GUI: Database programmering & OOP 2.HF

It & Data, Odense

ANNL/april 2019

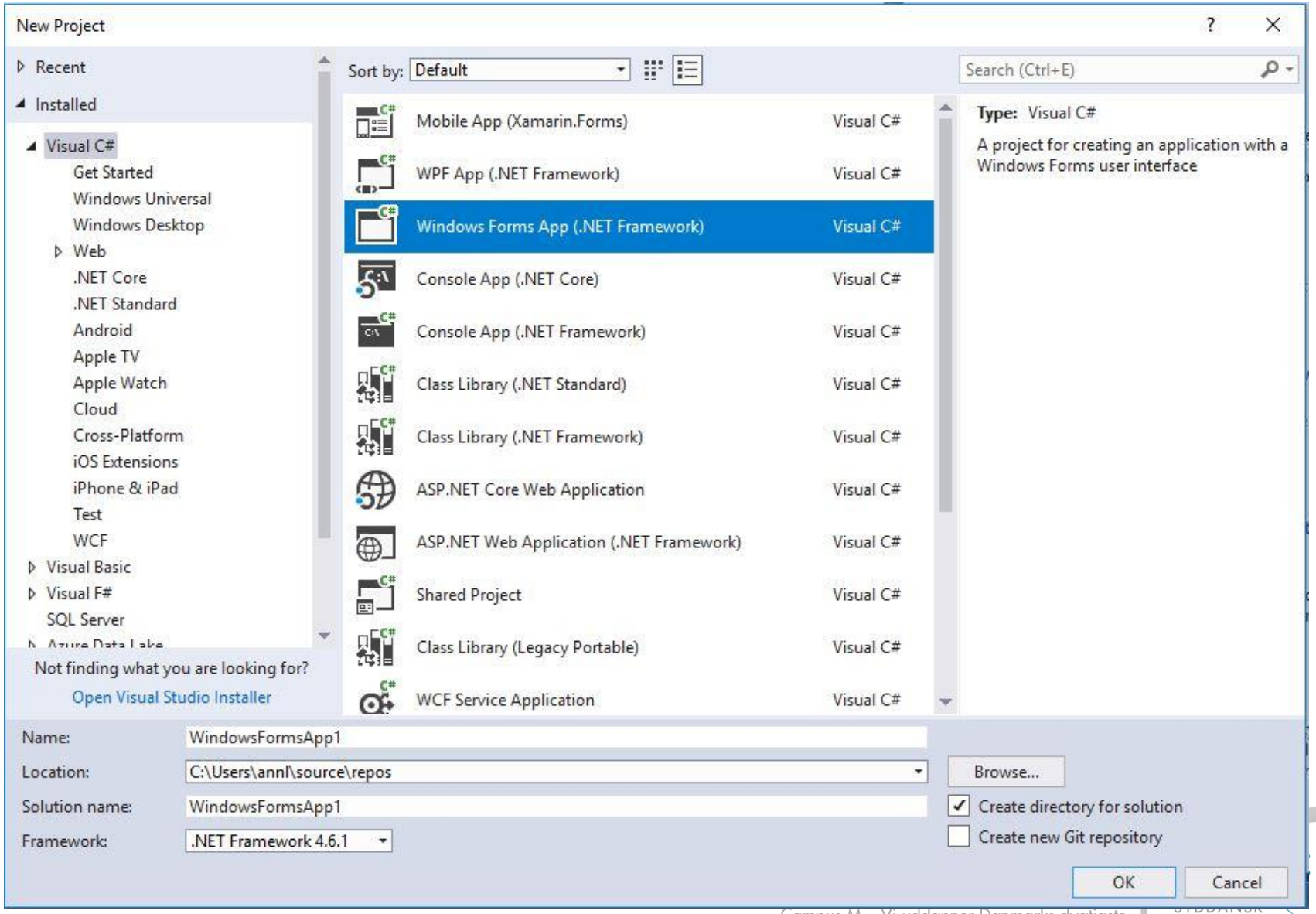
SYDDANSK ERHVERVSSKOLE

GUI = Graphical User Interface

Husk fokus på database design + OOP

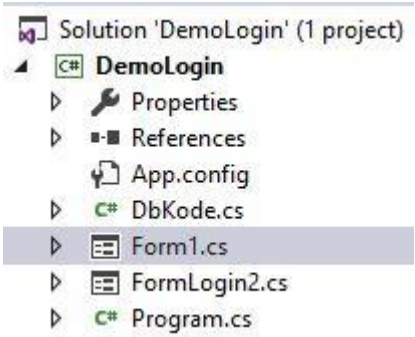
- Skal demonstrer OOP principper
- Vise data fra databasen (hentet via f.eks. Views)
- Gemme data (Form til indtastning + Stored Procedure)
- Redigere/slette data (Trigger)
- Det er ikke så vigtigt, hvilken type GUI, der oprettes.
- Vigtigt at GUI'en understøtter databasens design.
- Vigtigt at metoderne flyttes fra Form/windows ud i egen klasser og dermed understøtter OPP.

