", SqlDbType.NChar, code, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object ShowFromBook\_BooksFromCategory(string category)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "ShowFromBook\_BooksFromCategory";

spData.SetParam("@category", SqlDbType.NVarChar, category, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object ShowFromBook\_BooksFromFirstName\_Author(string firstName\_Author)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "ShowFromBook\_BooksFromFirstName\_Author";

spData.SetParam("@firstName\_Author", SqlDbType.NVarChar, firstName\_Author, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object ShowFromBook\_BooksFromLastName\_Author(string lastName\_Author)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "ShowFromBook\_BooksFromLastName\_Author";

spData.SetParam("@lastName\_Author", SqlDbType.NVarChar, lastName\_Author, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object ShowFromBook\_BooksFromName\_Author(string firstName\_Author, string lastName\_Author)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "ShowFromBook\_BooksFromName\_Author";

spData.SetParam("@firstName\_Author", SqlDbType.NVarChar, firstName\_Author, ParameterDirection.Input);

spData.SetParam("@lastName\_Author", SqlDbType.NVarChar, lastName\_Author, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object ShowFromBook\_BooksFromPublicationYear(int publicationYear)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "ShowFromBook\_BooksFromPublicationYear";

spData.SetParam("@publicationYear", SqlDbType.Int, publicationYear, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object ShowFromBook\_BooksFromTitle(string title)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "ShowFromBook\_BooksFromTitle";

spData.SetParam("@title", SqlDbType.NVarChar, title, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

}

public object updateSelectedBook(Book updateBook)

{

if (connection.State != ConnectionState.Open)

connection.Open();

StoredProcedureCollection spCollection = new StoredProcedureCollection();

StoredProcedure spData = new StoredProcedure();

spData.ProcName = "updateSelectedBook";

spData.SetParam("@updateBook\_Code", SqlDbType.NChar, updateBook.Code, ParameterDirection.Input);

spData.SetParam("@updateBook\_Title", SqlDbType.NVarChar, updateBook.Title, ParameterDirection.Input);

spData.SetParam("@updateBook\_FirstName\_Author", SqlDbType.NVarChar, updateBook.FirstName\_Author, ParameterDirection.Input);

spData.SetParam("@updateBook\_LastName\_Author", SqlDbType.NVarChar, updateBook.LastName\_Author, ParameterDirection.Input);

spData.SetParam("@updateBook\_PublicationDate", SqlDbType.DateTime, updateBook.PublicationDate, ParameterDirection.Input);

spData.SetParam("@updateBook\_Category", SqlDbType.NVarChar, updateBook.Category, ParameterDirection.Input);

spData.SetParam("@updateBook\_SecondaryCategory", SqlDbType.NVarChar, updateBook.SecondaryCategory, ParameterDirection.Input);

spData.SetParam("@ERROR", SqlDbType.NVarChar, "", ParameterDirection.InputOutput, 500);

spCollection.add(spData);

return Execute.ExecuteSps(spCollection, connection);

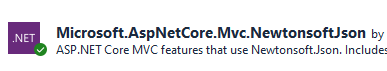
}

}

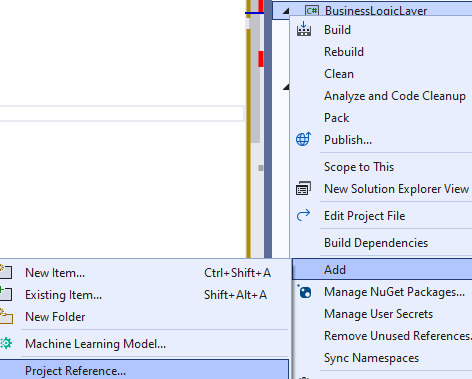
## **אם נרצה בדרך יותר מסודרת ומתוחכמת - BusinessLogicLayer :**



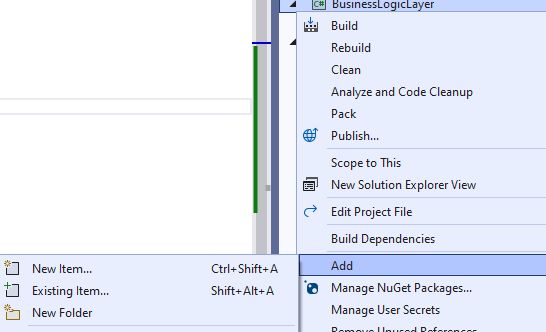
**BusinessLogicLayer**



Microsoft.AspNetCore.Mvc.NewtonsoftJson







**actionFiles – FileError :**

using DataAccessLayer.Entities;

using Newtonsoft.Json;

public class FileError

{

string directoryPath { get; set; }

string filePath { get; set; }

public FileError()

{

//PresentionLayer

DirectoryInfo pathToApp = Directory.GetParent(Directory.GetCurrentDirectory())!.Parent!.Parent!;

directoryPath = Path.Combine(pathToApp.FullName, "Files");

filePath = Path.Combine(directoryPath, "Errors.txt");

}

//// Create textWriter to add and read errors to file

public void addError(string type, string desc)

{

TextWriter txt = new StreamWriter(filePath, append: true);

Server\_Error newServerErrorObj = new Server\_Error()

{

typeRequest = type,

description = desc,

errorTime = DateTime.Now.ToString("dd'-'MM'-'yyyy' 'HH':'mm':'ss")

};

string newServerErrorStr = JsonConvert.SerializeObject(newServerErrorObj);

txt.WriteLine(newServerErrorStr);

txt.Close();

}

public void getErrors()

{

string[] lines = { };

lines = System.IO.File.ReadAllLines(filePath);

List<Server\_Error> errors = new List<Server\_Error>();

foreach (string line in lines)

{

errors.Add(JsonConvert.DeserializeObject<Server\_Error>(line)!);

}

}

}

**Validation– Validation\_General.cs:**

using System.Text.RegularExpressions;

using BusinessLogicLayer.actionFiles;

public static class Validation\_General

{

private static FileError \_fileError = new FileError();

public static string? checkOnlyLetter(string word)

{

if (!Regex.IsMatch(word, @"^[a-zA-Zא-ת]+$"))

{

return "Must write only letters ! -> " + word;

}

else

{

return null;

}

}

public static string insertErr(string err, string newErr, string kindErr)

{

\_fileError.addError(kindErr, newErr);

if (err != "")

err += "\n" + newErr;

else

err = newErr;

return err;

}

}

**Validation** – **Validation\_CheckBook.cs**

using BusinessLogicLayer.actionFiles;

using DataAccessLayer.Entities;

using System.Text.RegularExpressions;

public static class Validation\_CheckBook

{

private static FileError \_fileError = new FileError();

public static string? **checkDate**(DateTime dateBook)

{

if (dateBook > DateTime.Now)

{

return "Date cannot be greater than current date !";

}

else

{

return null;

}

}

public static string? **checkCode**(string codeBook)

{

if (codeBook.Length != 13)

{

return "Barcode must contain 13 digits !";

}

//Note that the IsDigit() method does not strictly check for a character in the range 0 through 9.

//It allows a few characters such as Thai numerals ๐ ๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙.

//We can use the following code to strictly check for ASCII digits:

else if (!codeBook.All(c => (c >= 48 && c <= 57)))

{

return "Barcode must contain only digits ! -> " + codeBook;

}

else

{

return null;

}

}

public static string? **checkEmail**(string email)

{

var trimmedEmail = email.Trim();

// ^ Begin the match at the start of the string.

// [^@\s] + Match one or more occurrences of any character other than the @ character or whitespace.

// @ Match the @ character.

// \. Match a single period character.

// $ End the match at the end of the string.

// a@a.a

string emailReg = @"^[^@\s]+@[^@\s]+\.[^@\s]+$";

if (!Regex.Match(trimmedEmail, emailReg).Success ||

trimmedEmail.EndsWith(".") || trimmedEmail.StartsWith(".") || trimmedEmail.Contains("..") || trimmedEmail.Contains("..") ||

// compare IndexOf to LastIndexOf to check

// if there is more than one @

trimmedEmail.IndexOf("@") != trimmedEmail.LastIndexOf("@"))

{

return "The email is not written correctly !";

}

try

{

var addr = new System.Net.Mail.MailAddress(email);

if (addr.Address != trimmedEmail)

{

return "The email is not written correctly !";

}

}

catch

{

return "The email is not written correctly !";

}

return null;

}

public static string? **checkPassword**(string password)

{

if (password.Length != 10)

{

return "Password must be 10 characters in length !";

}

else

{

string resCheck = "";

if (!password.Any(char.IsUpper))

{

resCheck = "Password must contain an uppercase letter !";

}

if (!password.Any(char.IsLower))

{

if (resCheck != "")

resCheck += "\n" + "Password must contain an uppercase letter !";

}

if (!password.Any(char.IsLower))

{

if (resCheck != "")

resCheck += "\n" + "Password must contain an uppercase letter !";

}

Regex rgx = new Regex("[^A-Za-z0-9]");

if (!rgx.IsMatch(password))

{

if (resCheck != "")

resCheck += "\n" + "Password must contain a special character !";

}

if (resCheck != "")

{ return resCheck; }

else

{ return null; }

}

}

public static string? **checkId**(string id)

{

if (id.Length != 9)

{

return "Id must contain 9 digits ! ";

}

else

{

// The test coefficient is in the form of

// 1 2 1 2 1 2 1 2 1

int[] id\_12\_digits = { 1, 2, 1, 2, 1, 2, 1, 2, 1 };

int count = 0;

// The right digit is the check digit

id = id.PadLeft(9, '0');

for (int i = 0; i < 9; i++)

{

//Multiply a digit by a check factor and add decimal digits

int num = Int32.Parse(id.Substring(i, 1)) \* id\_12\_digits[i];

if (num > 9)

num = (num / 10) + (num % 10);

count += num;

}

//Checking if divisible by 10

if (count % 10 != 0)

{

return "The id format is incorrect ";

}

else

{

return null;

}

}

}

public static string? **checkBook**(Book checkBook)

