

## Softwareontwikkeling 6IT – preteaching programma

### Leerstof – Databases met MySQL : Een C#-toepassing bouwen waarmee je een MySQL-database kan beheren.

Behandelt aspecten van leerplandoelstellingen : 19, 26, 27, 28

#### GEGEVEN

In de vorige oefening heb je het database-schema productenDB opgebouwd. In deze oefening ga je dat schema gebruiken om een C#-toepassing te bouwen waarmee je die database kan beheren.

Na deze oefening zal je in staat zijn om volgende zaken uit te voeren

- Je kan een verbinding met de database openen en sluiten
- Je kan uitleggen wat het effect is van het using()-statement in .NET
- Je kan uitleggen waarom men try...catch gebruikt
- Je kan vanuit C# een query uitvoeren op de database
- Je kan vanuit C# parameters gebruiken in jouw query's
- Je leert voor een aantal controls (datagridview, combobox) hoe je er databasegegevens in kan weergeven.

Je hoeft niet vanaf nul te starten. Er is een voorbeeldtoepassing gemaakt die alle elementen die je moet kennen demonstreert. Deze toepassing werkt met de ITProDB-database die in de instructiefilmpjes is opgebouwd. Je krijgt ook het schema van deze database ter beschikking als op zichzelf staand .sql-bestand.

#### GEVRAAGD

Maak een nieuw VS C# project aan voor een Windows Forms Toepassing. Noem dit project 'MySQL\_productenDB'. *Screenshots van de toepassing kan je aan het einde van dit document terugvinden. Het is ook aangewezen om de [online tutorial over de MySQL Connector/NET](#) bij de hand te houden.*

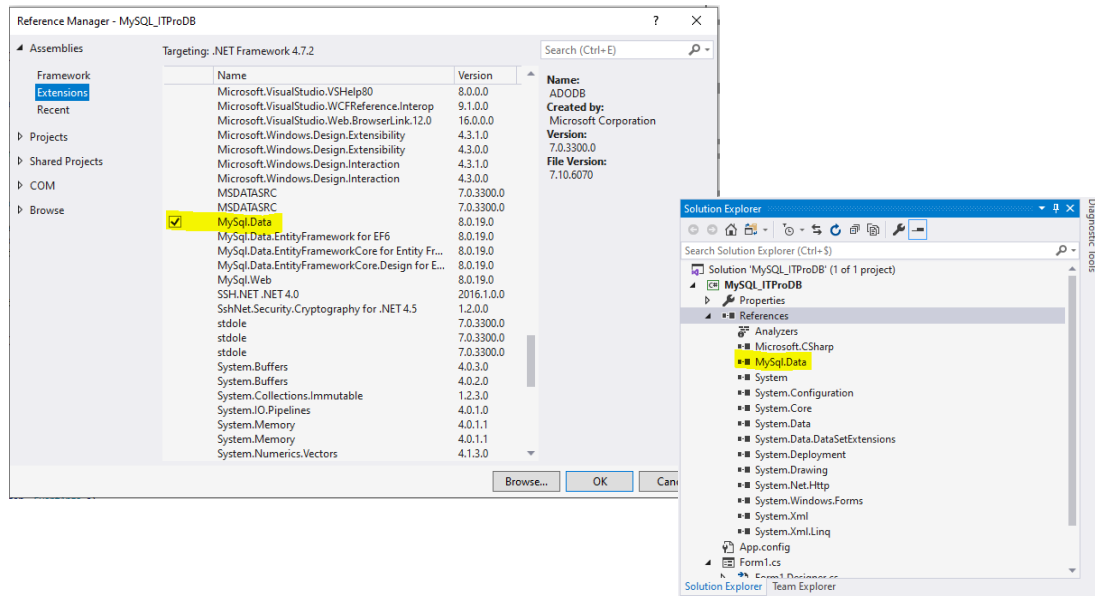
#### A. Openen en sluiten van een MySQL-verbinding