GUI – Opret et Windows Form projekt



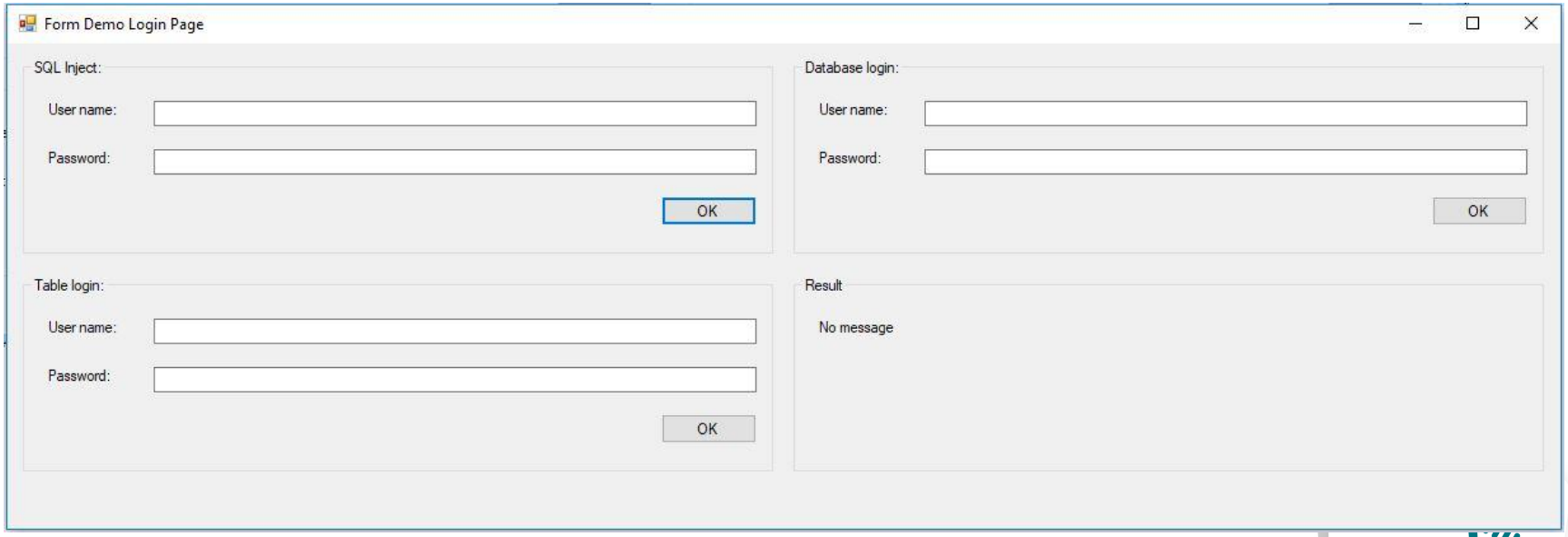
Windows Form: Demo projekt

- Dette projekt (namespace) indeholder to forms:
  - Form1.cs
    - Standard form, der oprettes fra starten af (kan omdøbes).
    - Indeholder en klasse ved navn ”FormDemo”
  - FormLogin2, der er tilføjet til projektet
    - Add -> Windows Form…
    - Indeholder en klasse ved samme navn ”FormLogin2”
- En klasse af eget design
  - DbKode.cs
    - Add -> Class…
- Resten af filerne i projektet er standard komponenter, der oprettes i starten.



