

SQL Server Express og C#

I denne sektion skal du lære at oprette en database med SQL Server Express. Du kan gøre det med den software, der følger med Visual Studio. Når du har oprettet en database lærer du hvordan man laver udtræk fra databasen og viser dem i en Windows formular. Du skal også lære hvordan man navigere rundt i de poster der er i databasen, og hvordan man tilføjer nye poster.

Hvis du foretrækker at arbejde med en Access database, kan springe denne sektion over. Når du når til oprettelse af forbindelsen til databasen, gennemgår vi begge måder – Access og SQL Server Express.

Hvad er SQL Server Express?

SQL Server Express er et databasesystem fra Microsoft. Det er en simpel version af SQL Server, som bliver brugt meget i store forretningsløsninger rundt omkring i verden. Heldigvis har Microsoft lavet en Express udgave som kan downloades gratis.

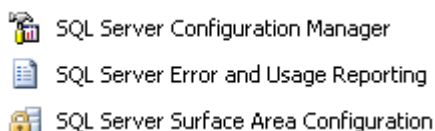
Du skal derfor installere SQL Server Express sammen med din C#. Hvis du har gjort det ser du følgende i din startmenu:



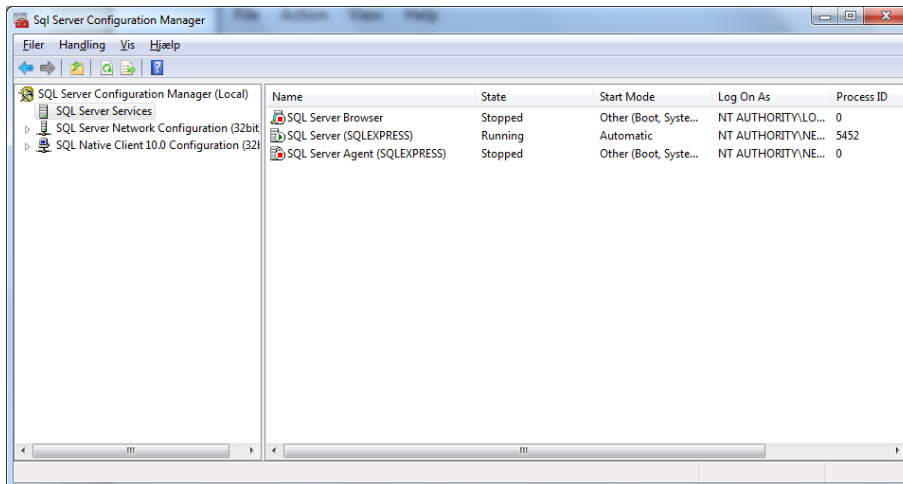
Hvis du kan se punktet SQL Server er programmet installeret.

Det punkt der hedder SQL Server Management Studio Express er et selvstændigt program, der lader dig oprette og vedligeholde dine databaser uden at du behøver at starte C#.

Når du har downloadet og installeret SQL Server Express skal du være sikker på at den kører. Klik på menupunktet Configuration Tools for at se følgende menu:



Vælg SQL Server Configuration Manager. Du ser nu følgende vindue:

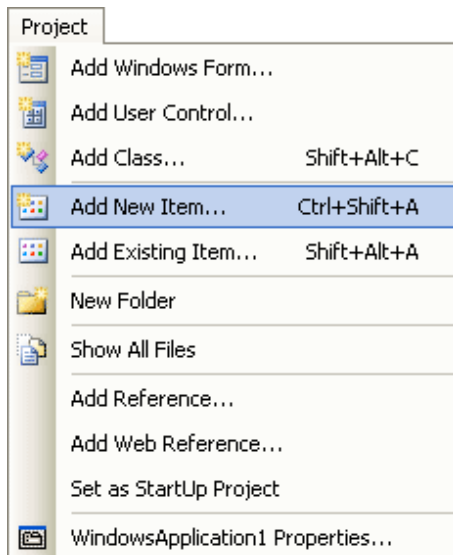


I den højre side skal du sikre dig at statussen er Running. Hvis ikke så højre klik og vælg Start. Du kan derefter lukke vinduet Configuration Manager.

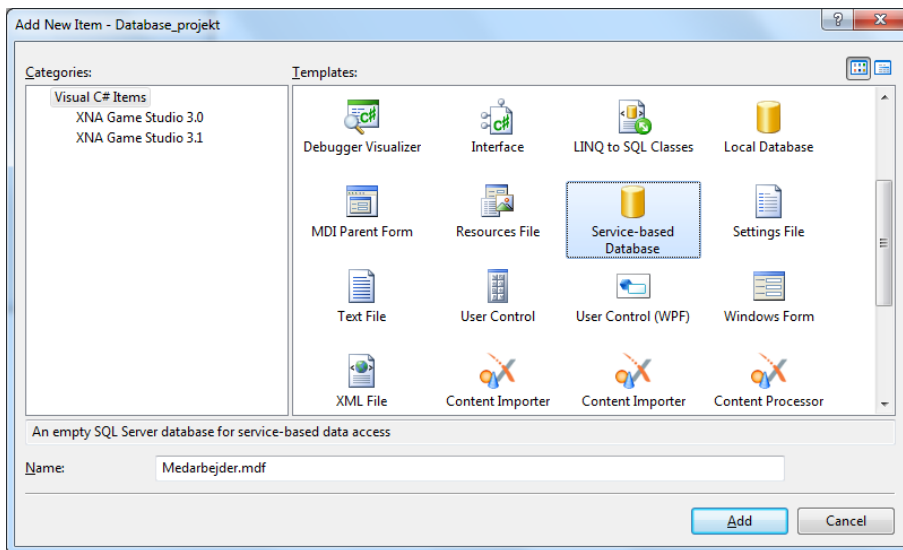
Hvordan opretter du en database med SQL Server Express

Når du skal oprette en database starter du med at starte C#. Opret et nyt Windows projekt ved at vælge **File** og **New Project**. Navngiv dit projekt – vi skal dog ikke bruge formularen til noget. Uden SQL Server Management Studio Express installeret er du nødt til at have en Windows Application for at oprette en SQL Server Express database.

Vælg **Project** og **Add New Item**:



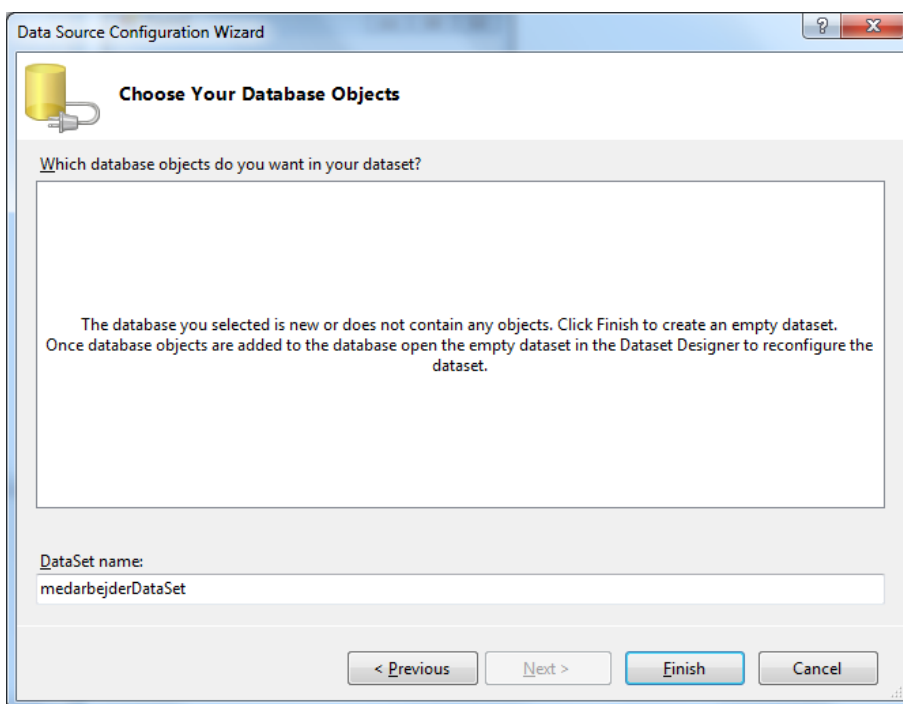
Når du klikke på Add New Item ser du følgende dialogboks:



Giv din database et navn. Kald den **Medarbejder.mdf**. Vi vil oprette en database med fiktive personer på en arbejdsplads, hvor vi vil give dem en jobbeskrivelse.

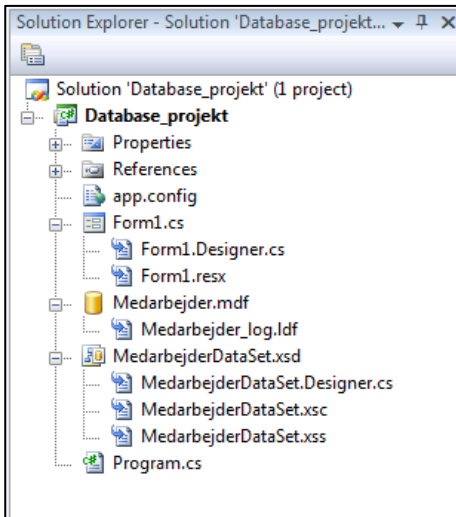
Klik på Add. I C# 2010 bliver du spurgt om hvilken database model du vil have. Lad den blive ved standard valget **Dataset** og klik Next.

I C# 2008 ser du følgende:



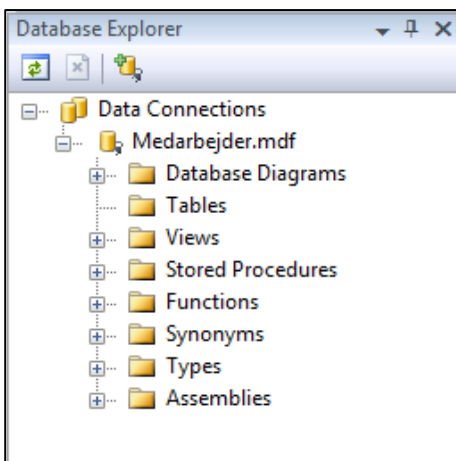
Klik på **Finish** for at vende tilbage til C#.