{

#region Checking the correct input

// Validation for null

string errsValidBook = "";

if (checkBook == null)

{

errsValidBook = Validation\_General.insertErr("", "Values cannot be null", "Client Exception");

return errsValidBook;

}

// Validation for values

else

{

string? checkValid;

checkValid = checkDate(checkBook.PublicationDate);

if (checkValid != null)

{

errsValidBook = Validation\_General.insertErr("", checkValid, "Client Exception");

}

checkValid = checkCode(checkBook.Code);

if (checkValid != null)

{

errsValidBook = Validation\_General.insertErr(errsValidBook, checkValid, "Client Exception");

}

checkValid = Validation\_General.checkOnlyLetter(checkBook.FirstName\_Author);

if (checkValid != null)

{

errsValidBook = Validation\_General.insertErr(errsValidBook, checkValid, "Client Exception");

}

checkValid = Validation\_General.checkOnlyLetter(checkBook.LastName\_Author);

if (checkValid != null)

{

errsValidBook = Validation\_General.insertErr(errsValidBook, checkValid, "Client Exception");

}

checkValid = Validation\_General.checkOnlyLetter(checkBook.Category);

if (checkValid != null)

{

errsValidBook = Validation\_General.insertErr(errsValidBook, checkValid, "Client Exception");

}

if(checkBook.SecondaryCategory != null)

{

checkValid = Validation\_General.checkOnlyLetter(checkBook.SecondaryCategory);

if (checkValid != null)

{

errsValidBook = Validation\_General.insertErr(errsValidBook, checkValid, "Client Exception");

}

}

}

if (errsValidBook != "")

{ return errsValidBook; }

else

{ return null; }

#endregion

}

**StoredProceduresLogic** - **BookLogic.cs**

using DataAccessLayer.Entities;

using DataAccessLayer.Interfaces;

using BusinessLogicLayer.actionFiles;

using BusinessLogicLayer.Validation;

using System.Data;

public class BookLogic

{

// We will implement the functions we defined

// in the "DataAccessLayer" and check the results

private IBook \_IBook = new DataAccessLayer.StoredProcedures.BookStoredProcedures();

private FileError \_fileError = new FileError();

public object **addNewBook**(Book newBook)

{

#region Checking the correct input

string? checkBook = Validation\_CheckBook.checkBook(newBook);

if (checkBook != null)

{

return checkBook;

}

#endregion

#region Query execution

try

{

object result = \_IBook.addNewBook(newBook);

// check if return true / message

if (result.GetType() != typeof(Boolean))

{

return Validation\_General.insertErr("", result.ToString()!, "SQL Exception");

}

else

return "Book successfully added !";

}

catch (Exception ex)

{

return Validation\_General.insertErr("", ex.Message, "Server Exception");

}

#endregion

}

public object **deleteSelectedBook**(string selectedCode)

{

#region Checking the correct input

string? checkValid = Validation\_CheckBook.checkCode(selectedCode);

if (checkValid != null)

{

return Validation\_General.insertErr("", checkValid, "Client Exception");

}

#endregion

#region Query execution

try

{

object result = \_IBook.deleteSelectedBook(selectedCode);

// check if return true / message

if (result.GetType() != typeof(Boolean))

{

return Validation\_General.insertErr("", result.ToString()!, "SQL Exception");

}

else

return "Book successfully deleted !";

}

catch (Exception ex)

{

\_fileError.addError("Server Exception", ex.Message);

return Validation\_General.insertErr("", ex.Message, "Server Exception");

}

#endregion

}

public object **getBooks**()

{

try

{

object result = \_IBook.getBooks()!;

// check if type of DataTable -> data to show

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

{

\_fileError.addError("SQL Exception", result + "");

}

return result;

}

catch (Exception ex)

{

return Validation\_General.insertErr("", ex.Message, "Server Exception");

}

}

public object **ShowFromBook\_BookFromSpecificCode**(string code)

{

#region Checking the correct input

string? checkValid = Validation\_CheckBook.checkCode(code);

if (checkValid != null)

{

return Validation\_General.insertErr("", checkValid, "Client Exception");

}

#endregion

#region Query execution

try

{

object result = \_IBook.ShowFromBook\_BookFromSpecificCode(code);

// check if type of DataTable -> data to show

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

{

\_fileError.addError("SQL Exception", result + "");

}

return result;

}

catch (Exception ex)

{

return Validation\_General.insertErr("", ex.Message, "Server Exception");

}

#endregion

}

public object **updateSelectedBook**(Book updateBook)

{

#region Checking the correct input

string? checkBook = Validation\_CheckBook.checkBook(updateBook);

if (checkBook != null)

{

return checkBook;

}

#endregion

#region Query execution

try

{

object result = \_IBook.updateSelectedBook(updateBook)!;

// check if return true / message

if (result.GetType() != typeof(Boolean))

{

return Validation\_General.insertErr("", result.ToString()!, "SQL Exception");

}

else

return "Book successfully updated !";

}

catch (Exception ex)

{

return Validation\_General.insertErr("", ex.Message, "Server Exception");

}

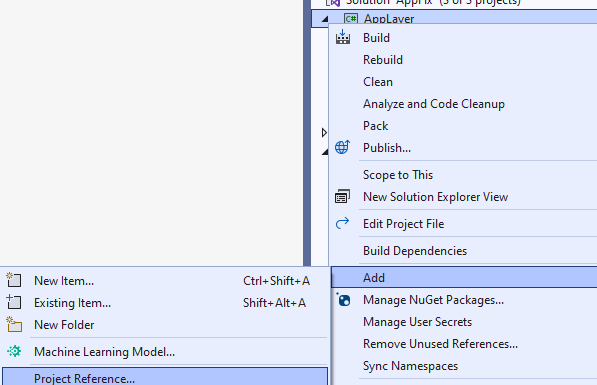
#endregion

}

## **אם נרצה בדרך יותר מסודרת ומתוחכמת - AppLayer :**



**AppLayer / PresentationLayer**

****

****

****

**Models - BookViewModel.cs – not in C# app**

public class BookViewModel

{

public string Code { get; set; }

public string Title { get; set; }

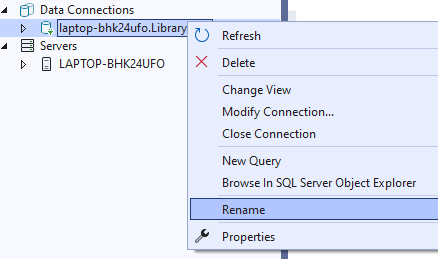
public string FirstName\_Author { get; set; }

public string LastName\_Author { get; set; }

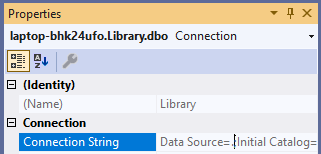
public DateTime PublicationDate { get; set; }

public string Category { get; set; }

public string SecondaryCategory { get; set; } }

****

**appsettings.json**

****

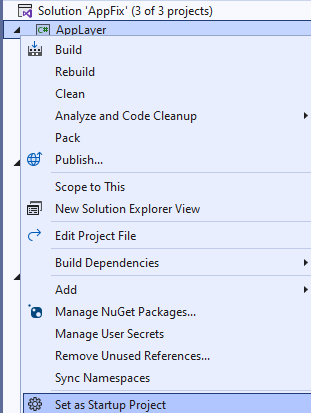
{

"ConnectionStrings": {

"DefaultConnestion": "Data Source=.;Initial Catalog=Library;Integrated Security=True"

}

}

****

**Main Form**

using BusinessLogicLayer.StoredProceduresLogic;

using DataAccessLayer.Entities;

public partial class Form1 : Form

{

// General login

private BookLogic bookLogic;

public Form1()

{

bookLogic = new BookLogic();

InitializeComponent();

}

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

{

Book book = new Book()

{

Code = "1312345678123",

Title = "aa",

Category = "A",

SecondaryCategory = "B",

FirstName\_Author = "A",

LastName\_Author = "B",

PublicationDate = DateTime.Now

};

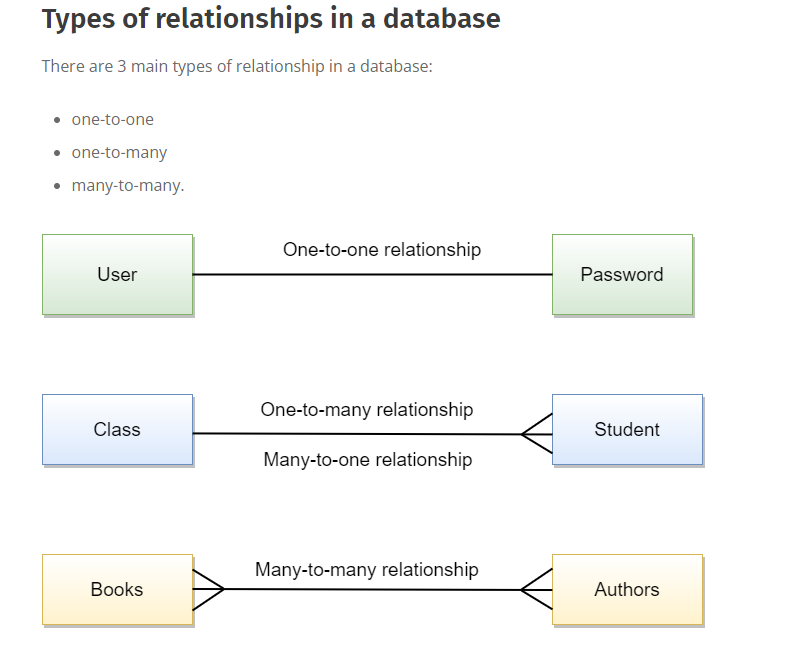
label1.Text = bookLogic.addNewBook(book).ToString();

}

}

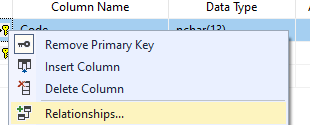
# **פעולות SQL :**

## **Relationships**

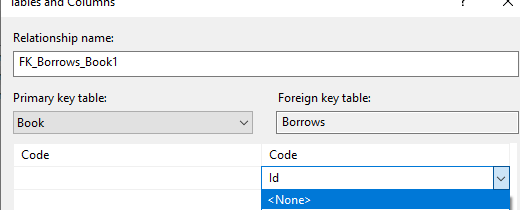
****

## **Add Foreign key**

****

****

****

****

## **Add Index**

## **Index's**

אינדקסים מאפשרים לשאילתות **לרוץ מהר יותר לבחירת נתונים ומהווים חלק מרכזי בביצועים גבוהים** עבור SQL Server.