Form demo: Login

- Denne form giver brugeren mulighed login på 3 forskellige måder
  1. ”SQL Inject” er den metode, I skal undgå i jeres løsning. Den bagved liggende kode, bruger input direkte i koden.
  2. ”Table login” er baseret på parametre, der fås fra en bruger tabel, hvilket er en mere sikker måde at håndtere bruger input på.
  3. ”Database login” er baseret på et server login, der også forhindre SQL injektioner.



Form demo: Koden bagved.

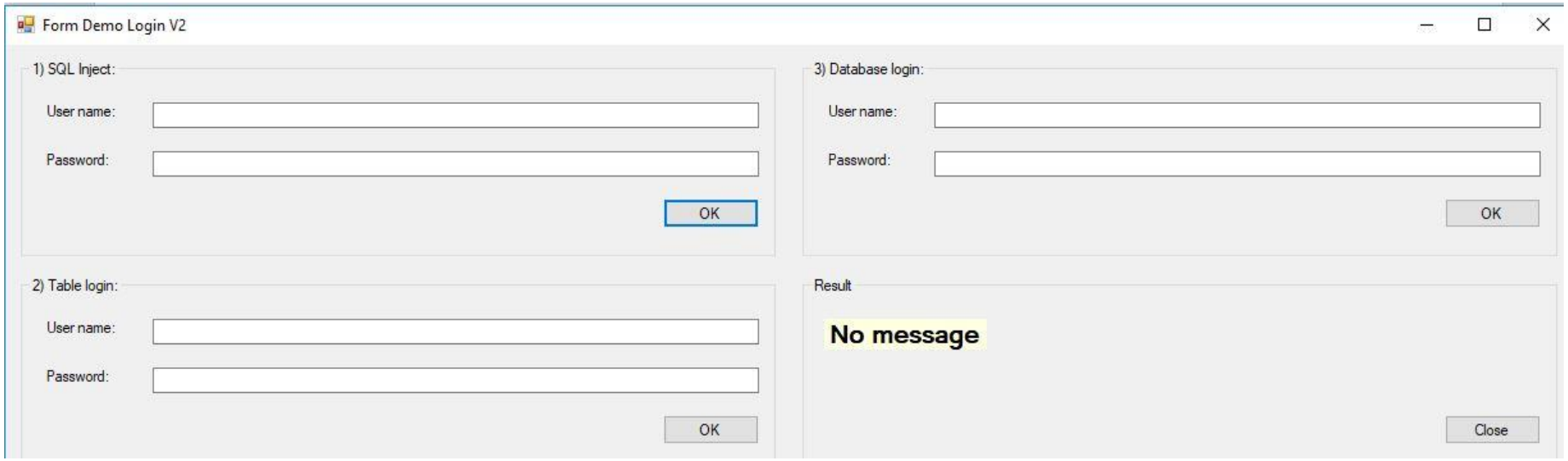
- Her ses et udsnit af koden ved tryk på knappen ”OK” (Tabel login)

```
private void btnTbl_Click(object sender, EventArgs e)
{
    // Own table login with parameters
    string strUser = txtUserTbl.Text;
    string strPword = txtPwordTbl.Text;
    labelMessage.Text = "";

    if (strUser != "" && strPword != "")
    {
        string connStr = "Data Source=(local); Initial Catalog=myNewDb; Integrated Secur
        SqlConnection conn = new SqlConnection(connStr);
        try
        {
            conn.Open();
            if (conn.State == System.Data.ConnectionState.Open)
            {
                string strSQL = "SELECT * FROM myUsers WHERE userName = @UserName AND pw

                SqlCommand cmd = new SqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = strSQL;
                cmd.Parameters.AddWithValue("@UserName", strUser);
                cmd.Parameters.AddWithValue("@Pword", strPword);

                // Udfør kommando
                SqlDataReader reader = cmd.ExecuteReader();
                // Tjek om data er klar
                if (reader.HasRows)
                {
                    labelMessage.Text = "Congrats! - The table user has access to the da
                }
```

- Der er rigtig meget kode i selve formen!

Form Login 2

- Ser umiddelbart ud som den første form.
- Den kan udfører de samme ting, som den første form:
  1. Login med risiko for SQL injektion
  2. Tabel login med parameter (større sikkerhed)
  3. Database login ved hjælp af bruger med login oprettet på serveren (MS SQL server).