I alle versioner af C# ser det ud som om intet er sket. Men hvis du ser efter i Solution Explorer i højre side kan du se din database er tilføjet til dit projekt:

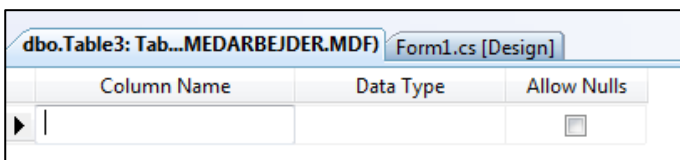


Din database er dog tom. Vi skal have tilføjet en tabel. Højre klik på **Medarbejder.mdf**. Vælg **Open** i menuen.

Du ser nu Database Explorer:



Højre klik på **Tables** og vælg **Add New Table** i menuen. Du ser nu en ny tabel, der dukker op i hovedvinduet:



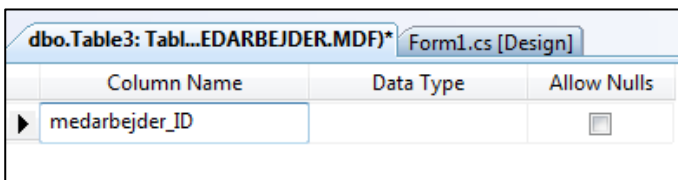
Column er de datafelter du vil have med i din tabel. Det er meningen at du skriver et navn til datafeltet, og derefter specificer hvilken type data, der skal opbevares i feltet. Det kan være tekst, Yes/No, værdier osv. Allows Nulls betyder "Skal feltet udfyldes?" Hvis du f.eks. har et felt til et mellemnavn, vil det indimellem være tomt, da ikke alle har et mellemnavn, kan det være praktisk at feltet ikke behøver at bliver udfyldt. Hvis du til gengæld har brug for data, som f.eks. et identifikationsnummer, afkrydser du ikke Allow Nulls.

Vi opretter en meget simpel tabel med bare fire felter:

medarbejder_ID
for_Navn
efter_Navn
job_Titel

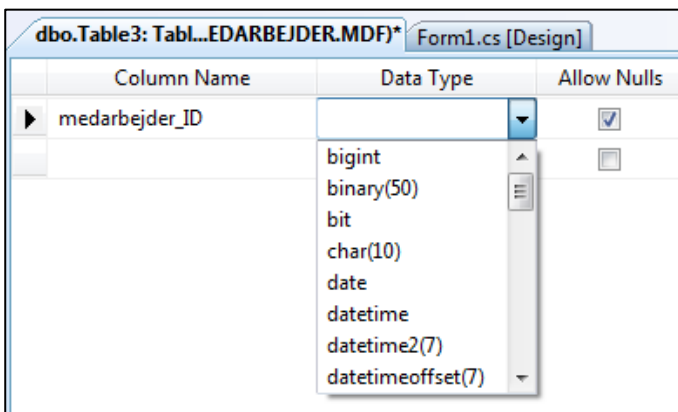
Det første felt Medarbejder_ID skal være et tal. Vi lader databasen tage sig af det. Hver gang en medarbejder tilføjes vil SQL Server Express tilføje et nyt nummer til medarbejderen. Det kaldes også for autonummerering i database verdenen.

Tast Worker_ID i kolonnen Name:



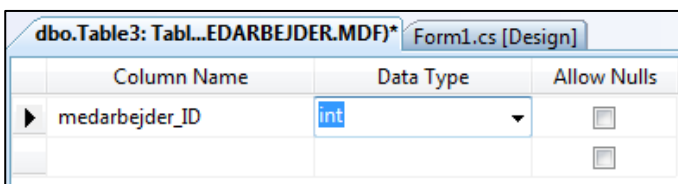
Column Name	Data Type	Allow Nulls
medarbejder_ID		<input type="checkbox"/>

Det næste du skal fortælle SQL Server Express er hvilken type data, der skal være i medarbejder_ID feltet. Klik i kolonnen Data Type. Du ser følgende listefelt med de muligheder der er:



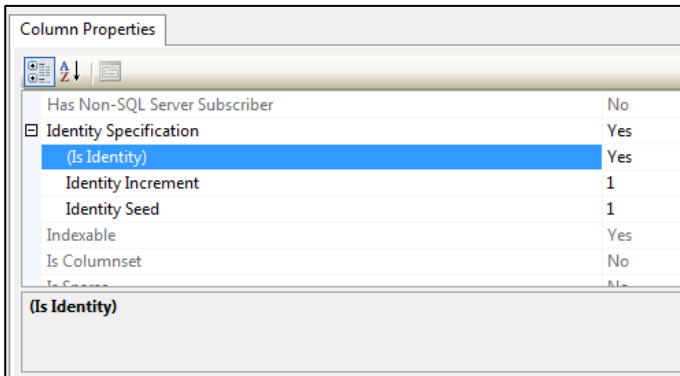
Column Name	Data Type	Allow Nulls
medarbejder_ID	<ul style="list-style-type: none">bigintbinary(50)bitchar(10)datedatetimedatetime2(7)datetimeoffset(7)	<input checked="" type="checkbox"/>

Som du kan se er der nok at vælge imellem. Vælg **int** i listen. Lad feltet Allow Null være. Dine indstillinger ser nu sådan ud:



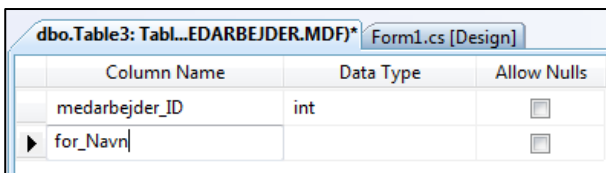
Column Name	Data Type	Allow Nulls
medarbejder_ID	int	<input type="checkbox"/>

Der er en ting tilbage vi skal have gjort med dette felt. Se i bunde af din skærm. Her ser du en liste med betingelser. Den vi skal se på er **Identity Specification**. Sæt **Identity Specification** til Yes, og så dukker **Identity Increment** og **Identity Seed** op:



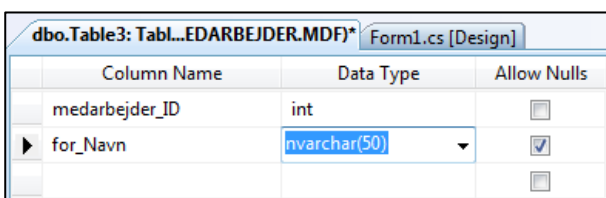
Når du angiver disse værdier vil SQL Server Express lægge 1 til feltet medarbejder_ID når der tilføjes en ny medarbejder.

Klik i kolonnen Column Name og indtast det næste felt:



Vælg datatypen nvarchar(50). Datatypen **varchar** er en forkortelse for variable-length character string. Med nvarchar er **n** en forkortelse for Unicode, og informationerne vil blive gemt i formatet UTF-16. Brug nvarchar når du skal gemme data, der indeholder ikke Engelske karakterer. Ellers kan du bruge varchar. Det samme gælder for de andre datatyper der starter med n.

Afmærk feltet **Allow Nulls** og dit nye felt vil se sådan ud:



Angiv følgende værdier for de to andre felter:

Column Name: efter_Navn

Data Type: nvarchar(50)

Allow Nulls: Yes

Column Name: job_Titel

Data Type: nvarchar(50)

Allow Nulls: Yes

Din tabel ser nu sådan ud:

Column Name	Data Type	Allow Nulls
medarbejder_ID	int	<input type="checkbox"/>
for_Navn	nvarchar(50)	<input checked="" type="checkbox"/>
efter_Navn	nvarchar(50)	<input checked="" type="checkbox"/>
job_Titel	nvarchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Du kan nu angive en primærnøgle. En primærnøgle er med til at sikre at alle posterne i det givne felt er unikke – der er altså ikke to felter der kan have den samme værdi. Man kan ikke lade fornavnet være en primær, da det er oplagt at man kan have medarbejder med det samme fornavn. Derimod er kolonnen med medarbejder_ID unik og kunne bruges som primærnøgle. Hvis vi havde en anden tabel kunne vi bruge primærnøglen og fremmednøglen til at linke de to tabeller sammen. SQL Serveren er det man kalder en relations database, og primærnøglen bruges meget til at linke de forskellige elementer sammen.

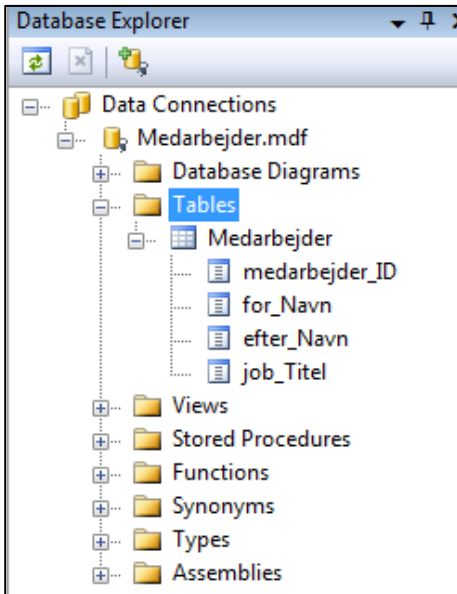
Når du skal angive en primærnøgle højre klikker du på medarbejder_ID. I menuen vælger du Primary Key. Vi vil dog ikke tilføje en primærnøgle til vores tabel, da vi vil holde tingene simple. Hvis du vil være en database guru, er du nødt til at have en forståelse for de aspekter, der findes i en SQL Server og SQL Server Express.

Lad os fortsætte.

Klik på **File** og **Save All** for at gemme dit arbejde. Du bliver bedt om at angive et navn til din nye tabel.

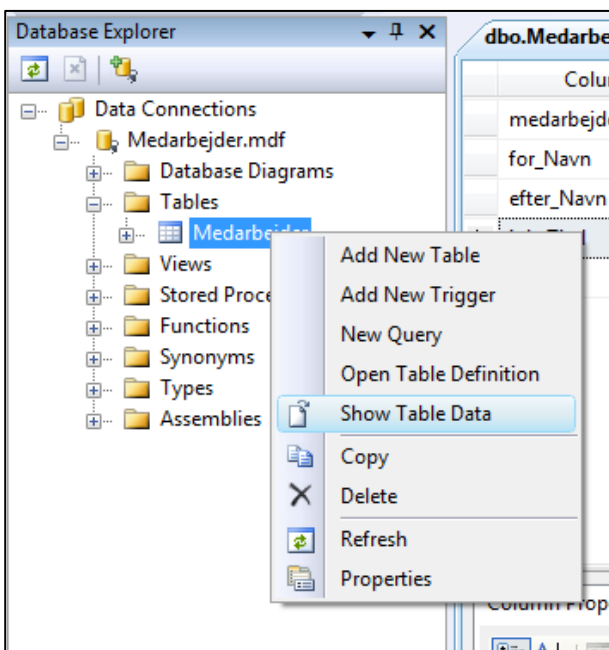
Kalden for **tblMedarbejder**:

Klik OK og du vender tilbage til hovedvinduet og Database Explorer. Udvid Table sektionen for at se dine nye kolonner:



Nu skal vi bare have indtastet lidt prøve data i tabellen.