**אינדקסים מקובצים ממיינים פיזית את דפי הנתונים לפי עמודה או ע**מודות שהם חלק מהאינדקס המקובץ. ניתן **ליצור אינדקס מקובץ על עמודה אחת או עמודות מרובות** ותוכל לקבל רק אינדקס מקבץ אחד בכל טבלה. הסיבה לכך היא שלטבלה יכולה להיות **רק סדר מיון פיזי יחיד** ברמת דף הנתונים המבוסס על האינדקס המקובץ.

דבר נוסף שצריך לשים לב אליו הוא **שאם יש לנו מפתח ראשי בטבלה**, **ברוב** **המקרים זה האינדקס** המקובץ, אבל זה לא נכון בכל המקרים.

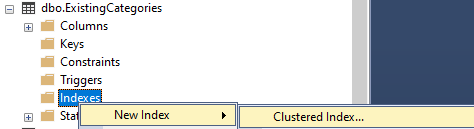
שם האינדקס בתור **CI\_ROOM\_NUM**. מוסכמות השמות שבה השתמשתי היא **CI\_** עבור **Clustered Index** ו-**ROOM\_NUM** עבור שם העמודה. זה יהיה מאוד שימושי כאשר אנו מטפלים באינדקסים רבים כדי שנוכל פשוט להסתכל על השם כדי לקבל מושג כיצד האינדקס נוצר.

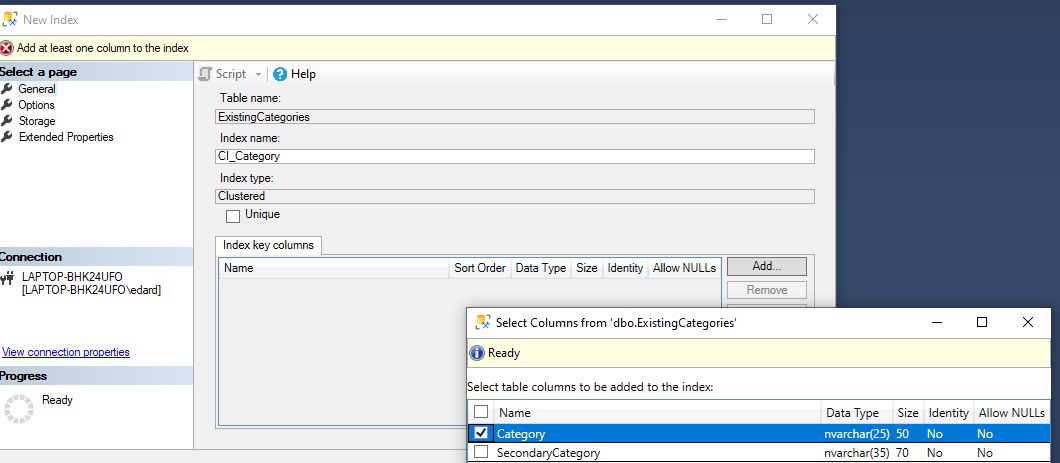
**דפי האינדקס הלא מקובצים ממוינים לפי האינדקס**, **NCI\_ROOM\_NUM** אבל **זה לא משפיע על הנתונים בפועל**. זהו ההבדל העיקרי בהשוואה לאינדקס מקובץ. **אינדקס שאינו מקובץ אינו נוגע בדפי הנתונים** בפועל **למעט מצביעים המצביעים על דפי הנתונים**. לאינדקסים לא מקובצים יש **מבנה נפרד משורות הנתונים.** אינדקס לא מקובץ **מכיל את ערכי מפתח האינדקס הלא מקובצים ולכל הזנת ערך מפתח יש מצביע לשורת הנתונים** המכילה את ערך המפתח.

**אינדקסים מסוננים** הם תכונה שימושית מאוד של SQL Server שיכולה להיות בעלת השפעה **חיובית על ביצועי השאילתות**. על ידי שימוש בקריטריוני סינון, נוכל לוודא שהשאילתה משתמשת **בתת-קבוצה** של הטבלה המלאה בהתבסס על הגדרות הסינון. זה מביא לביצועי אינדקס יעילים **ולחילוץ מהיר יותר של תוצאות**. תכונה זו נותנת לנו יד על העליונה בכל הנוגע **לאופטימיזציה** של **שאילתות**. אינדקסים מסוננים **שימושיים כאשר עלינו לסנן נתוני NULL,** לסנן נתונים **לפי מיקוד מסוים,** לסנן נתונים עם קוד עבודה מסוים, למיין על סמך רבעונים פיננסיים נוכחיים בשנה וכן הלאה. אינדקסים מסננים **עובדים היטב בהשוואה לאינדקס טבלה מלא.**

[**https://www.mssqltips.com/sqlservertip/5573/creating-indexes-with-sql-server-management-studio/**](https://www.mssqltips.com/sqlservertip/5573/creating-indexes-with-sql-server-management-studio/)