Form Login 2

- Den store forskel på de to eksempler ligger i koden bagved.
- Koden er forkortet ned til det helt basale.

Button click, label og input tekst er alle elementer, der hører til på en form.

```
1 reference
private void btnSQL_Click(object sender, EventArgs e)
{
    //1
    labelMessage.Text = DbKode.testConn(txtUserSQL.Text, txtPwordSQL.Text, 1);
}

1 reference
private void btnTbl_Click(object sender, EventArgs e)
{
    //2
    labelMessage.Text = DbKode.testConn(txtUserTbl.Text, txtPwordTbl.Text, 2);
}

1 reference
private void btnDB_Click(object sender, EventArgs e)
{
    //3
    labelMessage.Text = DbKode.testConn(txtUserDb.Text, txtPwordDb.Text, 3);
}

1 reference
private void btnClose_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Form Login 2 / Klasse DbKode (Class)

- Resten af koden er flyttet til en separat klasse ”DbKode”
- Alle funktioner for de tre knapper er inkluderet i en fælles metode.

Samarbejdet mellem klassen og form sker via en metode og parametre

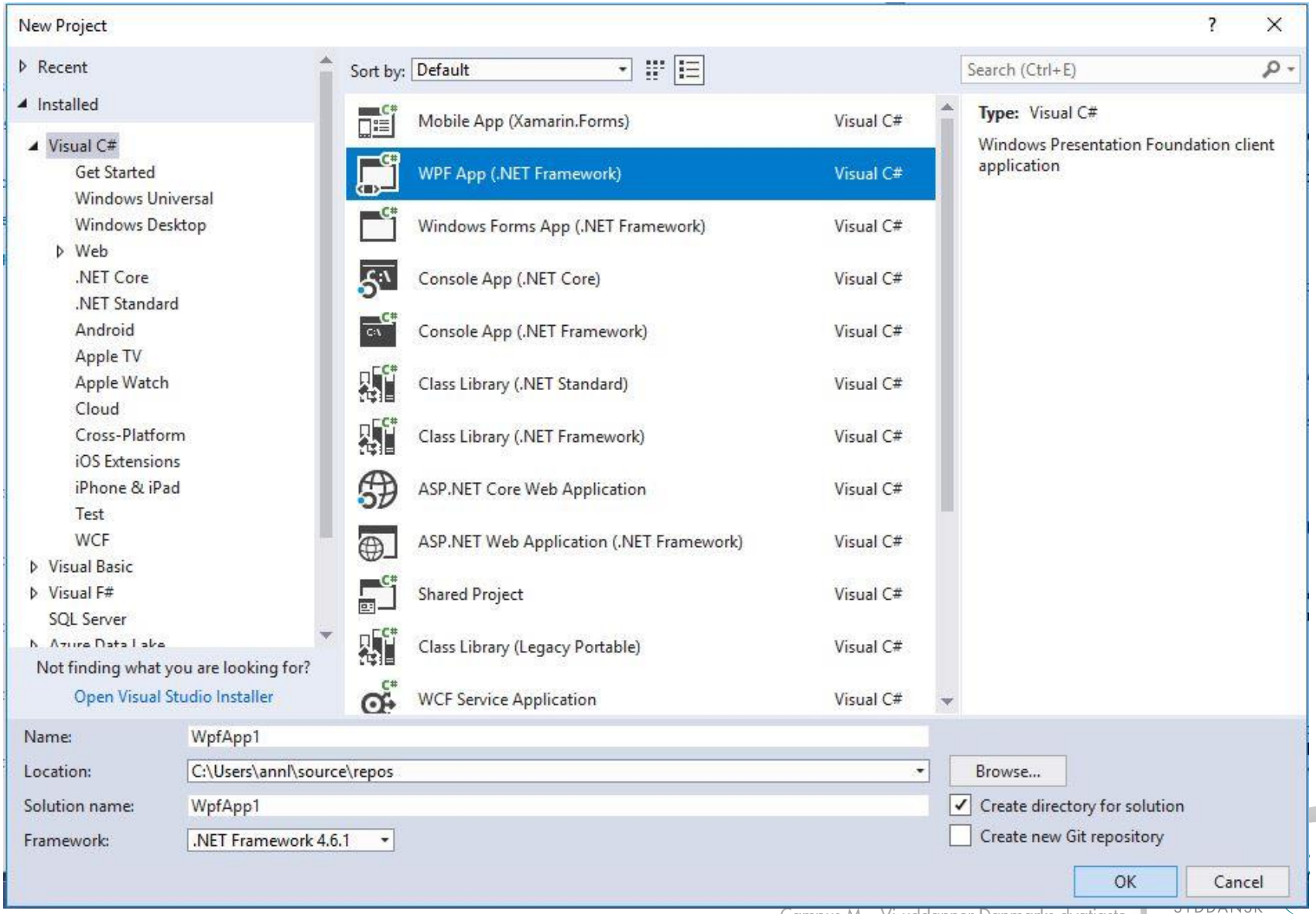
```
public class DbKode
{
    // SQL connection string - kun til intern brug
    private string connStr;
    // Selve database forbindelsen - kan anvendes af andre klasser via arv
    protected SqlConnection conn;
    // Mulighed for at tjekke om forbindelsen er OK - kan læses overalt
    public readonly bool bConnOK = true;