Når du skal tilføje data til din tabel højre klikker du på tabellens navn. Du ser nu følgende menu:



Vælg **Show Table Data**. Du ser nu en ny fane:

	medarbejder_ID	for_Navn	efter_Navn	job_Titel
*	NULL	NULL	NULL	NULL

Du ser alle kolonnenavnene, som venter på at blive udfyldt. Når du skal tilføje data klikke du i en celle og starter med at taste.

Klik i kolonnen for_Navn. Tast et fornavn. Klik i efter_Navn og tast et efternavn. Klik i job_Titel og indtast en jobtitel. Indtast de samme detaljer som er vist nedenfor, hvis du foretrækker det:

Medarbejder: Qu...EDARBEJDER.MDF		dbo.Medarbejder...EDARBEJDER.MDF		
	medarbejder_ID	for_Navn	efter_Navn	job_Titel
✎	NULL	Hanne	Jensen	IT chef
*	NULL	NULL	NULL	NULL

Bemærk advarselssymbolet i cellerne. Det sker når en celledata ændres. Feltet medarbejder_ID er stadig NULL i billede ovenfor. Når du klikker ned i den næste række, vil nummeret automatisk vises i den første celle for medarbejder_ID:

Medarbejder: Qu...EDARBEJDER.MDF		dbo.Medarbejder...EDARBEJDER.MDF		
	medarbejder_ID	for_Navn	efter_Navn	job_Titel
	1	Hanne	Jensen	IT chef
▶*	NULL	NULL	NULL	NULL

Vi har nu oprette den første række i vores database tabel. Udfyld med lidt flere rækker. Du kan bruge informationerne vist nedenfor:

Medarbejder: Qu...EDARBEJDER.MDF		dbo.Medarbejder...EDARBEJDER.MDF			Start
	medarbejder_ID	for_Navn	efter_Navn	job_Titel	
	1	Hanne	Jensen	IT chef	
	2	Jens	Hansen	Programmør	
	3	Helle	Clausen	Logistik	
	4	Frederik	Petersen	Lagermedarbejder	
	5	Jette	Hilbert	Kantine	
▶*	NULL	NULL	NULL	NULL	

Gem dit arbejde. Du har nu oprette din første SQL Server Express database. Det er et stort område og der er skrevet store tykke bøger om dette emne. Vi har kun berørt det helt grundlæggende. Det vi skal lave nu er noget kode i C#, så vi kan åbne databasen.

Forbind en SQL Server Express Database med C#

Luk evt. dit projekt ned hvis du har projekter åbnet. Opret et nyt projekt.

Når du skal finde din MDF database du oprettede i den forgående øvelse, skal du se efter den i din Projekt mappe under Visual Studio. Dobbeltklik på det navn du gav dit projekt og du ser nu din medarbejder.mdf.

Kopier den til en ny placering som f.eks. i en mappe i roden i C:\ du kalder database. Det er så du ikke skal arbejde med alt for lange stier. Når du har kopieret den over i dit C drev har den stien:

C:\database\medarbejder.mdf

Hvis du lade den ligge vil stien være noget i stil med dette:

C:\\Documents and Settings\\pc_name\\My Documents\\Visual Studio 2005\\Projects\\cSharp\\dbtests\\medarbejder.mdf

Hvilket er noget langt i det!

Hvis du arbejder med Vista eller Windows 7 kan du kopiere databasefilen over til din Dokument mappe. Stien er så noget i stil med:

"C:\\brugere\\ejer\\Dokumenter\\medarbejder.mdf"

Opret forbindelse til en SQL Server Express Database

Når du skal oprette forbindelse til en SQL Server Express database, skal du først oprette et SQL Connection objekt. Du skal bruge noget man kalder en connection string, der skal fortælle C# hvor databasen er.

Når du skal oprette et connection objekt dobbeltklikker du på din tomme formular. Lige udenfor Form Load event tilføjer du følgende kode:

System.Data.SqlClient.SqlConnection con;

Dit kodevindue ser nu sådan ud:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    System.Data.SqlClient.SqlConnection con;
    private void Form1_Load(object sender, EventArgs e)
    {

    }
}
```

Inde i Form Load event tilføjer du følgende:

con = new System.Data.SqlClient.SqlConnection();

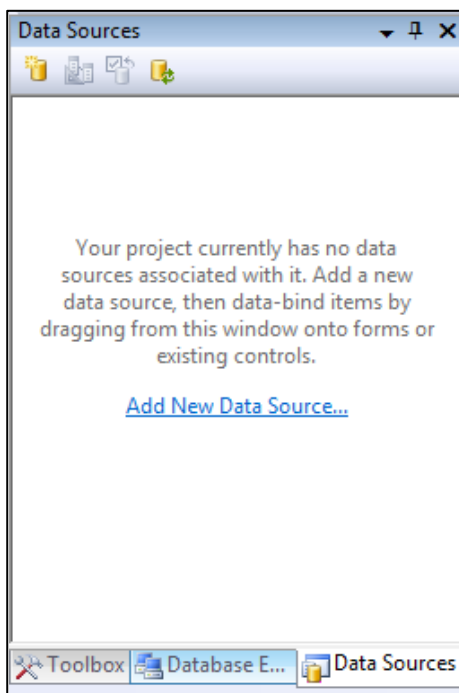
Når din formular åbnes oprettes et nyt SQL Connection objekt med navnet con. Det skal se sådan ud:

```

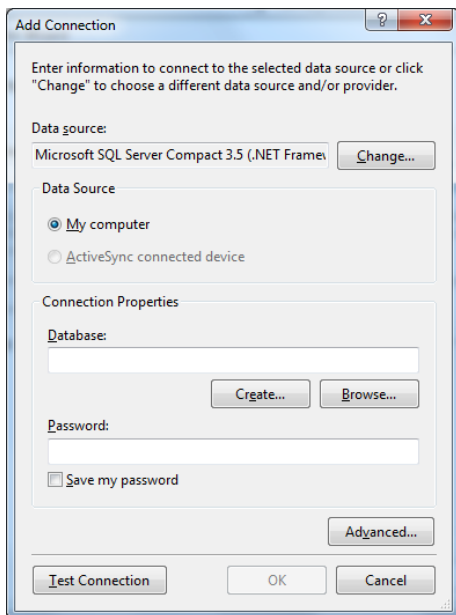
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    System.Data.SqlClient.SqlConnection con;
    private void Form1_Load(object sender, EventArgs e)
    {
        con = new System.Data.SqlClient.SqlConnection();
    }
}

```

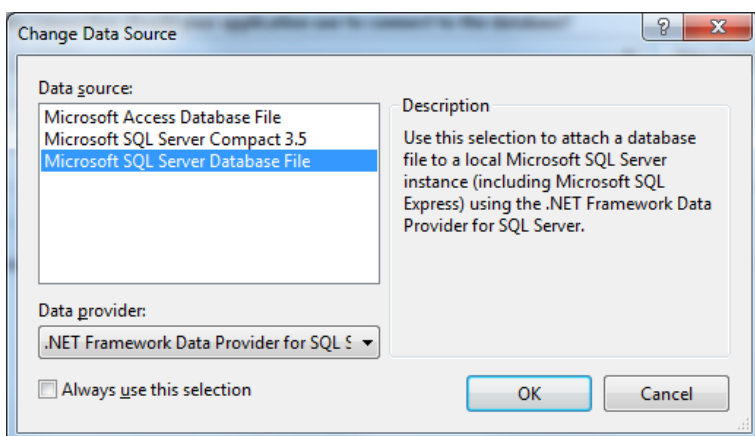
Nu da vi har et connection objekt kan vi tilgå det med betingelsen `ConnectionString`. For at se denne string vælger du menupunktet **Data**. Vælg derefter **ShowData Sources**. Du ser en ny fane i Solution Explorer:



Følger du denne guide. Klik på **Add New Data Source** for at starte en guide. I det første vindue skal du være sikker på at Database er valgt og klik derefter **Next** for at se trinnet **Choose your Data Connection**. Klik på knappen **New Connection**. Du ser nu følgende:

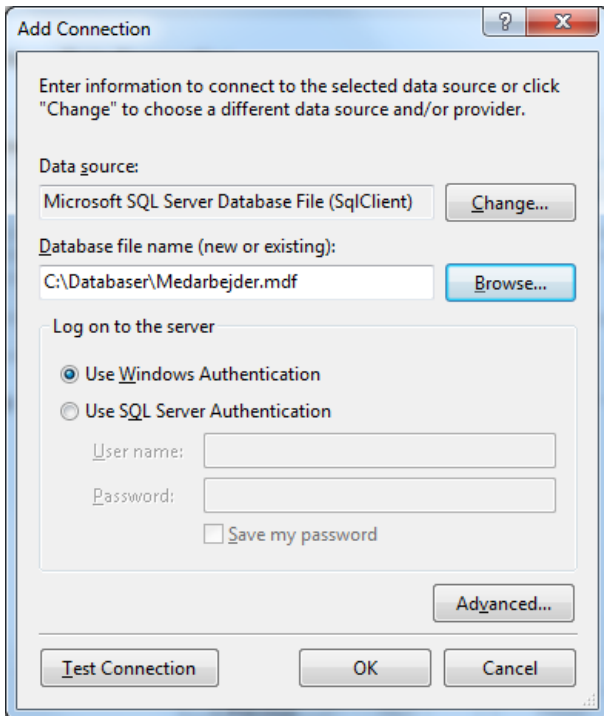


Feltet Data Source har knappen Change. Klik for at få følgende dialogboks:

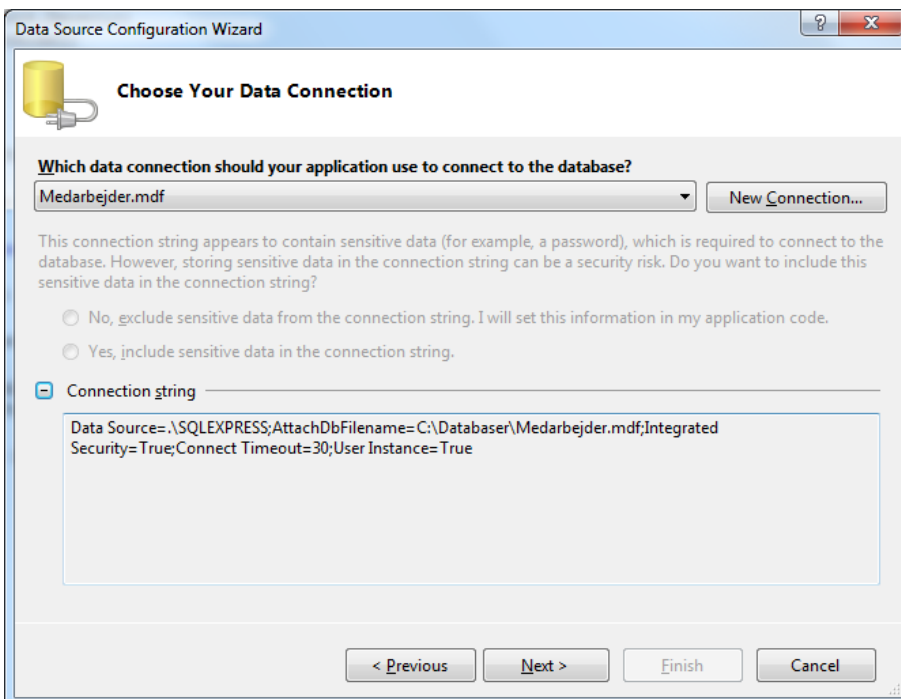


Vælg **Microsoft SQL Server Database File (SqlClient)**. Klik derefter OK.

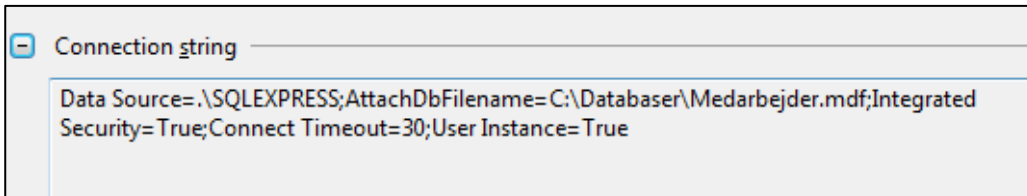
Klik på knappen Browse for at finde din database. Dialogboksen **Add Connection** ser sådan ud:



Klik på knappen Test Connection for at undersøge om det hele virker. Klik derefter på OK for at vende tilbage til trinnet **Choose your Data Connection**. Udvid området **Connection String** og din dialogboks ser nu sådan ud:



Marker din Connection string og lave en kopi.



Vend tilbage til din event Form Load i dit kodevindue. Indsæt din connection string. Du ser en masse røde understregninger, men det skal du ikke bekymre dig om nu. Cancel din guide da vi ser færdig med den – det eneste vi var interesseret i var denne connection string.

Lige efter linjen `SqlConnection()` skriver du følgende:

`con.ConnectionString`

Tast derefter et lighedstegn og et anførselstegn:

`con.ConnectionString = "`

Flyt nu din connection string hen efter anførselstegnet:

`con.ConnectionString = "DataSource=.\SQLEXPRESS; AttachDbFilename =C:\databaser\medarbejder.mdf;Integrated Security=True;Connect Timeout=30;User Instance=True";`

I slutningen af den lange connection string taster du nu endnu et anførselstegn. Afslut linjen med det sædvanlige semikolon. Dit kodevindue ser nu sådan ud:

```
private void Form1_Load(object sender, EventArgs e)
{
    con = new System.Data.SqlClient.SqlConnection();

    con.ConnectionString = "Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\databaser\Medarbejder.mdf;Integrated
Security=True;User Instance=True";
}
```

Bemærk at du stadig har fejl understregninger i din connection string. . Det er backslash tegnet der er problemet. Det opfattes som et specialtegn i C#. Det er vi nød til at omgå og det kan gøres ved at skrive et ekstra backslash lige foran:

```
private void Form1_Load(object sender, EventArgs e)
{
    con = new System.Data.SqlClient.SqlConnection();

    con.ConnectionString = "Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\\databaser\\Medarbejder.mdf;Integrated
Security=True;User Instance=True";
}
```

Der er nu seks backslash tegn i koden ovenfor. To før `SQLEXPRESS`, to før `database`, og to før `medarbejder.mdf`. Tilføj flere backslash til din kode for at slippe for eventuelle fejl.

Det vores kode gør er at det fortæller C# hvor databasen er, og angiver derefter nogle få betingelser. Du kan angive flere databasebetingelser her. For eksempel kan du angive et brugernavn hvis databasen kræver det: Du gør sådan:

User ID=your_user_name;

Efter din connection string kan du prøve at åbne en forbindelse til database. Igen skal vi bruge vores con objekt:

con.Open();

Når forbindelsen er oprettet skal vi skrive kode for at få fat i vores poster. Når det er gjort kan vi lukke forbindelsen:

con.Close();

tilføj to meddelelsesboks til din kode og derefter skal dit kodevindue se sådan ud:

```
System.Data.SqlClient.SqlConnection con;

private void Form1_Load(object sender, EventArgs e)
{
    con = new System.Data.SqlClient.SqlConnection();
    con.ConnectionString = "Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\\Databaser\\Medarbejder.mdf;Integrated
Security=True;Connect Timeout=30;User Instance=True";

    con.Open();

    MessageBox.Show("Databasen er åben");

    con.Close();

    MessageBox.Show("Databasen er lukket");
}
```

Kør dit program og test det. Du ser nu først en meddelelse med "Databasen er åben" efterfulgt af meddelelsen "Databasen er lukket". Derefter skulle formularen starte.

Tillykke! Alt det hårde arbejde og du har nu oprettet en forbindelse til din SQL Server Express database!

Forbindelsen til databasen oprettes med en Open method i dit connection objekt:

con.Open();

Luk forbindelsen på samme måde:

con.Close();

Du kan også bruge kommandoen Dispose i slutningen hvis du vil. Det foretager oprydningen for:

con.Dispose();

Når du har oprettet en forbindelse til en database skal du lære om Datasæt og DataAdapter.

Datasæt og Data Adapters i C#

Vi har oprettet en forbindelse til databasen. Det næste skridt er at lave et udtræk fra vores tabel. Det skal vi bruge et datasæt og en DataAdapter til.

Et datasæt er der hvor al din data befinder sig når den hentes fra databasens tabel. Tænk på det som et gitterværk du ser i et regneark. Kolonnerne i gitteret er kolonnerne fra din database tabel. Rækkerne repræsenterer en enkelt post i tabellen.

Datasættet skal udfyldes med data. Men da datasættet og dit Connection objekt ikke kan se hinanden har de brug for lidt hjælp i midten – DataAdapteren. DataAdapteren vil udfylde Datasættet med poster fra databasen.

Vi skal derfor oprette yderligere to objekter, et datasæt og en DataAdapter. Det gælder uanset om du bruger en SQL Server database eller en Access database.

Når du skal oprette et Datasæt objekt tilføjer du følgende kode før din form load event:

DataSet ds1;

Inden i dit form load event opretter du et nyt objekt ud fra det Dataset typen vi kaldte ds1:

ds1 = new DataSet();

Det ser nu sådan ud:

```
System.Data.SqlClient.SqlConnection con;
DataSet ds1;

private void Form1_Load(object sender, EventArgs e)
{
    con = new System.Data.SqlClient.SqlConnection();
    ds1 = new DataSet();

    con.ConnectionString = "Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\\Databaser\\Medarbejder.mdf;Integrated
Security=True;Connect Timeout=30;User Instance=True";

    con.Open();

    MessageBox.Show("Databasen er åben");

    con.Close();

    MessageBox.Show("Databasen er lukket");

    con.Dispose();
}
```

Til DataAdapteren skal du bruge følgende uden for form load event:

System.Data.SqlClient.SqlDataAdapter da;

Vi opretter en DataAdapter variabel vi kalder da.

Inden i form load event kan vi oprette et nyt objekt ud fra vores da variabel:

```
string sql = "SELECT * From tblMedarbejder";  
da = new System.Data.SqlClient.SqlDataAdapter( sql, con );
```

Den første linje er den same:

```
string sql = "SELECT * From tblMedarbejder";
```

Her oprettes en string variable der hedder sql. SQL står for Structured Query Language. Det er et sprog, der bruges til at hente poster ud fra en database, og forskellige udgaver at det bruges i alle database systemer. Når du bruger SQL Server, bruger du Structured Query Language på selve databasen. (SQL Serverens variant hedder T-SQL. T står for Transact.)

Nogle af de nøgleord du ser i SQL er SELECT, UPDATE, WHERE og mange andre. Symbolet * betyder "alle poster". Det vi siger her er "Vælg alle poster fra tabellen, der hedder tblMedarbejder".

Objektet DataAdapter vil bruge SQL kommandoer til at hente poster fra databasen. Du er dog nødt til at fortælle den hvilket connection objekt den skal bruge. Det er grund til at vi i mellem parenteserne i begge koder har skrevet dette:

(sql, con);

Vores DataAdapter objekt vil derefter vide hvilke poster der skal hentes (sql) og hvor de kommer fra (con).

Her ser du det du skal:

```
System.Data.SqlClient.SqlConnection con;  
System.Data.SqlClient.SqlDataAdapter da;  
DataSet ds1;  
  
private void Form1_Load(object sender, EventArgs e)  
{  
    con = new System.Data.SqlClient.SqlConnection();  
    ds1 = new DataSet();  
  
    con.ConnectionString = "Data  
Source=.\SQLEXPRESS;AttachDbFilename=C:\\Databases\\Medarbejder.mdf;Integrated  
Security=True;Connect Timeout=30;User Instance=True";  
  
    con.Open();  
  
    string sql = "SELECT * From tblMedarbejder";  
    da = new System.Data.SqlClient.SqlDataAdapter(sql, con);  
  
    con.Close();  
}
```

Kør dit program og se om det virker. Den eneste måde vi kan se om det virker er hvis der dukker fejlmeddelelser op på skærmen! Eller hvis programmeren crasher!

Når vi skal udfylde datasættet med poster fra databasen bruger du din DataAdapter, og kommandoen Fill. Det er det samme for både Access og SQL Server Express:

```
da.Fill( ds1, "medarbejder" );
```

Det der sker, er at Fill udfylder et Dataset, der hedder ds1. Efter kommaet kan du skrive et navn, der identificere den bestemte Fill. Vi kalder vores for "medarbejder".

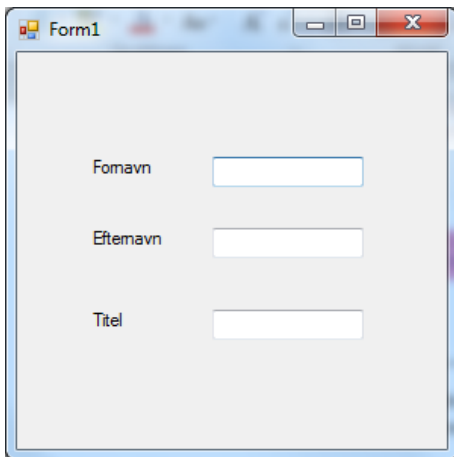
Når Fill kommandoen er udført, vil posterne fra SQL kommandoen blive gemt i dit Dataset. Det er bare et gitterværk med rækker og kolonner.

Tilføj linjen til din kode. Placer den lige før linen med con.Close().

Vi kan nu vise data fra vores database i en formular.

Vis data fra et Dataset i C#

Tilføj tre tekstbokse og tre labels til din formular, så det ser ud på følgende måde:



Når formularen hentes vil vi gerne have vist den første post fra vores Dataset i tekstboksene.

Vi gør det med en method. Lige efter koden der henter din form ind (load), tilføjer vi en ny method og kalder den **NavigateRecords**. Den skal returnere en værdi, så du kan lave den som en **void** method:

```
private void Form1_Load(object sender, EventArgs e)
{
    con = new System.Data.SqlClient.SqlConnection();
    dsl = new DataSet();

    con.ConnectionString = "Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\\Databaser\\Medarbejder.mdf;Integrated
Security=True;Connect Timeout=30;User Instance=True";

    string sql = "SELECT * From tblmedarbejder";
    da = new System.Data.SqlClient.SqlDataAdapter(sql, con);

    con.Open();

    da.Fill(dsl, "medarbejder");
```

```

        con.Close();
        con.Dispose();
    }
    private void NavigateRecords()
    {
    }
}

```

At få data ud fra et Dataset kan være en besværlig opgave; hovedsageligt fordi der er så mange betingelser og methods man kan bruge. Den nemmeste er først at oprette en ny variabel vi kalder **DataRow**:

DataRow dRow;

Den vil referere til en række i vores Dataset:

DataRow dRow = ds1.Tables["medarbejder"].Rows[0];

Efter lighedstegnet har vi dette udtryk:

ds1.Tables["Workers"].Rows[0];

Du skriver først navnet på dit datasæt, som er **ds1** hos os. Efter punktum vælges **Tables** fra IntelliSense listen. **Tables** er en samling som gemmer en liste med alle tilgængelige tabeller (en tabel er bare et gitterværk som vi nævnte). Når vi skal fortælle C# hvilken tabel vi ønsker, skriver vi navnet på tabellen mellem de kantede parenteser samt et par anførselstegn. Efter endnu et punktum vælges Rows fra IntelliSense listen. I de kantede parenteser angiver du hvilken række fra dit Dataset du ønsker. Row nul [0] er den første række i tabellen.

Tilføj følgende til din kode, så din navigateRecords ser sådan ud:

```

private void NavigateRecords()
{
    DataRow dRow = ds1.Tables["medarbejder"].Rows[0];
}

```

Det er dog ikke det eneste vi skal. Vi har kun henvist til en række i vores Dataset. Vi er også nød til at specificere en kolonne.

For at få en kolonne i rækken skriver vi denne kode:

dRow.ItemArray.GetValue(1).ToString()

Vi er startet med vores Row objekt, som vi kaldte dRow. Efter et punktum vælger du ItemArray fra Intellisense listen. Det er en Array med alle elementerne (kolonner) i din række. Vi har fire kolonner i vores database: medarbejder_ID, for_Navn, efter_Navn og job_Titel. Et ItemArray starter med nul, så medarbejder_ID vil være element 0, for_Navn vil være 1, efter_Navn vil være 2 og job_Titel vil være element 3.

Efter endnu et punktum vælger dy GetValue fra listen. Som navnet antyder, vil det hente værdien fra din kolonne. Imellem parenteserne skriver du det nummeret på det element du vil hente fra arrayet.

GetValue(1) vil referere til kolonnen for_Navn i dit dataset.. til slut skal det konverteres til en string med ToString(). Når det er konverteret til en string kan det puttes direkte ind i en tekstboks.

Hvis vi samler det hele kan du tilføje følgende kode til din method NavigateRecords:

```
private void NavigateRecords()
{
    DataRow dRow = dsl.Tables["medarbejder"].Rows[0];
    forNavn.Text = dRow.ItemArray.GetValue(1).ToString();
    efterNavn.Text = dRow.ItemArray.GetValue(2).ToString();
    jobTitel.Text = dRow.ItemArray.GetValue(3).ToString();
}
```

Hvis du foretrækker det kan du putte al sammen på den samme linje. Den bliver dog meget lang. Her er den:

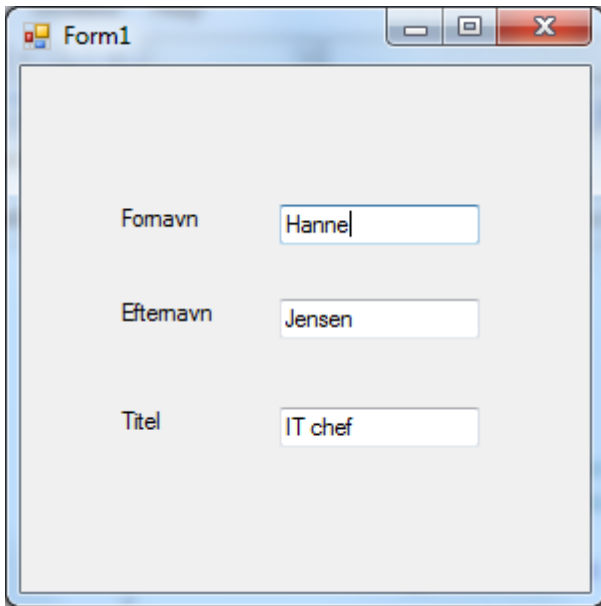
forNavn.Text = ds1.Tables["medarbejder"].Rows[0].ItemArray.GetValue(1).ToString();

Nu er du næsten klar til at afprøve det. Det sidste vi skal gøre er at oprette et kald til din method NavigateRecords. Anbring dette kald lige før linjen med con.Close, som vist nedenfor:

```
con.Open();

da.Fill(dsl, "medarbejder");
NavigateRecords();
con.Close();
con.Dispose();
}
private void NavigateRecords()
{
    DataRow dRow = dsl.Tables["medarbejder"].Rows[0];
    forNavn.Text = dRow.ItemArray.GetValue(1).ToString();
    efterNavn.Text = dRow.ItemArray.GetValue(2).ToString();
    jobTitel.Text = dRow.ItemArray.GetValue(3).ToString();
}
```

Nu kan du teste det. Kør dit program og din formular viser nu den første post i din database. Den ser ud på følgende måde:



Nu da vi har vist en enkelt post kan vi tilføje knapper så vi kan navigere frem og tilbage mellem vores poster i vores database.

Database navigations knapper

Vi skal allerførst give brugerne mulighed for at bevæge sig frem og tilbage i posterne i databasen. Det kan gøres med programmerings logik og manipulering med **Row** værdien i vores Dataset.

For at få det til at virke skal vi have oprettet nogle få variabler. Vend tilbage til kodevinduet og tilføj følgende to variabler udenfor dit load evnet, lige før de tre du allerede har:

```
int MaxRows = 0;
```

```
int inc = 0;
```

Dit kodevindue ser nu sådan ud:

```
System.Data.SqlClient.SqlConnection con;  
System.Data.SqlClient.SqlDataAdapter da;  
DataSet dsl;  
  
int MaxRows = 0;  
int inc = 0;  
  
private void Form1_Load(object sender, EventArgs e)  
{
```

Variablen **MaxRows** vil opbevare antallet af rækker der er i dit Dataset. Det er så vi ikke kommer forbi den sidste post når vi klikker på knappen Næste. Hvis prøver at komme forbi den sidste post vil programmet crashe! (Vi tilføjer en knap lige om lidt.)

Variablen **inc** skal bruges til at holde styr på hvilken række vi befinder os i.

Når vi skal finde antallet af rækker i et Dataset kan vi bruge betingelsen **Count i Rows**. Tilføj denne kode til din form load event, lige neden under dit kald til `NavigateRecords()`:

```
MaxRows = ds1.Tables["medarbejder"].Rows.Count;
```

I stedet for at specificere en bestemt Row i mellem de kantede parenteser, taster vi denne gang et punktum, og vælg derefter selectCount fra IntelliSense listen. Du får da returneret antallet af rækker der er i et givet Dataset. Her kan du se hvordan koden skal være:

```
da.Fill(ds1, "medarbejder");

NavigateRecords();
MaxRows = ds1.Tables["medarbejder"].Rows.Count;

con.Close();
con.Dispose();
}
private void NavigateRecords()
{
```

Når formularen hentes vil MaxRows indeholde en optælling af hvor mange Rows (rækker) der er i dit Dataset der hedder **ds1**.

Til din method NavigateRecords skal vi foretage en mindre ændring. Lige nu har vi denne kode:

```
DataRow dRow = ds1.Tables["medarbejder"].Rows[0];
```

Den vil pege på **Row[0]** hele tiden. Vi kan bruge variabelen **inc** her. Det vi skal gøre er at øge denne værdi når der klikkes på knappen Næste post. Vi lægger 1 til **inc** hver gang.

Ændre din linje til dette:

```
DataRow dRow = ds1.Tables["medarbejder"].Rows[inc];
```

Den eneste ændringer er mellem de kantede parenteser i **Rows**.

Kør dit program og se om det virker. Du ser nu den første post i din tekstboks.

Stop dit program og vend tilbage til design visningen. Tilføj en knap til din formular. Skift Text betingelsen til Næste Post. Skift også Name betingelsen til btnNext. Din formular ser nu sådan ud:

Dobbeltklik på din knap for at se kodevinduet. I koden er vi nødt til at undersøge hvad der er i MaxRows og være sikker på at vi ikke kommer forbi dette. Vi skal også optælle **inc** variabelen. Det er den variabel der vil få os videre til den næste post.

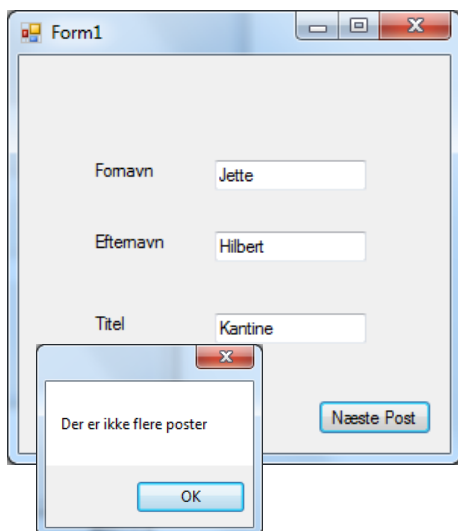
Tilføj følgende if sætning til din knap:

```
private void btnNext_Click(object sender, EventArgs e)
{
    if (inc != MaxRows - 1)
    {
        inc++;
        NavigateRecords();
    }
    else
    {
        MessageBox.Show("Der er ikke flere poster");
    }
}
```

Den første linje i if sætningen siger "hvis **inc** ikke er lig med **MaxRows** minus 1". Hvis den ikke er det så øger vi værdien af variabelen **inc** medog kalder **NavigateRecords**. Kan du se hvorfor vi skal sige **MaxRows -1**? Det er på grund af vores linje **Rows[inc]** i vores method **NavigateRecords**. Tælleren til vores Row starter med nul. Hvis vi derfor kun har 4 poster i vores database, vil optællingen være fra 0 til 3. **MaxRows** vil dog være 4. Hvis vi ikke trækker 1 fra vil programmet crashe med en fejl: **IndexOutOfRangeException**.

Hvis **MaxRows** er nået vil vi vise en meddelelse til brugeren.

Kør dit program og test det. Du burde være i stand til at bevæge dig fremad gennem databasen. Her kan du se hvordan formularen ser ud når den sidste post er nået:



I næste afsnit skal vi se hvordan vi bevæger os baglæns i databasen.

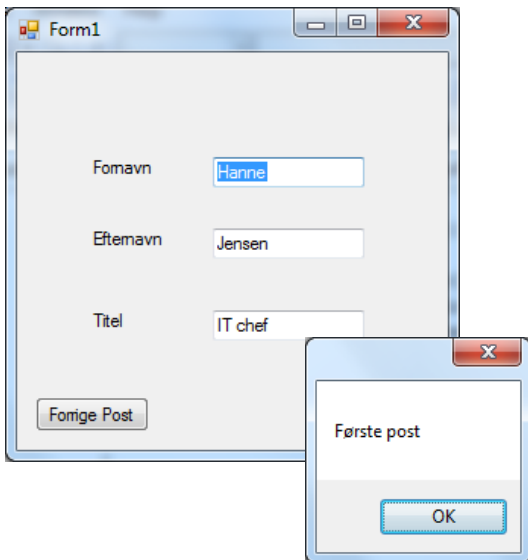
Bevæg dig baglæns i databasen

Vi kan bruge en lignende kode når vi skal bevæge os baglæns i databasen. Tilføj en ny knap til din formular. Skift Text betingelsen til **Forrige Post**. Skift betingelsen Name til **btnPrevious**.

Dobbeltklik på din nye knap for at se kodevinduet. Tilføj følgende kode:

```
private void btnPrevious_Click(object sender, EventArgs e)
{
    if (inc > 0)
    {
        inc--;
        NavigateRecords();
    }
    else
    {
        MessageBox.Show("Første post");
    }
}
```

Vores if sætning undersøger denne gang kun variabelen **inc**. Vi skal undersøge om den er større end nul. Hvis det gælder, kan vi trække 1 fra **inc**, og derefter kaldes vores method **NavigateRecords**. Husk på at når formularen hentes ind til **inc** være 0. Hvis vi derfor prøver at gå en post tilbage efter formularen startet vil programmet crashes. Det crasher fordi vi prøver at gå til Rows[-1].



Klik på begge knapper for at være sikker på du kan komme både frem og tilbage gennem din poster. Dit program crasher ikke!

Nu sjak vi se hvordan du kan springe til første og sidste post i databasen.

Hvordan springer du til første og sidste post i en database

Spring til sidste post i din database

Når du skal springe til den sidste post i databasen, har du kun brug for at sikre dig at variablen `inc` og `MaxRows` har den samme værdi.

Tilføj en ny knap til din formular. Skift Text betingelsen til Sidste Post, og betingelen Name til `btnLast`. Dobbeltklik på knappen og tilføj følgende kode:

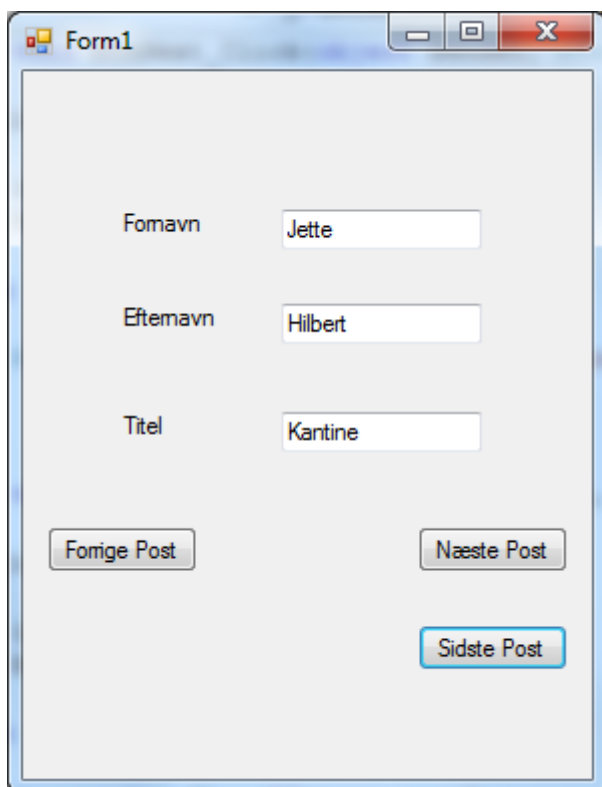
```
private void btnLast_Click(object sender, EventArgs e)
{
    if (inc != MaxRows - 1)
    {
        inc = MaxRows - 1;
        NavigateRecords();
    }
}
```

Vores if sætning undersøger igen om `inc` er forskellige fra `MaxRows` minus 1. Hvis det ikke er opfyldt sker der dette:

`inc = MaxRows - 1;`

`MaxRows` minus 1 giver 3 i vores fire poster store database. Da `Rows[inc]` går fra 0 til 3, er det nok til at springe til sidste post efter kaldet af `NavigateRecords`.

Her kan du se hvordan formularen ser ud når du tester den og klikker på din nye knap:



The screenshot shows a Windows application window titled "Form1". Inside the window, there is a form with three text input fields and three buttons. The first text box is labeled "Fornavn" and contains the text "Jette". The second text box is labeled "Efternavn" and contains the text "Hilbert". The third text box is labeled "Titel" and contains the text "Kantine". Below the text boxes, there are three buttons: "Forrige Post" (disabled), "Næste Post" (disabled), and "Sidste Post" (active/highlighted).

Spring til den første post i din database

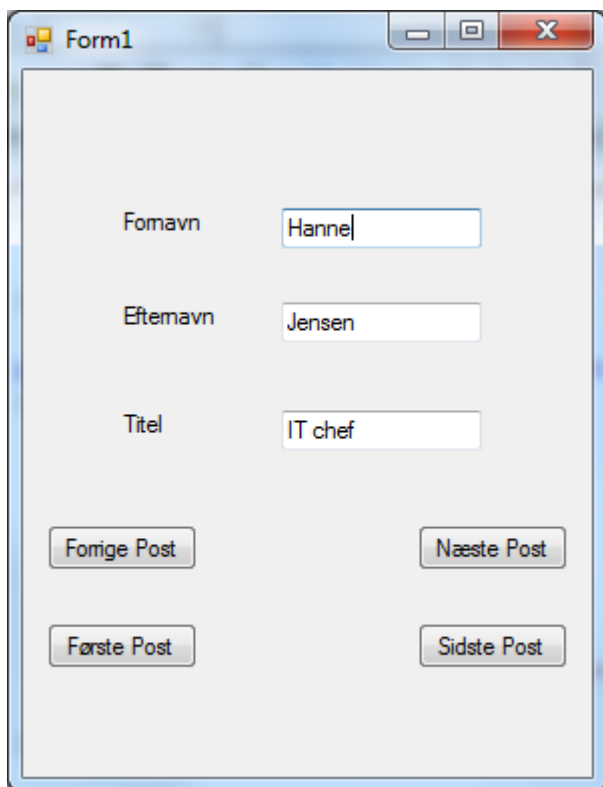
Når vi skal springe til første post i databasen skal vi sætte **inc** lig med nul.

Tilføj endnu en knap til din formular. Skift Text betingelsen til Første Post. Skift Name betingelsen til btnFirst. Dobbeltklik på den nye knap og tilføj følgende kode:

```
private void btnFirst_Click(object sender, EventArgs e)
{
    if (inc != 0)
    {
        inc = 0;
        NavigateRecords();
    }
}
```

Her undersøges om inc ikke allerede er nul. Hvis det ikke er tilfældet sætter vi variabelen **inc** til 0. Derefter kalder vi vores method NavigateRecords.

Når du tester din nye knap, ser din formular sådan ud:



Du er nu i stand til at bevæge dig igennem din poster i din database uden at programmet crasher. Nu skal vi gøre det muligt at tilføje en ny post til databasen. Der noget mere komplekst end navigationen, så du skal følge godt med!

Tilføj en ny post til Databasen

Når du tilføjer en ny post, skal du tilføje den til dit DataSet og den underliggende database. Lad os se hvordan. Aller først hvis du har tilføjet linjen `con.Dispose`, så kommenter den ud! Det kan ellers skabe problemer senere.

Tilføj to nye knapper til din formular. Angiv følgende betingelser for dem:

Name: btnAddNew

Text: Tilføj ny Post

Name: btnSave

Text: Gem

Knappen **Tilføj ny Post** vil rent faktisk ikke tilføje en ny post. Det eneste den gør, er at den nulstiller felterne i tekstboksene, så de er klar til at tilføje en ny post. Knappen **Gem** vil tilføje posten til dit DataSet og derefter til databasen.

Dobbelt klik på knappen Tilføj ny Post og tilføj følgende kode for at slette indholdet i tekstboksene:

```
private void btnAddNew_Click(object sender, EventArgs e)
{
    forNavn.Clear();
    efterNavn.Clear();
    jobTitel.Clear();
}
```

Det er alt hvad vi skal gøre her. Du kan teste det hvis du har lyst. Alt koden gør, er at den sletter indholdet i tekstboksene. Brugeren kan derefter tilføje en ny post.

Når der er indtastet en ny post i tekstboksene kan vi gemme den. Dobbeltklik på knappen Gem for at se koden.

Når du skal gemme er der to ting du skal gøre: gemme den i dit DataSet, og gemme den i den underliggende database. Du er nødt til at gøre det i den rækkefølge fordi dit dataset med dets kopi ikke er i forbindelse med databasen. Det at gemme i et DataSet er IKKE det samme som at gemme i databasen.

Når du skal tilføje en post til et DataSet skal du oprette en ny Row (række):