* **חשוב לסגור את עיצוב הטבלה שנרצה להוסיף אינדקס**

****

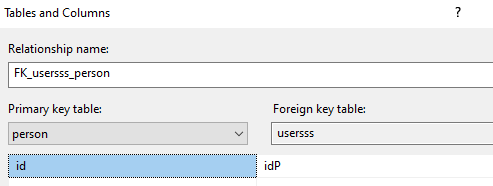
****

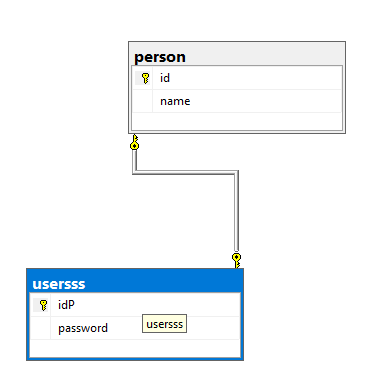
## **one-to-one relationship**

**2 FK as PK**

****

****

****

****

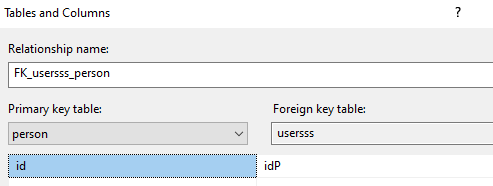
## **one-to-many relationship**

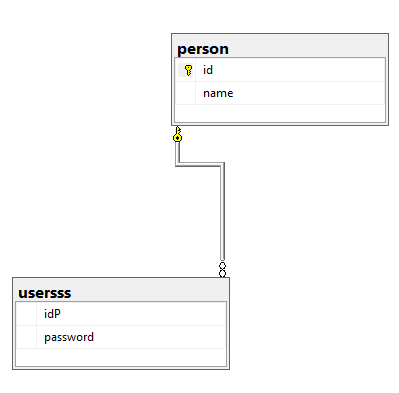
**1 FK as PK**

**ALTER TABLE** A **ADD CONSTRAINT** fk\_b **FOREIGN KEY** (b\_id) **references** b(id)

****



****

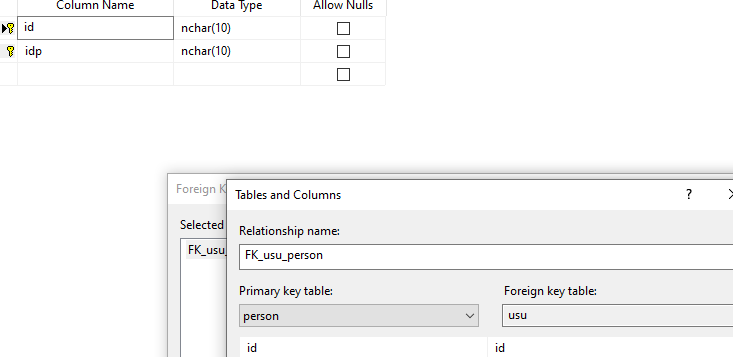
****

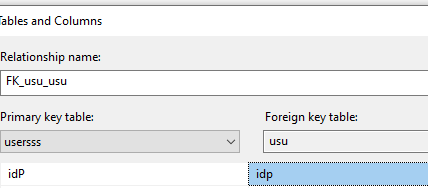
## **many-to-many relationship**

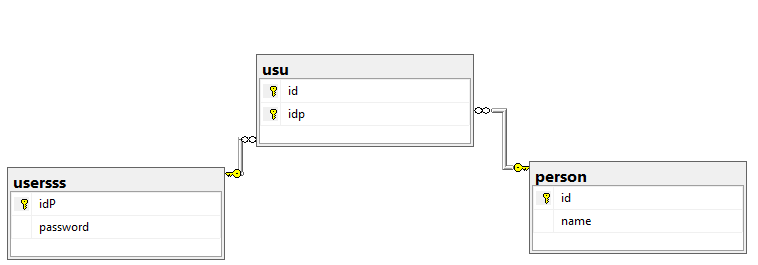
**2 FK as PK to one table**

****

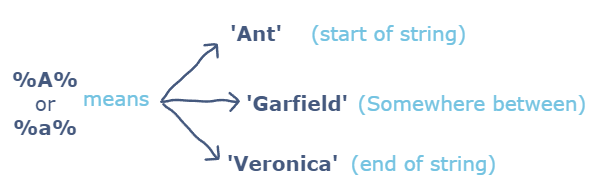
****

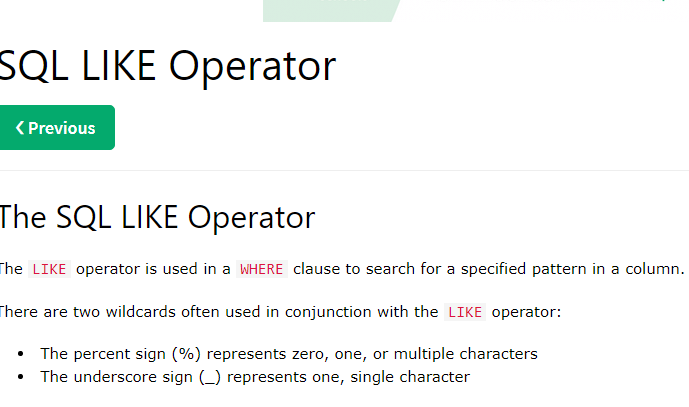
****

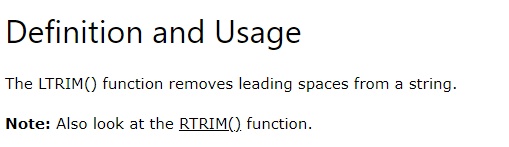
****

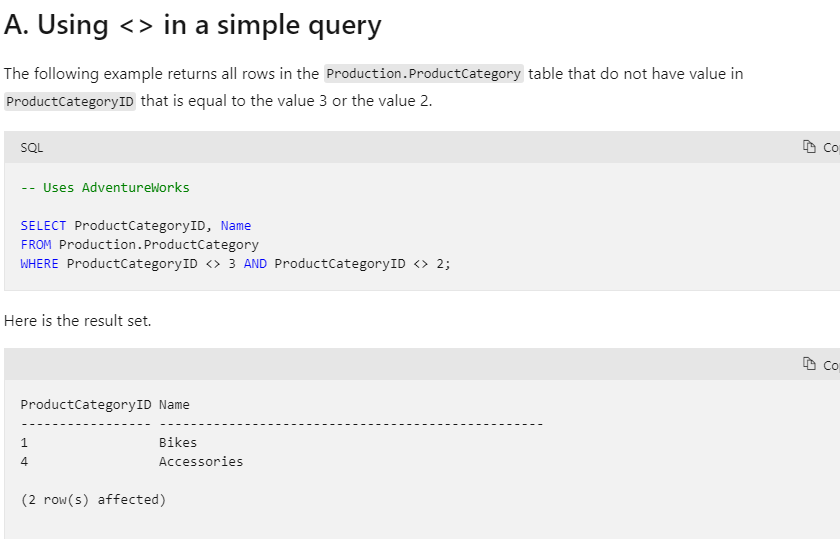
****

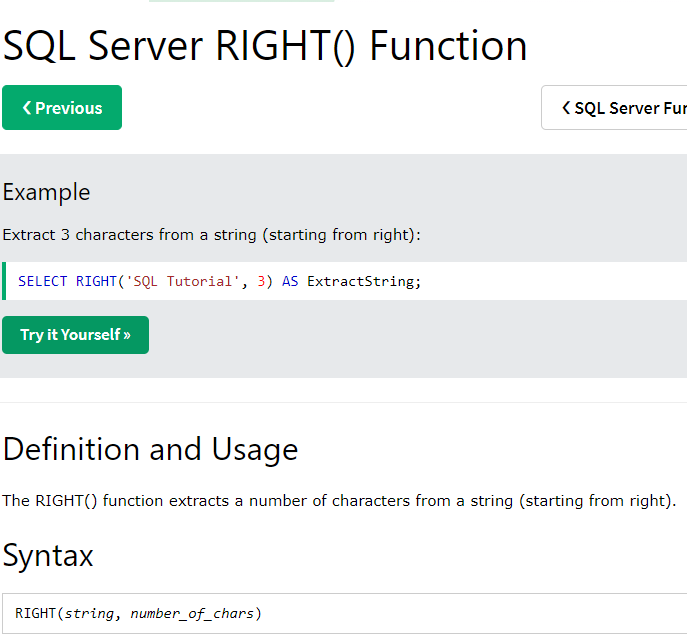
## **String Actions**

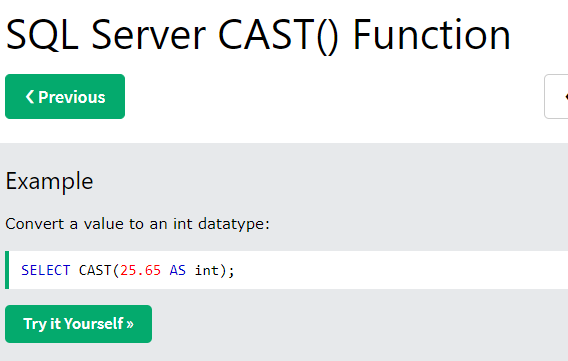
****

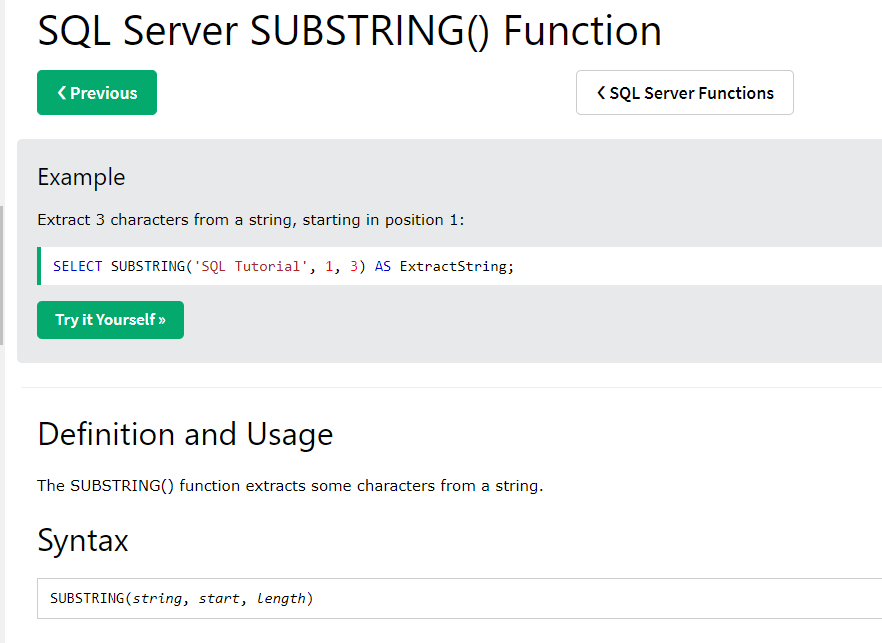
****

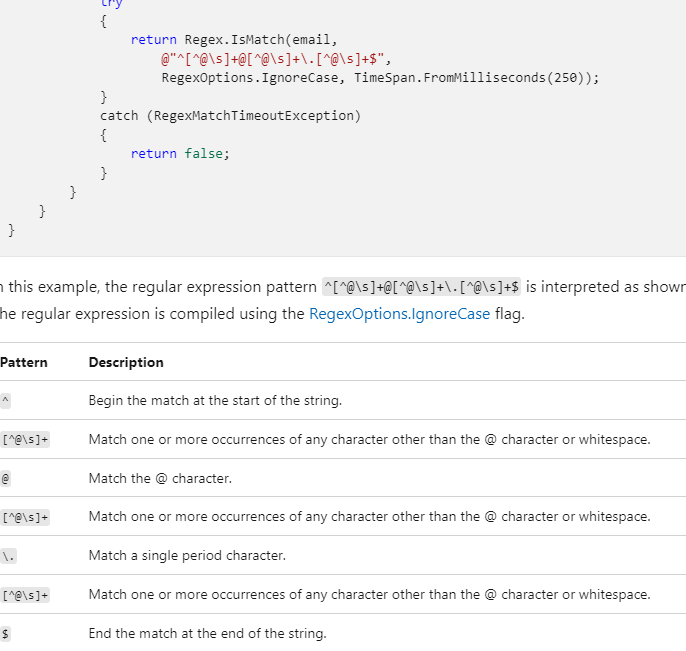
****

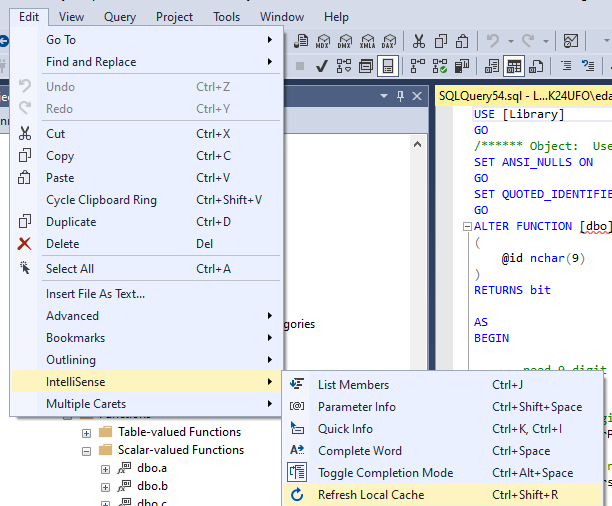
****

****

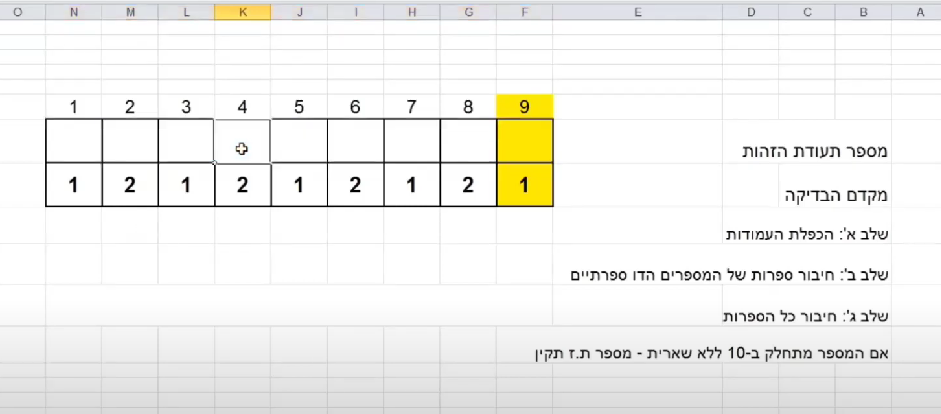
****

****

****

****

## **Function**

****

****

USE [Library]