    3 references
    public static string testConn(string sUser, string sPword, int iChoice)
    {
        string strResult = "";

        if (sUser != "" && sPword != "")
        {
            string connStr = "";
            if (iChoice < 3)
            {
                connStr = "Data Source=(local); Initial Catalog=myNewDb; Integrated Security
            }
            else
            {
                connStr = "Data Source=(local); Initial Catalog=myNewDb; User Id=" + sUser +
            }


            SqlConnection conn = new SqlConnection(connStr);
            try
            {
                conn.Open();
```

Konklusion: Windows Form

- Der er flere måder at programmere en løsning. Men i dette fag er det OOP, der tæller.
- **Husk** derfor at jeres løsning skal demonstrere jeres evner til at programmere objekt orienteret (OOP).
- Derfor skal jeres kode så vidt muligt flyttes ud i separate klasser, der sidenhen kan instantieres som objekter.
- Så af de to eksempler, er det kun Form Login 2, der giver point til evalueringen.
- Til slut – et eksempel på, hvordan en ny form åbnes.

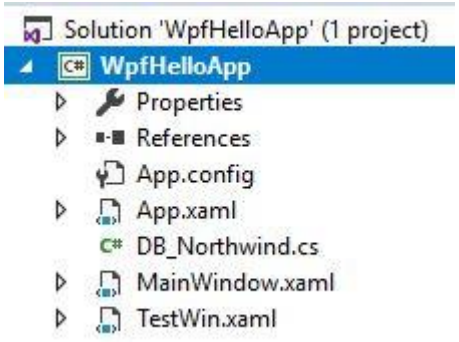
```
1 reference
private void btnOpen_Click(object sender, EventArgs e)
{
    //Open en ny Form ved navn FormLogin2

    //En måde at gøre det på:
    //FormLogin2 login2 = new FormLogin2();
    //login2.Show();

    //Den kortere udgave:
    new FormLogin2().Show();
}
```

GUI – Opret et WPF applikation



WPF projekt

- I lighed med første projekt, så indeholder WPF projektet to Vinduer (forms)
  - MainWindow
  - TestWin
- Vinduer i WPF er opdelt i design (grafisk) og XAML (kode)
- Ud over de to opdelinger af vinduet, er der tilhørende C# kode af type .cs
- I dette tilfælde er der tilføjet en klasse af eget design: DB_NorthWind
- Resten af filerne i projektet er standard komponenter, der oprettes i starten.



WPF kontra WinForms

- Det er nyere og dermed med på de nyeste tendenser inden for GUI
- Det bliver mere og mere anvendt (i Microsoft miljøer)
- Det er mere fleksibelt – men kræver derfor også mere arbejde.
- Ved brug af 3. parts elementer er chancen for at de er i WPF format større (fordi det er nyere).
- XAML gør det nemt at kreere og editere din GUI
- Læs mere her: https://www.wpf-tutorial.com/

Husk:

I lighed med Windows Form projekter gælder det om at få jeres egen kode lagt ud i separate klasser.

Forbindelse mellem database / GUI

- Database forbindelser findes i standard biblioteket ”System.Data”
  - ODBC (Open DataBase Connectivity)
  - OleDb (Object Linking and Embedding, Database)
  - Sql
  - SqlClient
- Der er ikke den store forskel mellem de forskellige muligheder – dog forlyder det, at OleDb er på vej ud.
- Generelt handler en forbindelse om 3 trin:
  1. Connection
  2. Command
  3. Retrieve data
- Nyttige informationer fås via dette link:

  Microsoft .NET ”Modtage og modifikation af data i ADO.NET”

https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/retrieving-and-modifying-data

Eksempel: login

- Dette eksempel er knyttet til et Form objekt.
- Omskriv eksemplet, så det kan bruges via egen metode i en klasse.
- Husk at anvende parametre.
- Husk try-catch-throw