##### 1. De SQL-connector toevoegen

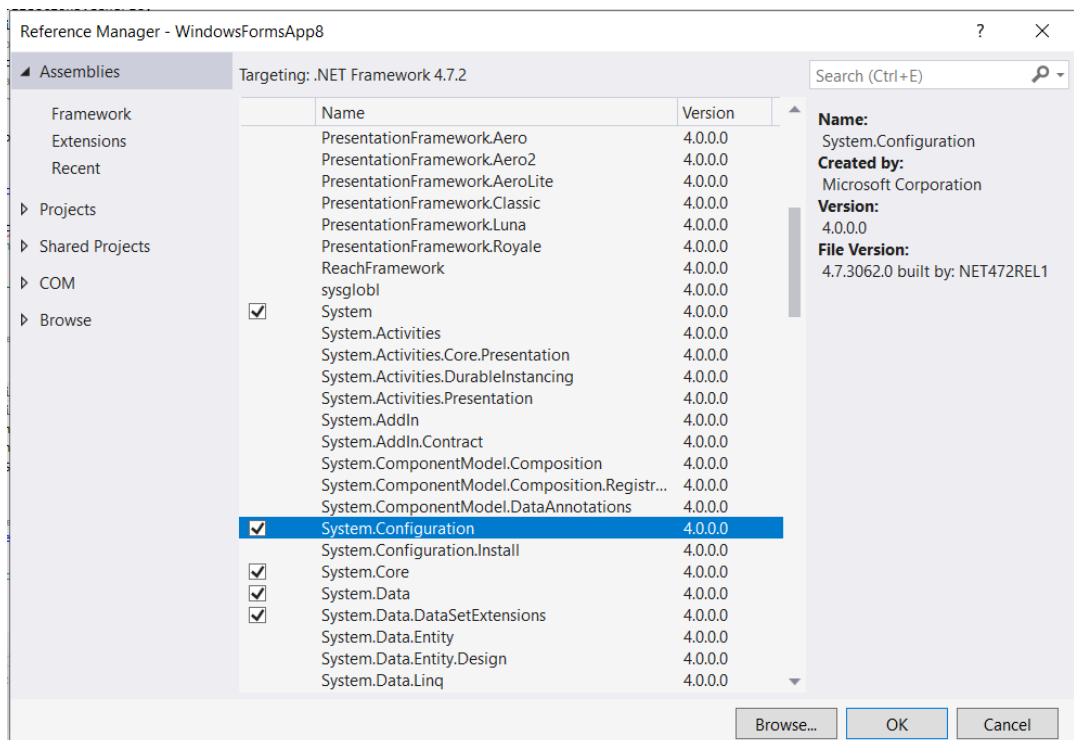
Om een MySQL-verbinding vanuit C# te kunnen maken heb je een MySQL-connector nodig. Deze connector kan je downloaden vanaf <https://dev.mysql.com/downloads/connector/net/>

Installeer de connector. Op jouw PC staat nu een API die je van functies voorziet om met de database te praten. Je moet nog een referentie naar die API toevoegen aan jouw toepassing. Bij de eerste

lessen rond C# hebben we gezien hoe je dit kan doen. Je zal een referentie naar de MySQL API aan je toepassing moeten toevoegen:



Voeg ook een referentie toe naar System.Configuration. Dit is een noodzakelijke voorwaarde om de opdracht succesvol te kunnen afronden.

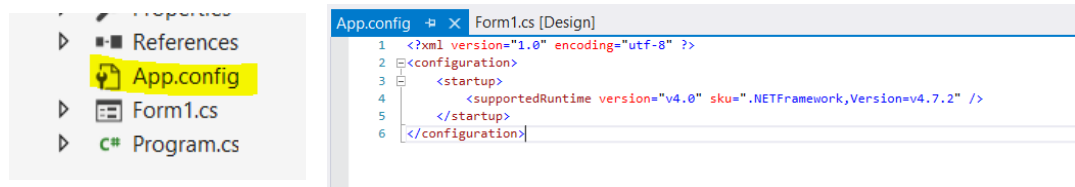


## 2. De SQL-connectiestring

Om verbing te leggen met de database heb je een 'connectiestring' nodig. Zoek op hoe een MySQL-connectiestring er uitziet. Stel de connectiestring voor de verbinding met de productenDB op.

## 3. Connectiestring toevoegen aan app.config

We moeten de connectiestring in onze toepassing gebruiken. Je zou deze connectiestring hardcoded kunnen toevoegen. Wanneer er gegevens (zoals het IP-adres) wijzigen dan moet je de code aanpassen, hercompileren, herverdelen... Dat is niet zo praktisch. We gaan het app.config bestand gebruiken dat bij het aanmaken van de Windows Forms toepassing is voorzien. Dat is een XML-bestand. Op de site van [w3schools](http://w3schools) wordt uitgelegd wat XML precies is.