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[Validation\_CheckIsraelID] Script Date: 30/09/2022 01:55:20 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER FUNCTION [dbo].[Validation\_CheckIsraelID]

(

@id nchar(9)

)

RETURNS bit

AS

BEGIN

-- need 9 digit

if len(@id)<>9 return 0;

--The right digit is the check digit

declare @numberPass TinyInt = Right(@id,1)

-- All but the rightmost digits are the body of the number

declare @numbersID nvarchar(10) = left(@id,8)

declare @numbeCheck TinyInt = 0;

declare @strNum nvarchar(20) = '';

declare @i int = 1;

--Accumulates the digits by multiplying them by weights

WHILE @i <= 8

begin

--The test coefficient is in the form of

--1 2 1 2 1 2 1 2 1

-- SUBSTRING(string, start, length)

-- get the next number

set @strNum += cast(Cast(SUBSTRING(@numbersID,@i,1) As TinyInt) \* (case when @i%2 = 0 then 2 else 1 end) as nvarchar);

set @i+=1;

end

set @i = 1;

--connect the generated digits

WHILE @i <= len(@strNum)

begin

set @numbeCheck += Cast(SUBSTRING(@strNum,@i,1) As TinyInt)

set @i+=1;

end

-- Updates to the number of complements to an exact multiple of ten

set @numbeCheck = (10 - (@numbeCheck%10))

-- Returns a value verified by checking whether the check digit matches

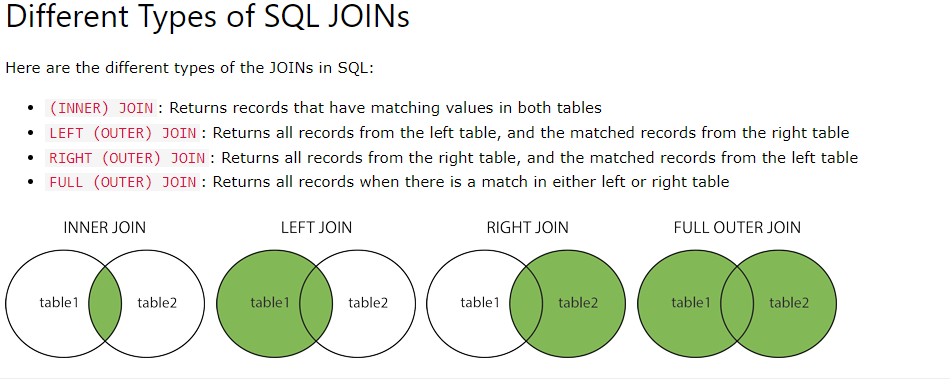
--If the number is divisible by 10 without a remainder, then the id is correct

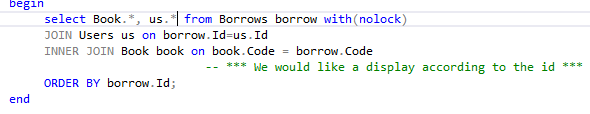
RETURN (case when @numbeCheck=@numberPass then 1 else 0 end)

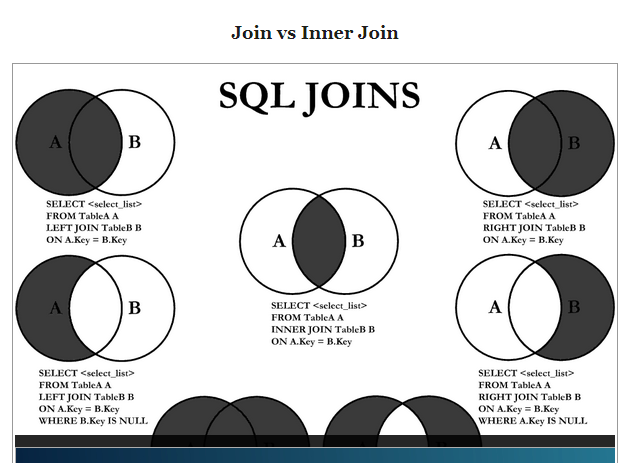
END

****

## **Join**

****

****

****

## **Program Stored + Fun**

### **ADD**

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[addNewBook] Script Date: 01/10/2022 11:53:24 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <07/09/2022>

-- Description: <Add a book to books >

-- =============================================

ALTER **PROCEDURE** [dbo].[addNewBook]

-- parameters for the new book

(@newBook\_Code nchar(13),

@newBook\_Title nvarchar(20),

@newBook\_FirstName\_Author nvarchar(20) ,

@newBook\_LastName\_Author nvarchar(20),

@newBook\_PublicationDate date,

@newBook\_Category nvarchar(25),

@newBook\_SecondaryCategory nvarchar(35) = null,

@ERROR nvarchar(500) OUT )

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

-- \*\*\* Checking if the code is already saved \*\*\*

if EXISTS(select top 1 \* from Book with(nolock) where Code = @newBook\_Code)

begin

SET @ERROR = 'Book with this code already exists !' + CHAR(13)+CHAR(10)

end

else

begin

--~~ We will go into in-depth tests ~~

set @ERROR = [dbo].[Validation\_CheckBook](@newBook\_Code,@newBook\_Title,@newBook\_FirstName\_Author,@newBook\_LastName\_Author,@newBook\_PublicationDate,@newBook\_Category,@newBook\_SecondaryCategory)

if(@ERROR IS NULL or @ERROR = '')

begin

set rowcount 1

INSERT INTO Book(Code, Title, FirstName\_Author, LastName\_Author, PublicationDate, Category, SecondaryCategory)

VALUES (@newBook\_Code, @newBook\_Title, @newBook\_FirstName\_Author, @newBook\_LastName\_Author, @newBook\_PublicationDate, @newBook\_Category, @newBook\_SecondaryCategory);

set rowcount 0

end

end

END

USE [Library]

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[Validation\_CheckBook] Script Date: 01/10/2022 11:55:48 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <07/09/2022>

-- Description: <Validation to Book values >

-- =============================================

ALTER **FUNCTION** [dbo].[Validation\_CheckBook]

(

@Book\_Code nchar(13),

@Book\_Title nvarchar(20),

@Book\_FirstName\_Author nvarchar(20) ,

@Book\_LastName\_Author nvarchar(20),

@Book\_PublicationDate date,

@Book\_Category nvarchar(25),

@Book\_SecondaryCategory nvarchar(35) = null

--There are situations in which we would like to test only on

)

RETURNS nvarchar(500)

AS

BEGIN

declare @Error nvarchar(500)

-- -- \*\*\* Check for values:

-- --- **Date greater than current date**

-- --- 13 digits code

-- --- only digits code

-- --- only letters name

-- --- only letters category

-- --- checking if a category exists \*\*\*

if (@Book\_PublicationDate > GETDATE())

begin

-- \*\*\* **Checking if errors are already written** \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Date cannot be greater than current date !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Date cannot be greater than current date !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* Check for 13 digits code \*\*\*

if (LEN(@Book\_Code) != 13)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Barcode must contain 13 digits !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Barcode must contain 13 digits !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* Check for only digits \*\*\*

if (@Book\_Code LIKE '%[^0-9]%' or @Book\_Code is null)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Barcode must contain only digits !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Barcode must contain only digits !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* Check for only letters \*\*\*

if (( @Book\_FirstName\_Author LIKE '%[^A-Za-zא-ת]%' or @Book\_FirstName\_Author Is Null or @Book\_FirstName\_Author = '' )

or ( @Book\_LastName\_Author LIKE '%[^A-Za-zא-ת]%' or @Book\_LastName\_Author Is Null or @Book\_LastName\_Author = ''))

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Name author must be only letters. !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Name author must be only letters !' + CHAR(13)+CHAR(10)

end

if (( @Book\_Category LIKE '%[^A-Za-zא-ת]%' or @Book\_Category IS NULl or @Book\_Category = '' )

or (@Book\_SecondaryCategory LIKE '%[^A-Za-zא-ת]%' or @Book\_SecondaryCategory = '' ) )

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Category must be only letters. !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Category must be only letters !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* Check if a category exists and enter secondary category \*\*\*

if (@Book\_SecondaryCategory is not null)

begin

if not EXISTS (select top 1 \* from ShowAllCategories with(nolock) where Category = @Book\_Category and SecondaryCategory = @Book\_SecondaryCategory)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

end

end

else

begin

if not EXISTS (select top 1 \* from ShowAllCategories with(nolock) where Category = @Book\_Category)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

end

end

RETURN @ERROR

END

-- -- \*\*\* **Check for only digits** \*\*\*

if (@Book\_Code LIKE '%[^0-9]%' or @Book\_Code is null)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Barcode must contain only digits !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Barcode must contain only digits !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* **Check for only letters** \*\*\*

if (( @Book\_FirstName\_Author LIKE '%[^A-Za-zא-ת]%' or @Book\_FirstName\_Author Is Null or @Book\_FirstName\_Author = '' )

or ( @Book\_LastName\_Author LIKE '%[^A-Za-zא-ת]%' or @Book\_LastName\_Author Is Null or @Book\_LastName\_Author = ''))

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Name author must be only letters. !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Name author must be only letters !' + CHAR(13)+CHAR(10)

end

if (( @Book\_Category LIKE '%[^A-Za-zא-ת]%' or @Book\_Category IS NULl or @Book\_Category = '' )

or (@Book\_SecondaryCategory LIKE '%[^A-Za-zא-ת]%' or @Book\_SecondaryCategory = '' ) )