```
DataRow dRow = ds1.Tables["medarbejder"].NewRow();
```

Det opretter en New DataRow, der hedder dRow. Der er dog ingen data i den nye række. Når du skal tilføje data til den nye række gør du det på denne måde:

```
dRow[1] = textBox1.Text;
```

efter variabelen DataRow (hos os **dRow**) burger du et par kantede parenteser. Imellem parenteserne skriver du positionen i rækken (Row). Det er kolonne nummeret. `dRow[1]` referere til kolonnen for `_Navn`. Efter lighedstegnet kan du skrive hvad du har lyst til at tilføje til denne kolonne – i vores tilfælde teksten fra tekstboksen `forNavn`.

Til sidst tilføjer du kommandoen:

```
ds1.Tables["medarbejder"].Rows.Add( dRow );
```

Efter **Add** og i mellem parenteserne skriver du navnet på den Row (række) du vil tilføje, som i vores tilfælde er dRow. Den nye Row vil derefter blive tilføjet til vores Dataset.

Tilføj denne kode til din Gem knap:

```
private void btnSave_Click(object sender, EventArgs e)
{
    DataRow dRow = ds1.Tables["medarbejder"].NewRow();

    dRow[1] = forNavn.Text;
    dRow[2] = efterNavn.Text;
    dRow[3] = jobTitel.Text;

    ds1.Tables["medarbejder"].Rows.Add(dRow);

    MaxRows = MaxRows + 1;
    inc = MaxRows - 1;
}
```

Bemærk de sidste to linjer:

```
MaxRows = MaxRows + 1;
```

```
inc = MaxRows - 1;
```

Da vi har tilføjet en ny Row til vores Dataset skal vi også lægge 1 til variabelen MaxRows. Variablen **inc** sættes til den sidste post i vores Dataset.

Prøv det. Når du starter programmet, klikker du på knappen Tilføj ny Post for at nulstille tekstboksene. Indtast en ny post i de blanke tekstbokse og klik på knappen Gem. Klik på Forrige og Næste. Du ser nu at den nye post er dukket op.

(Selvfølgelig skal vi også lave kode så vi undersøger om der er trykket på Gem knappen før knappen Tilføj ny Post. Eller du kan måske sætte betingelsen for Enabled til false for knappen Gem når formularen startes. Du kan derefter sætte Enabled til true i knappen Tilføj ny Post.)

Hvis du lukker dit program og starter det igen vil du opdage at den nye post er forsvundet. Den er forsvundet fordi vi ikke har tilføjet den til vores underliggende database. Vi har kun tilføjet posten til vores Dataset.

Når vi skal tilføje en ny post til databasen, skal du bruge din DataAdapter igen. Den har en method der hedder Update, der kan gøre arbejdet for dig. Det eneste du skal gøre er at fortælle hvilket Dataset, der opbevarer alle posterne og dets navn:

```
da.Update( ds1, "medarbejder" );
```

Koden referere til en DataAdapter der hedder da. I mellem parenteserne for Update methoden, har vi først vores Dataset (ds1) og så navnet på vores dataset (medarbejder).

Men da vores forbindelse (connection) til databasen er lukket skal vi bruge endnu et spøjst objekt – noget vi kalder en **CommandBuilder**.

Denne CommandBuilder vil genskabe forbindelsen til databasen. Det eneste du skal gøre er at overføre den til en DataAdapter. Koden du skal bruge for at lave et CommandBuilder objekt i Access ser sådan ud:

```
System.Data.OleDb.OleDbCommandBuilder cb;  
cb = new System.Data.OleDb.OleDbCommandBuilder( da );
```

Og til en SQL Server Express:

```
System.Data.SqlClient.SqlCommandBuilder cb;  
cb = new System.Data.SqlClient.SqlCommandBuilder( da );
```

I begge tilfælde opretter vi et CommandBuilder objekt der hedder **cb**. Mellem parenteserne i koden ovenfor har vi vores DataAdapter variabel, som er **da**.

Du skal ikke gøre noget med din CommandBuilder. Den ved hvad den skal gøre.

Her er kode du skal tilføje:

```
private void btnSave_Click(object sender, EventArgs e)  
{  
    System.Data.SqlClient.SqlCommandBuilder cb;  
    cb = new System.Data.SqlClient.SqlCommandBuilder(da);  
  
    DataRow dRow = ds1.Tables["medarbejder"].NewRow();  
  
    dRow[1] = forNavn.Text;  
    dRow[2] = efterNavn.Text;  
    dRow[3] = jobTitel.Text;  
  
    ds1.Tables["medarbejder"].Rows.Add(dRow);  
  
    MaxRows = MaxRows + 1;  
    inc = MaxRows - 1;  
  
    da.Update(ds1, "medarbejder");  
  
    MessageBox.Show("Post tilføjet");  
}
```

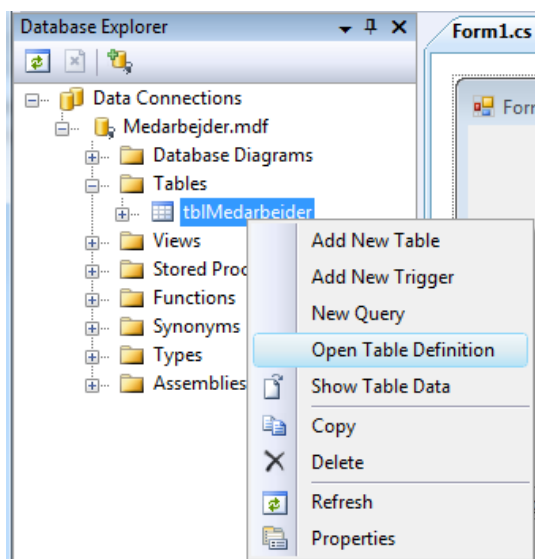
Du kan nu afprøve dit program. Du ser nu at den nye post er føjet til dit Dataset OG den underliggende database. Luk dit program og start det igen og se om den nye post er der.

Opdater og slet en post

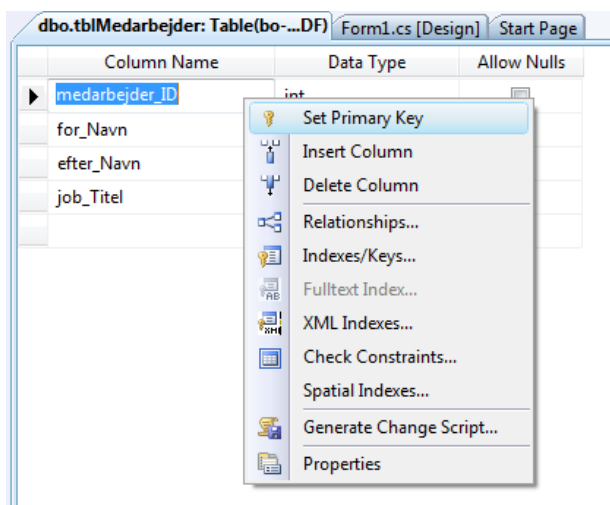
Ind imellem vil det også være praktisk at kunne opdatere en post i databasen. Det ligner meget den procedure vi gennemgik da vi tilføjede en post.

Før vi kan opdatere en post er vi nød til at definere en primærnøgle i database tabellen. Hvis det ikke gøres vil C# give en fejlmeddelelse. (Du kan springe dette over hvis du allerede har defineret en primærnøgle.) En

primærnøgle er en kolonne med unikke data, en der ikke har dubletter. I vores tilfælde er det kolonnen medarbejder_ID. Åben din tabel i Database Explorer. Udvid tabel delen og højre klik på din tabel:



I menuen vælger du **Open Table Definition**. Du ser nu kolonner og datatyper i din tabel. Højre klik på din ID kolonne og vælg "Set Primary Key".



Gem dit arbejde som normalt. Vi kan nu tilføje en Opdater mulighed.

Opdatere en post

Undersøg følgende kode:

```
private void bntUpdate_Click(object sender, EventArgs e)
{
    System.Data.SqlClient.SqlCommandBuilder cb;
    cb = new System.Data.SqlClient.SqlCommandBuilder(da);

    System.Data.DataRow dRow2 = ds1.Tables["medarbejde"].Rows[inc];

    dRow2[1] = forNavn.Text;
    dRow2[2] = efterNavn.Text;
```

```

        dRow2[3] = jobTitel.Text;

        da.Update(ds1, "medarbejder");

        MessageBox.Show("Data opdateret");
    }

```

Det er meget lig det der sker når vi tilføjer en ny række. Det eneste vi ikke gør er at tilføje en ny Row (række). Efter vi har oprettet en ny række vi kalder dRow2, sætter vi den til den øjeblikkelige række:

```
= ds1.Tables["medarbejder"].Rows[inc];
```

Hvad der end er i tekstboksen vil blive overført til dRow2[1], dRow2[2] og dRow2[3]. Det er vores kolonner i rækken. Derefter opdaterer vi databasen:

```
da.Update( ds1, "medarbejder" );
```

Før du prøver, ud kommenter din lind med **con.Dispose()**, hvis du har tilføje den. Ellers vil du få en Connection String fejl.

Når du kører din formular, så ret en af dine poster. Luk din formular og åben den igen. Se efter om dine rettelser stadig er der.

Dynamiske SQL fejl

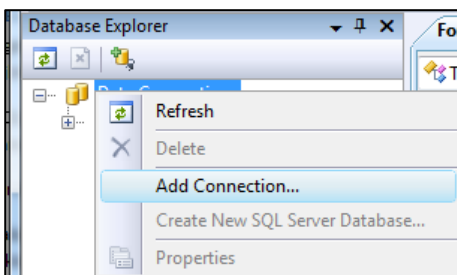
Hvis du bruger en SQL Server Express database, og har angivet en primærnøgle, kan du stadig løbe ind i følgende fejlmeddelelser:

"Dynamic SQL generation for the UpdateCommand is not supported against a SelectCommand that does not return any key column information."

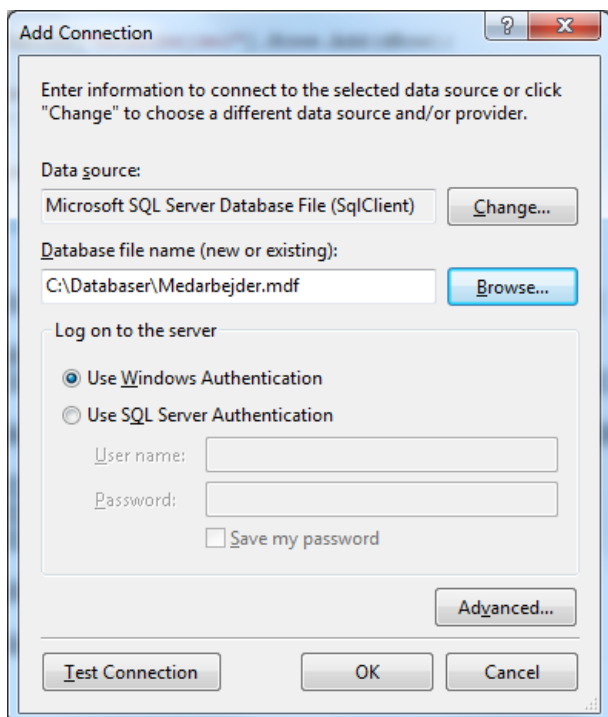
Hvis det sker så prøv følgende:

- Slet din database fra Database Explorer ved at højre klikke på database navnet. I menuen vælger du Delete (Du sletter ikke selve databasen, du fjerner den bare fra projektet.)
- Gør det samme hvis der er en database i Solution Explorer
- Slet eventuelle XSD elementer ligeså

I Database Explorer vinduet højre klikker du og vælger "Add Connection" i menuen:



I dialogboksen der dukker op vælger du **Browse** for at lede efter din database:



Klik på knappen **Test Connection** for at være sikker på at der er adgang til database. Klik derefter på OK. Det vil åbne den originale database. Nu kan du angive den primærnøgle som vist ovenfor. Gem dit program og test det.

Slet en post

Når du skal slette en post i databasen bruger du en method, der hedder **Delete**:

```
ds1.Tables["medarbejder"].Rows[inc].Delete();
```

Det er nok til at slette en hel række (**Rows[inc]**). Den slettes dog kun i Datasettet. Her er koden der skal til for at slette en post i databasen:

```
private void bntDelete_Click(object sender, EventArgs e)
{
    System.Data.SqlClient.SqlCommandBuilder cb;
    cb = new System.Data.SqlClient.SqlCommandBuilder(da);

    ds1.Tables["medarbejder"].Rows[inc].Delete();
    MaxRows--;
    inc = 0;
    NavigateRecords();

    da.Update(ds1, "medarbejder");

    MessageBox.Show("Posten er slettet");
}
```

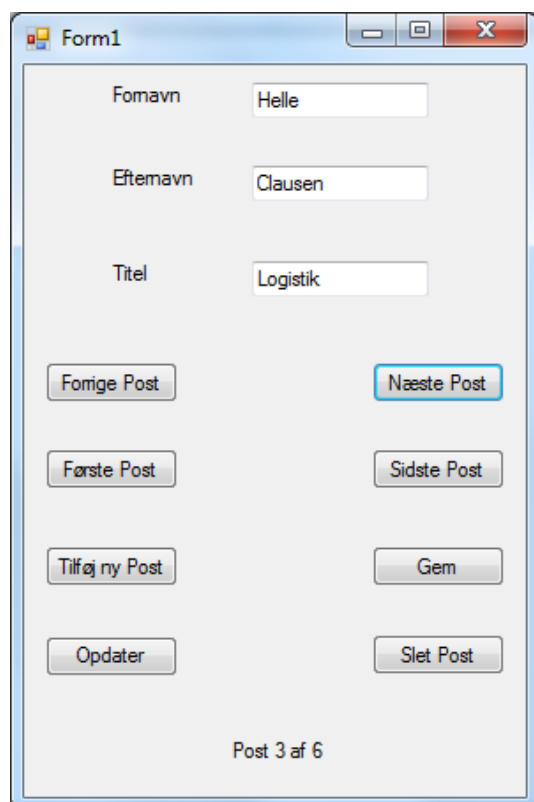
Bemærk at vi trækker 1 fra MaxRows (**MaxRows--**), og sætter **inc** til 0.

Prøv det selv. Tilføj en ny post til din database. Prøv derefter at slette den. Du kan løbe ind i fejlmeddelelse "concurrency violations". Hvis du lukker programmet og starter det igen, er du i stand til at slette din post uden fejl.

Fejlen concurrency violations kan opstå i mange situationer, men det er typisk i fordi dit dataset og databasen ikke er synkroniseret. Den nemme måde at løse dette problemer er at angive en Boolesk værdi når en post tilføjes. Hvis du prøver at slette den undersøg da om værdien er true. Hvis den er det så fortæl din bruger at de ikke kan slette den nye post.

Øvelse

Se nærmere på denne version af vores formular:



The screenshot shows a Windows application window titled "Form1". It contains three text input fields: "Fornavn" with the value "Helle", "Efternavn" with the value "Clausen", and "Titel" with the value "Logistik". Below the input fields are two columns of buttons. The left column contains "Forrige Post", "Første Post", "Tilføj ny Post", and "Opdater". The right column contains "Næste Post", "Sidste Post", "Gem", and "Slet Post". At the bottom center of the form, it displays "Post 3 af 6".

Hvis du ser i bunden vil du se en label der siger "Post 2 ud af 4". Implementer det i din eget program. Hvis du opretter en Method kan du kalde den når din formular hentes ind og igen fra NavigateRecords.

Søg efter poster i databasen

En meget brugbar feature i din formular kunne være en søg knap. Når der klikkes på en Søg knap vises den post brugeren søgte efter. Eller også vises der en "Ikke fundet" besked hvis der ikke findes poster med søgekriteriet.

Tilføj en ny knap til din formular. Sæt Text betingelsen til "Søg" og name betingelsen til **btnFind**. Dobbeltklik på knappen for at se koden.

Der er mange måder man kan implementere en søgning på. Den metode vi vil bruge er at udvælge en række fra datasettet. Vi vil lade brugeren søge efter et efternavn.

Tilføj følgende tre linjer med kode til din **btnFind**

```
string searchFor = "Hansen";  
int results = 0;  
DataRow[] returnedRows;
```

Den første variabel definerer en string der hedder **searchFor**. Det er den post vi vil finde. Vi har her håndskrevet værdien og bare tilføjet et efternavn fra databasen. Vi skal dog have værdien fra en tekstboks i formularen.

Den anden variabel, **results**, skal bruges til at fortælle os om der var fundet nogle resultater.

Den tredje linje er et DataRow array, som vi kalder **returnedRows**. Vi benytter et array, da vi kan få mere end en post i vores søgeresultat. Hver post vil blive opbevaret på en position i arrayet.

Når vi skal have en bestemt post i dit Dataset, kan du bruge metode Select. Her er koden:

```
returnedRows = ds1.Tables["medarbejder"].Select("efterNavn=" + searchFor + "");
```

Du starter med dit Dataset, som hos os er ds1. derefter skal du bruge navnet på en tabel i dit dataset. Vi vil søge i "medarbejder" tabellen. Efter et punktum har vi vores Select method:

```
Select("last_Name=' " + searchFor + " ' ");
```

Det ser en smule rodet ud med alle de anførselstegn. Men vi har et ydre par:

```
Select(" ");
```

Inden i de to anførselsteg har vi:

```
efterNavn=
```

Du skal skrive navnet på en kolonne i dit dataset. Vi bruger kolonnen efterNavn. Men vi kunne også have brugt kolonnen forNavn:

```
forNavn=
```

Kolonnenavnene er de samme som dem vi bruger i databasetabellen. Bemærk lighedstegnet. Select method giver dig mulighed for at benytte SQL. Hvis du ikke ønsker en eksakt søgning, for eksempel, kan du bruge LIKE i stedet for =.

```
Select("last_Name Like 'Hansen' ")
```

Bemærk de enkelt anførselsteg du burger omkring teksten du søger efter. Da vores søgning bruger en variabel bruger vi plus symboler til sammensætningen. Det er derfor det er noget rodet!

Her er koden du skal bruge:

```
private void btnFind_Click(object sender, EventArgs e)  
{  
    string searchFor = "Hansen";
```

```

    int results = 0;
    DataRow[] returnedRows;

    returnedRows = ds1.Tables["medarbejder"].Select("efterNavn='" + searchFor + "'");
}

```

Hvis der findes en række vil den blive opbevaret i arrayet returnedRows. Hvis du vil have antallet af rækker der er fundet kan du bruge denne kode:

results = returnedRows.Length;

Her bruger vi **Length** betingelsen på vores **returnedRows** array. Længden angiver hvor mange elementer der er i arrayet. Hvis det er større end nul betyder det at der er fundet et match. Vi kan bruge en IF sætning til at undersøge dette:

```

if (results > 0)
{
//Poster fundet
}
else
{
MessageBox.Show("Der er ikke fundet noget!");
}

```

Hvis der er fundet en post skal vi hente værdierne i kolonnerne. Vi kan oprette en ny række til dette:

```

if (results > 0)
{
DataRow dr1;

dr1 = returnedRows[0];
}

```

vi opretter nu en **DataRow** vi kalder **dr1**. Vi vil have gemt den første række der er fundet her. Den første række er **returnedRows[0]**;

Hvis vi samler det, er her koden til søgningen:

```

private void btnFind_Click(object sender, EventArgs e)
{
    string searchFor = "Hansen";
    int results = 0;
    DataRow[] returnedRows;

    returnedRows = ds1.Tables["medarbejder"].Select("efter_Navn='" + searchFor + "'");

    results = returnedRows.Length;

    if (results > 0)

```

```

    {
        DataRow dr1;

        dr1 = returnedRows[0];

        MessageBox.Show(dr1["job_Titel"].ToString());
    }
    else
    {
        MessageBox.Show("Der er ikke fundet resultater");
    }
}

```

Bemærk linjen der viser jobtitlen i meddelelsesboksen:

MessageBox.Show(dr1["jobTitel"].ToString());

Da dr1 nu er en DataRow kan du få adgang til den ved enten at bruge kolonnenavnene eller index nummeret. Disse linjer giver de samme værdier:

dr1["jobTitel"]
dr1["forNavn"]
dr1["efterNavn"]

dr1[3]
dr1[1]
dr1[2]

Det er op til dig hvad du vil bruge.

Prøv dit program. Klik på din Søg knap og jobtitel på personen med navnet Hansen vises i meddelelsesboksen.

Luk dit program. Skift navnet på personen, der søges efter og prøv igen.

Øvelse

Tilføj en tekstboks til din formular. Brug navnet på personen fra tekstboksen, i stedet for at skrive navnet i kode, som vi gør lige nu.