Voeg de connectiestring voor jouw toepassing toe aan het app.config bestand. Zoek in de voorbeeld-toepassing hoe dit wordt gedaan. Voorlopig mag je ook nog het paswoord in plain tekst erbij zetten. Het is uiteraard de bedoeling dat in een uiteindelijke toepassing dit wachtwoord in run-time aan de gebruiker wordt gevraagd. Uit testoverwegingen laten we dit nu toe.

## 4. Openen van de verbinding met de MySql-database

Twee zaken zijn belangrijk:

- De connectiestring : Die moet uitgelezen worden uit het app.config bestand
- Een object van de klasse MySqlConnection : Dit object moet je in de code definiëren.

Zoek in de voorbeeldtoepassing op hoe de connectiestring uit het configuratiebestand wordt ingelezen. Let op: Het kan zijn dat er op de machine waarop je werkt al een SQL-server draait. De voorbeeldtoepassing leest alle connectiestrings uit en plaatst die in een dictionary en koppelt deze dictionary aan de databron van een combobox. Probeer dit in de code goed te volgen.

Schrijf in jouw project de code die *noodzakelijk* is om een MySQL-verbinding te openen. Je voorziet jouw form van een tabcontrol.

Op een tab met opschrift "verbinding testen" zet je volgende controls:

- Je voorziet een *combobox* waarin de SQL-verbingen zichtbaar worden die op jouw computer zijn voorzien.
- Je voorziet een knop `btnOpenMySQL` met het opschrift "Open MySQL verbinding". Wanneer je op deze knop drukt dan moet de verbinding met de database gemaakt worden.

Met behulp van een messagebox meld je aan de gebruiker of de verbinding succesvol is of niet.

Start de USBWebserver en test dit codestuk uit. Kan je verbinding leggen met de database?

5. In opdracht 4 heb je moeten uitzoeken wat de onderstaande method doet. Leg uit waarom er in deze method een try...catch wordt gebruikt?

```
1 reference
private ConnectionStringsCollection GetConnectionStrings()
{
    ConnectionStringsCollection settings = new ConnectionStringsCollection();

    try
    {
        settings = ConfigurationManager.ConnectionStrings;
        LogtekstToevoegen(tbLoggingText, "App.config uitgelezen. Connecties geladen...");
    }
    catch (ConfigurationErrorsException err)
    {
        MessageBox.Show(err.Message, "Configuratie", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    return settings;
}
```

6. Het sluiten van een MySQL-verbinding

Zet op de tab "verbinding testen" een knop `btnCloseMySQL` met het opschrift "Sluit MySQL verbinding". Ga is de voorbeeldcode na hoe de verbing wordt gesloten. Probeer uit de voorbeeldcode de essentie te halen. In die code zijn er immers al zaken toegevoegd die meer op hergebruik van code zijn gericht. Programmeer de code voor de knop sluit MySQL database. Gebruik een messagebox om aan de gebruiker te melden wanneer de verbinding succesvol is verbroken.

## 7. Gebruik van het using-statement

### a. Theoretische achtergrond

Wanneer je in jouw toepassing op de knop "open MySql verbinding" drukt dan blijft de verbinding open tot wanneer je op de knop "sluit MySql verbinding" drukt.

Het is beter om een verbinding enkel open te houden wanneer het zin heeft. Wanneer je in jouw toepassing de verbinding opent en gedurende uren niets doet dan kan de server ondertussen down zijn zonder dat je er erg in hebt. Het is beter om een verbinding te openen, te doen wat je moet doen, en ze dan direct terug te sluiten. Dit geeft geen noemenswaardige extra belasting voor de server.

Je zou de twee stukken code die je al hebt geschreven gewoon kunnen samenbrengen in één routine. Tussen het openen en het sluiten kan je dan de uitvoering van een query zetten. Er zit echter nog een addertje onder het gras:

Een object van de klasse `MySqlConnection` gebruikt *managed* en *unmanaged resources*. Wanneer een dergelijk object uit focus gaat dan zal de *garbage collector* de *managed resources* opruimen. De *unmanaged resources* worden niet opgeruimd. Lees de tekst over unmanaged resources op [docs.microsoft.com](https://docs.microsoft.com). Het niet opruimen van unmanaged resources zorgt ervoor dat er onnodig geheugen wordt gebruikt. Wanneer dit geheugen niet wordt vrijgegeven dan kan er op de duur een geheugenlek ontstaan.