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Category must be only letters. !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Category must be only letters !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* **Check if a category exists and enter secondary** **category** \*\*\*

if (@Book\_SecondaryCategory is not null)

begin

if not EXISTS (select top 1 \* from ShowAllCategories with(nolock) where Category = @Book\_Category and SecondaryCategory = @Book\_SecondaryCategory)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

end

end

else

begin

if not EXISTS (select top 1 \* from ShowAllCategories with(nolock) where Category = @Book\_Category)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

end

end

RETURN @ERROR

END

else

begin

if **not EXISTS** (select top 1 \* from ShowAllCategories with(nolock) where Category = @Book\_Category)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The category or secondary - category does not exist !' + CHAR(13)+CHAR(10)

end

end

RETURN @ERROR

END

USE [Library]

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[Validation\_CheckUser] Script Date: 01/10/2022 12:00:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <07/09/2022>

-- Description: <Validation to User values >

-- =============================================

ALTER FUNCTION [dbo].[Validation\_CheckUser]

(

@User\_id nchar(9),

@User\_FirstName nvarchar(20),

@User\_LastName nvarchar(20) ,

@User\_Type bit,

@User\_Email nvarchar(20),

@User\_Password nchar(10)

)

RETURNS nvarchar(500)

AS

BEGIN

declare @Error nvarchar(500)

-- -- \*\*\* Check for values:

-- --- **id format ( 9 digit** )

-- --- **type value**

**-- --- only letters name**

-- --- Email is written correctly

-- --- A password must be 10 characters , contains a number, an uppercase letter, a lowercase letter, and a special character \*\*\*

declare @resCheckId bit;

set @resCheckId = [dbo].[Validation\_CheckIsraelID](@User\_id)

if(@resCheckId = 0 )

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Incorrect ID ! ' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Incorrect ID !' + CHAR(13)+CHAR(10)

end

if(@User\_Type > 1 or @User\_Type < 0)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The value of type must be either 0 or 1 ... !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The value of type must be either 0 or 1 ... !' + CHAR(13)+CHAR(10)

end

if (( @User\_FirstName LIKE '%[^A-Za-zא-ת]%' or @User\_FirstName IS NULL or @User\_FirstName = '' )

or ( @User\_LastName LIKE '%[^A-Za-zא-ת]%' or @User\_LastName IS NULL or @User\_LastName = '' ))

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Name user must be only letters. !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Name user must be only letters !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* Checking if the email is written correctly \*\*\*

if ( @User\_Email like '%[^a-z,0-9,@,.,!,#,$,%%,&,'',\*,+,--,/,=,?,^,\_,`,{,|,},~]%' --First Carat ^ means Not these characters in the LIKE clause. The list is the valid email characters.

--an email format \_@\_\_.\_\_

or @User\_Email not like '%\_@\_%\_.[a-z0-9][a-z]%'

--an email does not start / end at .

--an email does not contain a sequence of @ / .

Or @User\_Email like '%@%@%'

Or @User\_Email like '%..%'

Or @User\_Email like '.%'

Or @User\_Email like '%.'

)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The email is not written correctly !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The email is not written correctly !' + CHAR(13)+CHAR(10)

end

-- --- Check - A password must be 10 characters , contains a number, an uppercase letter, a lowercase letter, and a special character \*\*\*

if(LEN(@User\_Password) <> 10 )

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must be 10 characters in length !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must be 10 characters in length !' + CHAR(13)+CHAR(10)

end

--We will use a function PATINDEX

-- to check if there are values (the index is returned if there is)

-- COLLATE Latin1\_General\_100\_BIN2 : binary collation (Latin1\_General\_100\_BIN2).

--binary collations sort each case separately (like this: AB....YZ...ab...yz).

--Other collations tend to intermingle the uppercase and lowercase letters (like this: AaBb...YyZz),

--which would therefore match both uppercase and lowercase characters.

if (PATINDEX('%[A-Z]%',@User\_Password COLLATE Latin1\_General\_100\_BIN2) = 0)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain an uppercase letter !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain an uppercase letter !' + CHAR(13)+CHAR(10)

end

if(PATINDEX('%[a-z]%',@User\_Password COLLATE Latin1\_General\_100\_BIN2) = 0)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain a lowercase letter !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain a lowercase letter !' + CHAR(13)+CHAR(10)

end

if(@User\_Password not like '%[-+\_!@#$%^&\*.,?~^(){}=]%')

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain a special character !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain a special character !' + CHAR(13)+CHAR(10)

end

RETURN @ERROR

END

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Name user must be only letters. !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Name user must be only letters !' + CHAR(13)+CHAR(10)

end

-- -- \*\*\* **Checking if the email is written correctly** \*\*\*

if ( @User\_Email like '%[^a-z,0-9,@,.,!,#,$,%%,&,'',\*,+,--,/,=,?,^,\_,`,{,|,},~]%' --First Carat ^ means Not these characters in the LIKE clause. The list is the valid email characters.

--an email format \_@\_\_.\_\_

or @User\_Email not like '%\_@\_%\_.[a-z0-9][a-z]%'

--an email does not start / end at .

--an email does not contain a sequence of @ / .

Or @User\_Email like '%@%@%'

Or @User\_Email like '%..%'

Or @User\_Email like '.%'

Or @User\_Email like '%.'

)

begin

-- \*\*\* **Checking if errors are already written** \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'The email is not written correctly !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'The email is not written correctly !' + CHAR(13)+CHAR(10)

end

-- --- **Check - A password must be 10 characters , contains a number, an uppercase letter, a lowercase letter, and a special character \*\*\***

if(LEN(@User\_Password) <> 10 )

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must be 10 characters in length !' + CHAR(13)+CHAR(10)

else

SET @ERROR +=

'Password must be 10 characters in length !' + CHAR(13)+CHAR(10)

end

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain an uppercase letter !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain an uppercase letter !' + CHAR(13)+CHAR(10)

end

if(PATINDEX('%[a-z]%',@User\_Password COLLATE Latin1\_General\_100\_BIN2) = 0)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain a lowercase letter !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain a lowercase letter !' + CHAR(13)+CHAR(10)

end

if(@User\_Password not like '%[-+\_!@#$%^&\*.,?~^(){}=]%')

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain a special character !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain a special character !' + CHAR(13)+CHAR(10)

end

RETURN @ERROR

END

--We will use a function PATINDEX

-- to check if there are values (the index is returned if there is)

-- COLLATE Latin1\_General\_100\_BIN2 : binary collation (Latin1\_General\_100\_BIN2).

--binary collations sort each case separately (like this: AB....YZ...ab...yz).

--Other collations tend to intermingle the uppercase and lowercase letters (like this: AaBb...YyZz),

--which would therefore match both uppercase and lowercase characters.

if (PATINDEX('%[A-Z]%',@User\_Password COLLATE Latin1\_General\_100\_BIN2) = 0)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain an uppercase letter !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain an uppercase letter !' + CHAR(13)+CHAR(10)

end

if(PATINDEX('%[a-z]%',@User\_Password COLLATE Latin1\_General\_100\_BIN2) = 0)

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain a lowercase letter !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain a lowercase letter !' + CHAR(13)+CHAR(10)

end

if(@User\_Password not like '%[-+\_!@#$%^&\*.,?~^(){}=]%')

begin

-- \*\*\* Checking if errors are already written \*\*\*

if (@ERROR IS NULL or @ERROR = '')

-- linebreaks ->

SET @ERROR = 'Password must contain a special character !' + CHAR(13)+CHAR(10)

else

SET @ERROR += 'Password must contain a special character !' + CHAR(13)+CHAR(10)

end

RETURN @ERROR

END

### **Delete**

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[deleteSelectedBook] Script Date: 01/10/2022 12:05:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <07/09/2022>

-- Description: <Delete a book from Books >

-- =============================================

ALTER PROCEDURE [dbo].[deleteSelectedBook]

(@selectedBook\_Code nchar (13),

@ERROR nvarchar(500) OUT )

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

-- \*\*\* Checking if books exist \*\*\*

if NOT EXISTS (select top 1 \* from book with(nolock) where Code = @selectedBook\_Code )

begin

SET @ERROR = 'Sorry, no book with this code was found :< !'

end

-- \*\*\* Checking if all the values are correct and no error message was generated \*\*\*

if(@ERROR IS NULL or @ERROR = '')

begin

set rowcount 1

DELETE From Book Where Code = @selectedBook\_Code

set rowcount 0

end

END

### **GET**

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[getBooks] Script Date: 01/10/2022 12:06:14 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <07/09/2022>

-- Description: <Get all the books>

-- =============================================

ALTER PROCEDURE [dbo].[getBooks]

@ERROR nvarchar(500) OUT

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

-- \*\*\* Checking if books exist \*\*\*

if NOT EXISTS (select \* from Book with(nolock))

begin

SET @ERROR = 'It is not possible! You didn`t keep a single book @#@'

end

-- \*\*\* Checking if all the values are correct and no error message was generated \*\*\*

if(@ERROR IS NULL or @ERROR = '')

begin

select \* from Book with(nolock)

-- \*\*\* A representative according to the publication date of the books from day to the past \*\*\*

ORDER BY PublicationDate DESC;

end

----can return only integer values -> return the number of books we saved

DECLARE @booksCount int

SELECT @booksCount = count(\*) FROM Book

RETURN @booksCount

END

### **Show**

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ShowFromBook\_BookFromSpecificCode] Script Date: 01/10/2022 12:07:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <10/09/2022>

-- Description: <Get book of specific code>

-- =============================================

ALTER PROCEDURE [dbo].[ShowFromBook\_BookFromSpecificCode]

@code nchar(13),

@ERROR nvarchar(500) OUT

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

-- \*\*\* Checking if code exist \*\*\*

if NOT EXISTS (select \* from Book with(nolock) where Code = @code)

begin

SET @ERROR = 'God of all shifra! code not found @#@'

end

else

begin

select top 1 \* from Book with(nolock)

where Code = @code

end

----can return only integer values -> return the number of books we saved

DECLARE @BooksCount int

SELECT @BooksCount = count(\*) FROM Book

RETURN @BooksCount

END

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[deleteSelectedBook] Script Date: 01/10/2022 12:05:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <07/09/2022>