```
// Own table login with parameters
string strUser = txtUserTbl.Text;
string strPword = txtPwordTbl.Text;
labelMessage.Text = "";

if (strUser != "" && strPword != "")
{
    string connStr = "Data Source=(local); Initial Catalog=myNewDb; Integrated Security=true;";
    SqlConnection conn = new SqlConnection(connStr);
    try
    {
        conn.Open();
        if (conn.State == System.Data.ConnectionState.Open)
        {
            string strSQL = "SELECT * FROM myUsers WHERE userName = @UserName AND pword = @Pword";

            SqlCommand cmd = new SqlCommand();
            cmd.Connection = conn;
            cmd.CommandText = strSQL;
            cmd.Parameters.AddWithValue("@UserName", strUser);
            cmd.Parameters.AddWithValue("@Pword", strPword);

            // Udfør kommando
            SqlDataReader reader = cmd.ExecuteReader();
            // Tjek om data er klar
            if (reader.HasRows)
            {
                labelMessage.Text = "Congrats! - The table user has access to the database";
            }
            else
            {
                labelMessage.Text = "Sorry! - Access denied for the table user";
            }
            reader.Close();
        }
        else
        {
            labelMessage.Text = "Sorry! - Connection is not open";
        }
        conn.Close();
    }
    catch (Exception ex)
    {
        labelMessage.Text = ex.Message;
        //throw;
    }
}
```

Kort og godt om data forbindelser

- Brug Try-Catch for at fange eventuelle fejl. Der er mange ting, der kan gå galt, når der arbejdes med eksterne data.
- Husk ”throw” så fejlen sendes videre fra ”forretningslogikken” til GUI’en (præsentationslaget)
- Start ALTID med at teste forbindelsen til databasen. Uden forbindelse er resten ligegyldigt.
  - F.eks. SqlConnection (fra System.Data.SqlClient)
- Når der er hul igennem, kobles et kommando objekt på forbindelsen. Husk at bruge parametre så vidt det er muligt
  - SqlCommand
- Ved data retur (select statements), anvendes en data beholder
  - SqlDataReader

Eksempler

- Windows Form – eksempel på en simpel salgs GUI

https://docs.microsoft.com/en-us/visualstudio/data-tools/create-a-simple-data-application-by-using-adonet?view=vs-2019

- WPF (Vær opmærksom på at dette eksempel bruger Entitity FrameWorks 6, hvilket ikke er en ubetinget god ide i dette projekt).

https://docs.microsoft.com/en-us/visualstudio/data-tools/create-a-simple-data-application-with-wpf-and-entity-framework-6?view=vs-2019

- Udfør en kommando:

https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/executing-a-command

- Eksempler på brug af DataAdapter og DataReader

https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/dataadapters-and-datareaders

Nyttige link:

WinForm

- DataGridView control

https://docs.microsoft.com/en-us/dotnet/framework/winforms/controls/how-to-bind-data-to-the-windows-forms-datagridview-control

- Tutorial

http://csharp.net-informations.com/datagridview/csharp-datagridview-tutorial.htm

WPF

- DataGrid gennemgang

https://www.c-sharpcorner.com/UploadFile/mahesh/datagrid-in-wpf/

- DataGrid kolonner

https://www.wpf-tutorial.com/datagrid-control/custom-columns/

- DataGrid detaljer

https://www.wpf-tutorial.com/datagrid-control/details-row/

Slut på GUI
Kombination af database programmering & OOP