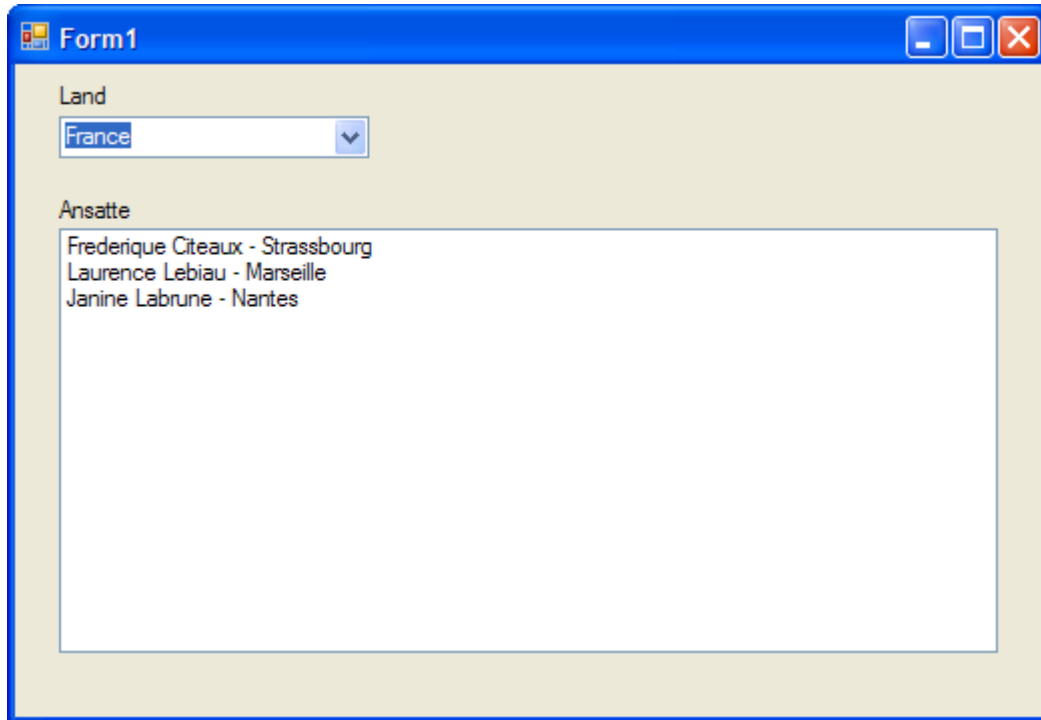


## Dag 06 .NET and C# Exercises ADO.NET

All the following exercises uses the NorthWind database.

### Exercise 1

Make a WPF Application with layout as below.



The screenshot shows a WPF application window titled "Form1". Inside the window, there is a label "Land" above a ComboBox control. The ComboBox currently displays "France". Below this, there is a label "Ansatte" above a ListBox control. The ListBox contains three items, each representing a customer: "Frederique Citeaux - Strassbourg", "Laurence Lebiau - Marseille", and "Janine Labrune - Nantes".

The ComboBox should be filled by values from the Country-column in the Customer-table: Use a SqlDataReader.

The easiest solution by now is to put all country names into a List<string>, and set the ComboBox's ItemsSource to the list.

When the user selects a country ("land") in the ComboBox'en, all customers from the selected country should be shown with ContactName and City in the control below (ListBox or TextBlock). On SelectionChanged in the ComboBox you must execute a new SQL-command and read the resultset using a SqlDataReader.

### Exercise 2

Make a ConsoleApplication which calculates the average freight price in the Orders-table in two different ways:

- Let the select-statement calculate the average, such that you read the scalar from the command.
- Use the SQL-statement

```
SELECT freight FROM orders
```

and calculate the average in the C#-code by reading all rows in the resultset.

**Exercise 3, Stored procedure returning a result table**

Make a WPFApplication executing the Stored Procedure whichs returns a table with two columns.

Put a TextBox, a Button and a ListBox in the GUI.

The stored procedure has an inputparameter, @CustomerID. Possible values of this is BERGS, EASTC or QUICK.

The input parameter can be typed in the TextBox. When the button is clicked, the application should call the stored procedure, custorderhist.

The procedure returns a table, which is read by a SqlDataReader. The output table has two columns ProductName and Total. The resulting table must be shown in the listbox. This can be done in more ways.

- a) The easiest is to put each row in an item of a List<string> and set the listbox's ItemsSource to the list.
- b) A more flexible way is to put each row in an item of a List of objects with a string and an int. Also in this case the listbox's ItemsSource is set to the list. What is displayed is the listbox can either be defined by a ToString methods or a property.