We kunnen dit probleem aanpakken door gebruik te maken van het `using()` statement. Om een object van de klasse `MySqlConnection` te gebruiken kan je de onderstaande code gebruiken:

```
using (MySqlConnection mySqlConn = new MySqlConnection(mySqlConnStr))
{
    //open de verbinding

    //doe zinvolle dingen

    //sluit de verbinding
}
```

### b. Opdrachten

Voeg een nieuwe tab toe aan de tabcontrol. Geef deze tab de naam 'Uitlezen tabel orders'

- Voeg aan jouw form een knop toe. Noem deze knop btnConnectAndClose. Geef de knop het opschrift 'Uitlezen tabel'. Wanneer je op deze knop drukt dan voer je het bovenstaande using-statement uit. Zorg dat de USBWebserver draaiend is en test de code uit.
- Stop via de USBWebserver de MySQL server en test jouw code opnieuw uit. Blijft jouw toepassing draaien?
- Pas de code die wordt uitgevoerd als de knop btnConnectAndClose aan. Zet het using-blok in een try...catch blok. Voer de test met gestopte MySQL-server terug uit. Wat stel je vast?

```
try
{
    using (MySqlConnection mySqlConnection = new MySqlConnection(mySqlConnectionStr))
    {
        //open de verbinding
        ...

        //doe zinvolle dingen
        ...

        //sluit de verbinding
        ...
    }
}
catch (MySqlException err)
{
    //toon de fout. Je kan dit doen door err.ToString()
    //in een messagebox te tonen.
}
}
```

## 8. Een MySQL commando configureren.

Bestudeer in het voorbeeldprogramma grondig de method `laadVaardigheden()` in het voorbeeldprogramma. In die method wordt de tabel 'vaardigheden' integraal ingelezen in het programma.

- a. Duidt het stuk code aan dat overeenkomt met de code die je in opdracht 7.b hebt geschreven. Dit stuk code heb je al bestudeerd.

Opmerking : De code in het using-statement ziet er een beetje anders uit. Dat komt omdat in de voorbeeldcode het openen van een verbinding uitbesteed is aan de method `OpenMySQLverbinding(...)`. Die method levert een referentie af naar een object van de klasse `MySQLConnection`. Ook het catch-blok ziet er een beetje anders uit.

In jouw toepassing werk je verder op wat je in opdracht 7 hebt gedaan.

- b. Om een sql-query uit te voeren hebben we een object nodig van de klasse `MySqlCommand`. Een commando kent verschillende attributen. Drie belangrijke attributen zijn:

- De verbinding die het commando gebruik
- De querytekst van het commando
- Het type commando

In de method `laadVaardigheden()` is een object van de klasse `MySqlCommand` gedeclareerd. Wat is de naam van dit object?

Wat is de waarde van de hierboven vermelde drie belangrijke attributen van dit object?

### Programmeeropdracht

Maak in jouw toepassing een object aan van de klasse `MySqlCommand` . De querytekst van het commando haal je uit opdracht 16 van de oefening rond het maken van een databaseschema met `MySql Workbench`.

## 9. Een MySql commando uitvoeren (deel 1)

In opdracht 8 heb je een MySQL commando geconfigureerd. De bedoeling is dat alle orders uit de database productenDB worden uitgelezen. In deze opdracht ga je code voorzien om deze actie vanuit C# uit te voeren. Je moet terug de method `laadVaardigheden()` bestuderen.

### a. Theoretische achtergrond

Om een MySQL-commando uit te voeren moet je één van de execute-commando's gebruiken. Een commando dat informatie leest uit de database maakt gebruik van een *reader*. In de .NET-connector is dat een object van de klasse `MySqlDataReader`. Om de reader te activeren kan je gebruik maken van de method `ExecuteReader`.

In de voorbeeldcode – bij staat deze reader ook in een using-statement.

```
using (MySqlDataReader mySqlDr = mySqlCommand.ExecuteReader())
```

De reden daarvoor is al eerder in deze opdracht besproken.

Zodra de reader is gestart kan men data uitlezen in een while-lus:

```
while (mySqlDr.Read())
{
    //doe iets met de ingelezen data...
    //Het object mySqlDr kan je geïndexeerd uitlezen
    // mySqlDr[0], mySqlDr[1], mySqlDr[2]
    //In de voorbeeldcode is er expliciete cast naar
    //int en string voorzien (vb. (int)mySqlDir[0] )
}
```

In het voorbeeldprogramma wordt de data naar een Dictionary geschreven. De ingelezen data is via indexering [...] beschikbaar. Het is (nog) niet de bedoeling dat je in jouw oefening de data naar een dictionary schrijft.