-- Description: <Delete a book from Books >

-- =============================================

ALTER PROCEDURE [dbo].[deleteSelectedBook]

(@selectedBook\_Code nchar (13),

@ERROR nvarchar(500) OUT )

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

-- \*\*\* Checking if books exist \*\*\*

if NOT EXISTS (select top 1 \* from book with(nolock) where Code = @selectedBook\_Code )

begin

SET @ERROR = 'Sorry, no book with this code was found :< !'

end

-- \*\*\* Checking if all the values are correct and no error message was generated \*\*\*

if(@ERROR IS NULL or @ERROR = '')

begin

set rowcount 1

DELETE From Book Where Code = @selectedBook\_Code

set rowcount 0

end

END

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ShowFromBook\_BooksFromCategory] Script Date: 01/10/2022 12:07:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <10/09/2022>

-- Description: <Get all the books of Category>

-- =============================================

ALTER PROCEDURE [dbo].[ShowFromBook\_BooksFromCategory]

@category nvarchar(25),

@ERROR nvarchar(500) OUT

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

-- \*\*\* Checking if category exist \*\*\*

if NOT EXISTS (select \* from Book with(nolock) where Category like '%'+@Category+'%')

begin

SET @ERROR = 'God of all shifra! category not found @#@'

end

else

begin

select \* from Book with(nolock)

where Category like '%'+@Category+'%'

-- \*\*\* A representative according to the publication date of the books from day to the past \*\*\*

ORDER BY PublicationDate DESC;

end

----can return only integer values -> return the number of books we saved

DECLARE @BooksCount int

SELECT @BooksCount = count(\*) FROM Book

RETURN @BooksCount

END

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ShowFromBook\_BooksFromPublicationYear] Script Date: 01/10/2022 12:08:05 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <10/09/2022>

-- Description: <Get all the books of year>

-- =============================================

ALTER PROCEDURE [dbo].[ShowFromBook\_BooksFromPublicationYear]

@publicationYear int,

@ERROR nvarchar(500) OUT

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

if(@publicationYear < 0 or @publicationYear > 9999)

begin

SET @ERROR = 'Do you want to travel in time? This year makes no sense @#@'

end

-- \*\*\* Checking if year exist \*\*\*

else if NOT EXISTS (select \* from Book with(nolock) where year(PublicationDate) = @publicationYear)

begin

SET @ERROR = 'God of all shifra! publication year not found @#@'

end

else

begin

select \* from Book with(nolock)

where year(PublicationDate) = @publicationYear

-- \*\*\* A representative according to the publication date of the books from day to the past \*\*\*

ORDER BY PublicationDate DESC;

end

----can return only integer values -> return the number of books we saved

DECLARE @BooksCount int

SELECT @BooksCount = count(\*) FROM Book

RETURN @BooksCount

END

### **Update**

USE [Library]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[updateSelectedBook] Script Date: 01/10/2022 12:09:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Daniel Artzi>

-- Create date: <07/09/2022>

-- Description: <Add a book to books >

-- =============================================

ALTER PROCEDURE [dbo].[updateSelectedBook]

-- parameters for the new book

(@updateBook\_Code nchar(13),

@updateBook\_Title nvarchar(20),

@updateBook\_FirstName\_Author nvarchar(20) ,

@updateBook\_LastName\_Author nvarchar(20),

@updateBook\_PublicationDate date,

@updateBook\_Category nvarchar(25),

@updateBook\_SecondaryCategory nvarchar(35) = null,

@ERROR nvarchar(500) OUT

)

AS

BEGIN

--message that indicates the number of rows

--that are affected by the T-SQL statement

--is not returned as part of the results.

SET NOCOUNT ON;

-- \*\*\* Checking if the code does not exist \*\*\*

if Not EXISTS(select top 1 \* from Book with(nolock) where Code = @updateBook\_Code)

begin

SET @ERROR = 'Sorry, no book with this code was found :<' + CHAR(13)+CHAR(10)

end

else

begin

--~~ We will go into in-depth tests ~~

set @ERROR = [dbo].[Validation\_CheckBook](@updateBook\_Code,@updateBook\_Title,@updateBook\_FirstName\_Author,@updateBook\_LastName\_Author,@updateBook\_PublicationDate,@updateBook\_Category,@updateBook\_SecondaryCategory)

if(@ERROR IS NULL or @ERROR = '')

begin

set rowcount 1

UPDATE Book

Set Title = @updateBook\_Title,

FirstName\_Author = @updateBook\_FirstName\_Author,

LastName\_Author = @updateBook\_LastName\_Author,

PublicationDate = @updateBook\_PublicationDate,

Category = @updateBook\_Category,

SecondaryCategory = @updateBook\_SecondaryCategory

Where Code = @updateBook\_Code

set rowcount 0

end

end

END

# **פיתוח App Form :**

/// <summary>

/// C'tor - creates a CheckComboBoxItem

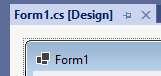
/// </summary>

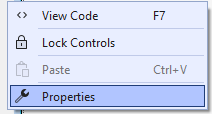
/// <param name="text">Label of the check box in the drop down list</param>

/// <param name="initialCheckState">Initial value for the check box (true=checked)</param>

/// <returns></returns>

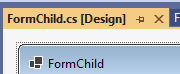
## **MDI -**  [**Multiple-Document Interface**](https://learn.microsoft.com/en-us/dotnet/desktop/winforms/advanced/multiple-document-interface-mdi-applications?view=netframeworkdesktop-4.8)

****

****

****

****

****

FormChild formChild = new FormChild();

formChild.MdiParent = this;

formChild.Show();

formChild.Activate();

//center child in parent

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

private void RemoveFormsAndShow(string AddOrShowForm, string kindAddOrShow)

{

// We will go through the open forms and close any form that is not the main form

// We can't do this in a loop for each

//Since the transition will be destroyed at the first closing of Open Form

//Go through the actual amount of Open Forms

FormCollection FormsOpen = Application.OpenForms;

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

{

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

FormsOpen[i].Close();

}

// We will check what type of form we would like to show / add

// And then what kind of add / show -> city / street

switch (AddOrShowForm)

{

case "add":

Add add = new Add(kindAddOrShow, this.cityList, this.streetList);

add.MdiParent = this;

add.Show();

add.Activate();

add.Location = new Point((this.Width - add.Width) / 2, 0);

break;

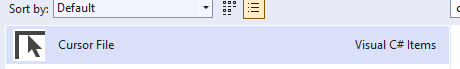
}

}

## **Style**

|  |  |
| --- | --- |
| **סטייל לטקסט כותרת ראשית** | **Showcard Gothic, 18pt**  **Old English Text MT, 18pt**  **Goudy Stout, 18pt , style=Bold**  **Jokerman, 18pt, style=Bold**  **Lucida Calligraphy, 18pt , style=Bold** |
| **סטייל לטקסט כותרת משנית** | **ROG Fonts, 15.749998pt**  **Algerian, 14.25pt, style=Bold** |
| **סטייל לטקסט כותרת פירוט** | **Modern No. 20, 14.25pt**  **Elephant, 14.25pt**  **Guttman Mantova-Decor, 14.249998pt**  **Guttman Drogolin, 14.249998pt**  **Guttman Stam, 14.249998pt**  **Narkisim, 14.25pt, style=Bold** |
| **סטייל לטקסט כותרת תחתונה** | **Castellar, 14.25pt** |
|  |  |
|  |  |
|  |  |
|  |  |

## **Cursor**

**  
**

****

****

****

****

****

Bitmap cm = new Bitmap(new Bitmap("Click.png"), 16, 16);

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

## **Button**

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();

}

}

}

## **Song**

**רק** wav

****

using System.Media;

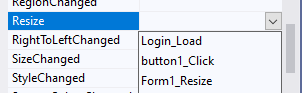
SoundPlayer simpleSound = new SoundPlayer(@"appSong.wav");

simpleSound.Stop();

simpleSound.PlayLooping();

simpleSound.Play();

## **Responsive**

****

private void resizeControl(Rectangle r, Control c)

{

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

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

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);

}

private Rectangle buttonOriginalRectangle;

private Rectangle originalFormSize;

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

{

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

buttonOriginalRectangle = new Rectangle(specialButton1.Location.X, specialButton1.Location.Y, specialButton1.Width, specialButton1.Height);

}

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

{

resizeControl(buttonOriginalRectangle, specialButton1);

}

public partial class Login : Form

{

// save values

private Rectangle[] controlerOriginalRectangle;

// save pointer to controls

private Control[] controls;

private Rectangle originalFormSize;

public Login()

{

InitializeComponent();

}

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

{

SoundPlayer simpleSound = new SoundPlayer(@"appSong.wav");

simpleSound.Stop();

simpleSound.PlayLooping();

//simpleSound.Play();

specialButton1.Cursor = new Cursor("Click.cur");

Bitmap cm = new Bitmap(new Bitmap("Move.png"), 80, 80);

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

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

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

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

// copy all collection to array from 0

this.Controls.CopyTo(controls, 0);

//// 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);

}

List<Control> listControls = controls.ToList();

}

private void button1\_Click(object sender, EventArgs e)

{

}

private void resizeControl(Rectangle r, Control c)

{

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

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

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);

}

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

{

// loop over controls and updates values

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

{

resizeControl(controlerOriginalRectangle[index], control);

}

}

private static void resizeControl(Rectangle r, Control c, Rectangle originalFormSize, Form thisForm)

{

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

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

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,Form thisForm)

{

// loop over controls and updates values

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

{

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

}

}

public static void Form\_Load(ref Rectangle originalFormSize, ref Control[] controls, ref Rectangle[] controlerOriginalRectangle, Form thisForm)

{

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);

//// 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);

}

}

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.Show();

a1.Show();

b1.Hide();

HelpFunc.Form\_Load(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);

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AppLayer.SpecialComponents

{

public class HelpFunc

{

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

{

float xRatio;

float yRatio;

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)

{

// 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 Form\_Load(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 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.Show();

a1.Show();

b1.Hide();