If the object are created as anonymous object like

```
new { Product=productName, Total=total };
```

You need to create a third property to display.

Next day we shall see how a table can be shown in a WPF GUI.

**Exercise 4, Multiuser system with optimistic locking**

Create a new database persons.

Use the persondata\_eng.sql file to create a table, Persondata, with attributtes id, name, address, rating, and a timestamp.  
id is an Identity.

*If you don't know TimeStamp in MSSQL, read the explanation next page.*

A few testdata are inserted by the script.

Make a WPFApplication with the following controls:

- A TextBox for Id to load the person with the specified Id
- TextBoxes for a) name, b) address og c) rating
- A label for the value of TimeStamp
- A Load-Button to load and display the person with the specified Id
- A Save-Button for saving the TextBox values of Name, Address and Rating in the person record

Multi user system

Possible Ids: 1, 2, 3, 4, 5, 6

Id: 3

Name: Christina

Address: Hovedgaden 18

Rating: 10

Timestamp: 2003

Buttons: Load, Save

Create a class file with declaration of a Person class with Properties Id (int), Name (string), Address (string), Rating (int) and TimeStamp (int64).

**Please note:** In the database timestamp is a byte array with 8 bytes; when reading the TimeStamp column in the SqlDataReader a byte-array is returned. To convert a byte-array to a Int64, you can convert it yourself or use the BitConverter class, which has a ToInt64 method.

Create a service class and put *all* access to the database in the service class with these Methods:

```
public Person LoadPerson(int id)
```

which load the person with the primary key id.

```
SavePerson(Person person)
```

to update a person. (The application is not supposed to create new persons, but update name, address and rating for the person where id is equal to person.Id)

In SavePerson you must start a transaction. Update takes place like this:

- a) Reread the record in DB going to be updated.
- b) Compare the new Timestamp with the Timestamp from first read of record.
- c1) If they are different: Inform the user and shown him the new values.
- c2) If the two timestamps are identical the new data is written to the database.
- d) Stop the transaction.

Test your application by starting two instances of the application simultaneously to see if the optimistic locking works: You can start a new instance of an application by right-clicking the project in Solution Explorer in Visual Studio and select **Debug → Start new instance**.

(You can also find the exe-file and double click it twice in File Explorer.)

## TimeStamp

As opposed to MySQL a TimeStamp in MSSQL is NOT a date- and time value, but a long value.

Suppose the person table now is

id	name	address	rating	ts (timestamp)
1	Lone M J	Obovej 40	6	0x000000000000055F1
2	Birgitte	Algade 1	12	0x00000000000003E82
3	Christina	Hovedgaden 18	12	0x00000000000002712
4	Dorte	Symfonialle 50	11	0x000000000000036B1
5	Susanne	Vintenvej 33	5	0x000000000000007D5
6	Sabina	Aarhusvej 123	8	0x000000000000007D6
7	Arne	Orkesteralle 20	7	0x00000000000000FA2

If we insert a new record it will get the next value in ts:

```
insert into person (name, address, rating)
values ('Hanne', 'Jyllands Alle 45', 10)
```

id	name	address	rating	ts (timestamp)
1	Lone M J	Obovej 40	6	0x000000000000055F1
2	Birgitte	Algade 1	12	0x00000000000003E82
3	Christina	Hovedgaden 18	12	0x00000000000002712
4	Dorte	Symfonialle 50	11	0x000000000000036B1
5	Susanne	Vintenvej 33	5	0x000000000000007D5
6	Sabina	Aarhusvej 123	8	0x000000000000007D6
7	Arne	Orkesteralle 20	7	0x00000000000000FA2
8	Hanne	Jyllands Alle 45	10	0x00000000000005DC1

If we update the row with id=2 it will get the next value in ts:

```
update person set address='Randersvej 288', rating=8
where id=2
```

id	name	address	rating	ts (timestamp)
1	Lone M J	Obovej 40	6	0x000000000000055F1
2	Birgitte	Randersvej 288	8	0x00000000000005DC2
3	Christina	Hovedgaden 18	12	0x00000000000002712
4	Dorte	Symfonialle 50	11	0x000000000000036B1
5	Susanne	Vintenvej 33	5	0x000000000000007D5
6	Sabina	Aarhusvej 123	8	0x000000000000007D6
7	Arne	Orkesteralle 20	7	0x00000000000000FA2
8	Hanne	Jyllands Alle 45	10	0x00000000000005DC1

The nice thing: Every time a row is updated, the TimeStamp in the row will change. Then we can be sure that an update has been execute if the timestamp has got a new value, since we last read the timetamp.