HelpFunc.Form\_Load(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 UCLogin()

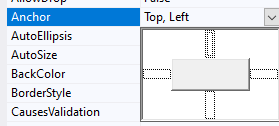
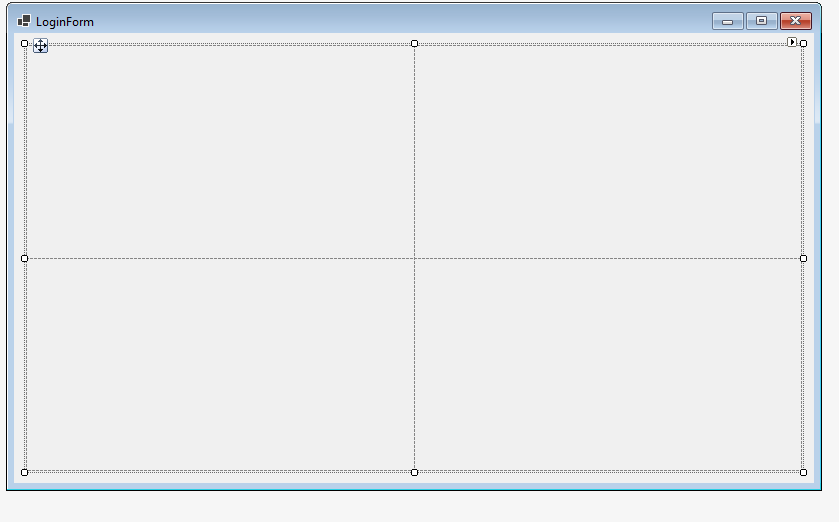
{

InitializeComponent();

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

}

****

****

<https://www.youtube.com/watch?v=B682wOeFunw>

## **Errors**



public void checkAndSetError(Control insertErrNext, string? checkRes)

{

if (checkRes != null)

{

errorPassword.SetError(insertErrNext, checkRes);

}

else

{

errorPassword.SetError(insertErrNext, String.Empty);

}

}

## **Navigation Form**

**Without mdi**

MainApp mainApp = new MainApp();

mainApp.Show();

this.Hide();

**With mdi**

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

{

Login login = new Login();

login.MdiParent = this;

login.Show();

}

MainApp mainApp = new MainApp();

mainApp.TopLevel = false;

mainApp.Parent = this.MdiParent;

mainApp.Activate();

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

mainApp.Show();

this.Close();

**With userControl**

****

a1.Show();

b1.Hide();

this.Hide();

Control c = this.Parent.Controls.Find("b1", false)[0];

c.Show();

**-**

**Hide Border**

****

User controls are a way of **making a custom**, reusable **component**. A user control can contain other **controls but must be hosted by a form.**

Windows forms are the **container** for **controls**, including **user controls**. While it contains **many similar attributes** as a **user** **control**, it's primary purpose is **to** **host** controls.

## **Menu Item**

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

showToolStripMenuItem2,

showToolStripMenuItem3});

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

{

ToolStripMenuItem menuStrip = (ToolStripMenuItem)sender;

ToolStripItem parent = menuStrip.OwnerItem;

MessageBox.Show(parent.Text);

}

## **Combo Box**

type.SelectedIndex = 1;

MessageBox.Show(type.SelectedIndex.ToString());

MessageBox.Show(type.Text);

ComboBoxItem typeItem = (ComboBoxItem)type.SelectedItem;

string value = typeItem.Content.ToString();

publicationDate.MaxDate = DateTime.Now;

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

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

//category.ValueMember = "cityName";

//Cities.DisplayMember = "cityName";

object resFun = existingCategories.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;

//category.ValueMember = "categories";

//category.DisplayMember = "categories";

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

{

secondaryCategory.DataSource = null;

secondaryCategorySelect.Clear();

object resFun = existingCategorylogic.ShowFromExistingCategories\_SubcategoryFromCategory(category.Text);

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;

}

## **UC**

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

{

if(ucs.Length != 4)

{

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

return;

}

foreach (UserControl uc in ucs)

{

uc.Hide();

}

switch (kindAction)

{

case "Add":

ucs[0].Show();

break;

case "Delete":

ucs[1].Show();

break;

case "Show":

ucs[2].Show();

break;

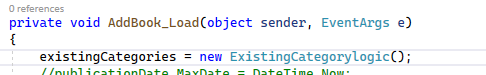
case "Update":

ucs[3].Show();

break;

}

}

****

**אף פעם לא להגדיר אוביקטים ב בבנאי**

## **Errrrrrrrrorrrrr Instance Object**

ExistingCategorylogic? existingCategorylogic = null;

List<string> categories;

List<string> secondaryCategorySelect;

public AddBook()

{

InitializeComponent();

}

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

{

try

{

existingCategorylogic = new ExistingCategorylogic();

}

catch

{

return;

}

publicationDate.MaxDate = DateTime.Now;

categories = new List<string>();

secondaryCategorySelect = new List<string>();

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;

}

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

{

secondaryCategory.DataSource = null;

secondaryCategorySelect.Clear();

if (existingCategorylogic != null)

{

object resFun = existingCategorylogic.ShowFromExistingCategories\_SubcategoryFromCategory(category.Text);

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"]);

}

}

}

else

{

MessageBox.Show("Errror");

}

secondaryCategory.DataSource = secondaryCategorySelect;

}

## **Passs parameters**

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

{

Code = "a",

FirstName\_Author = "b",

LastName\_Author = "c",

Title = "d",

PublicationDate = DateTime.Now,

Category = "A",

SecondaryCategory = "B"

};

updateBook1.setBook(book);

updateBook1.Show();

updateBook1.Focus();

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

{

ExistingCategorylogic existingCategorylogic = new ExistingCategorylogic();

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 )

{

ExistingCategorylogic existingCategorylogic = new ExistingCategorylogic();

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 void setBook(DataAccessLayer.Entities.Book upBook)

{

book = upBook;

}

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

{

categories = new List<string>();

secondaryCategorySelect = new List<string>();

HelpFunc.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;

}

}

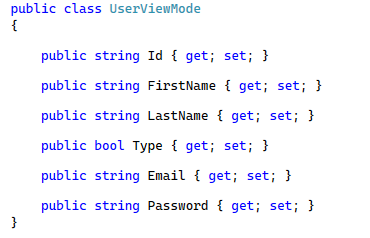
private void category\_SelectedIndexChanged(object sender, EventArgs e)

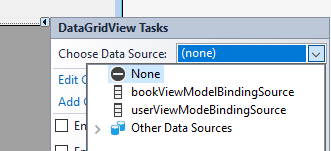
{

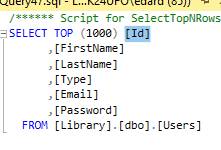
HelpFunc.category\_SelectedIndexChanged(secondaryCategorySelect, secondaryCategory, category.Text);

}

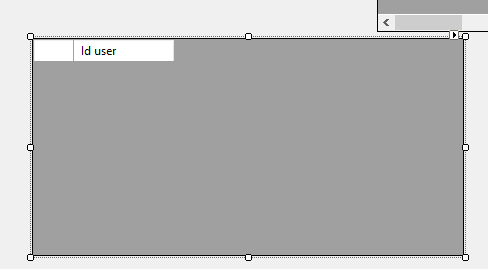
## **Sql create and play**

****

****

****

****

****

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);

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);

}

}

**Get data value**

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

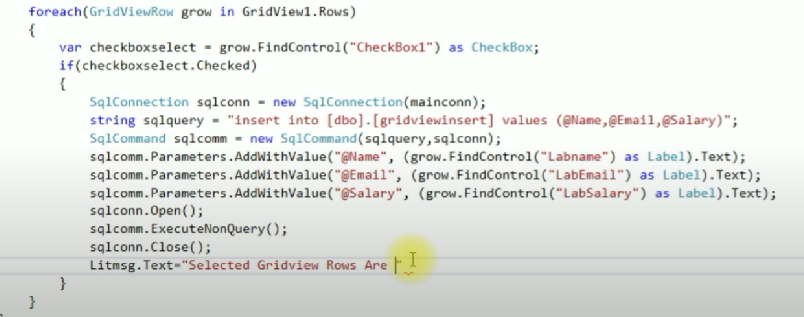
{

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

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

}

[**https://www.youtube.com/watch?v=\_sUaAFMhsgI**](https://www.youtube.com/watch?v=_sUaAFMhsgI)

****

## **Custom table data**

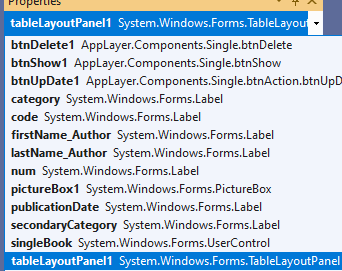
****

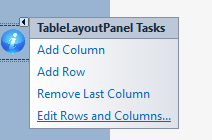
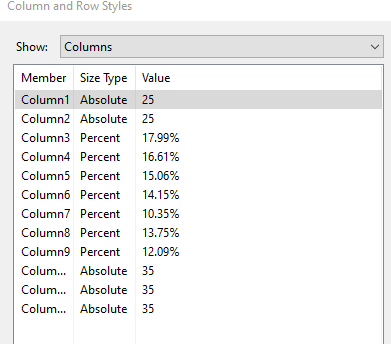
****

****

****

****

****

****

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

{

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

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

}

## **CheckComboBox as DropDownList :**

**CheckComboBoxItem.cs**

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; }

}

#endregion

}

}

**CheckComboBox.cs**

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;

}

}

// 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;

}

}

****

**Form**

public ShowBook()

{

InitializeComponent();

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

checkComboBox1.Items.Add(new CheckComboBoxItem("One", true));

checkComboBox1.Items.Add(new CheckComboBoxItem("Two", true));

checkComboBox1.Items.Add(new CheckComboBoxItem("Three", true));

//// wire up the check state changed event

this.checkComboBox1.CheckStateChanged += new System.EventHandler(this.checkComboBox1\_CheckStateChanged);

}

// 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;

}

}

}