### b. Programmeeropdracht

In opdracht 8 heb je een MySQL commando klaargemaakt. In deze opdracht ga je het commando uitvoeren. Zet in de tab 'Uitlezen tabel' een control van de klasse `TextBox`. Voer de reader uit. Toon de informatie in het tekstvenster. Elke lijn toont één record. De velden worden door tabs gescheiden (tip `'\t'`).



## 10. Een MySQL commando uitvoeren (deel 2)

In opdracht 9 heb je een commando uitgevoerd waarmee je informatie uit de database kan lezen met behulp van een reader. Er zijn ook commando's waarbij je geen informatie leest maar wel informatie wijzigt. In deze opdracht ga je code voorzien om dergelijke acties vanuit C# uit te voeren.

### a. Theoretische achtergrond

#### *SQL-parameters*

In opdracht 8 zijn al drie belangrijke attributen bod gekomen die van belang zijn bij voor een object van de klasse MySqlCommand. Een vierde belangrijk attribuut zijn de zogenaamde *parameters*.

Wanneer je een MySqlCommand-object configureerd dan geef je via het `commandText`-attribuut aan welke query je op de database wil uitvoeren.

Het `commandText`-attribuut is een string tekst. De sql-server interpreteert die tekst als query en zal hij die integraal uitvoeren. Dat kan zeer grote problemen geven.

Stel je eens het volgende fragment van een GUI-toepassing voor voor orderbeheer:



Wanneer er op controleer wordt gedrukt dan zal het order waarvan de gebruiker in de textbox het orderID heeft ingegeven via een select-query uit de database worden gehaald.

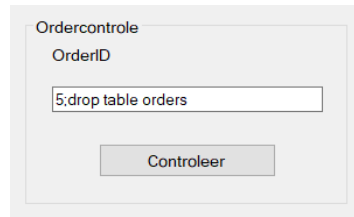
Het `CommandText`-attribuut van het MySqlCommand-object wordt dan als volgt opgebouwd:

```
mysqlComm.CommandText = "delete from orders where orderID  
= " + textBox1.Text + " ;" ;
```

In dit geval zal de MySQL-server de volgende query uitvoeren:

```
delete from orders where orderID = 5;
```

Als de gebruiker in de textbox het volgende ingeeft dan loopt het mis:



De server voert dan de volgende query uit:

```
delete from orders where orderID = 5; drop table orders;
```

Het gevolg hiervan is dat we onze orders-tabel hebben gewist...

We kunnen dus "kwaadaardige" query's uitvoeren op onze MySQL-server. Dat noemen we SQL-injectie. Wanneer een database op dergelijke wijze wordt bedreigt dan spreekt men van een "SQL injection attack". Je kan hier meer over lezen via [w3schools.com](http://w3schools.com). Dit probleem beperkt zich niet tot een C#-toepassing. Elke interface langs waar een gebruiker gegevens moet ingeven om data uit een database te halen kan risico's op injectie met zich meebrengen. Hoewel SQL-injectie één van de oudste soorten aanvallen is blijft het tot op vandaag een groot probleem. Elke programmeur moet weten wat SQL-injectie is. Je moet ook doordrongen zijn van het volgende: "DON'T TRUST USER INPUT!!!".

Gelukkig biedt de SQL-connector in .NET de mogelijkheid om dit probleem aan te pakken: SQL PARAMETERS

In plaats van de gebruikersinvoer gewoon toe te voegen aan de query ga je die opslaan in een SQL-parameter. De query *met* parameters wordt doorgegeven aan de SQL-server.

Wanneer de gebruiker een tekst invoert in de textbox dan gaan we die tekst als volgt toekennen aan een SQL-parameter:

```
mySqlCommand.Parameters.AddWithValue("@orderID", textBox1.Text);
```

De parameter heet "orderID". Zijn waarde komt via de gebruikersinvoer in textBox1.text.

Om die parameter te gebruiken in een sql-commando schrijven we dat commando als volgt:

```
mySqlCommand.CommandText = "DELETE FROM orders WHERE orderID = @orderID";
```

De @-prefix geeft aan dat we met een *"named parameter"* te maken hebben. In de CommandText moet je dan de naam van de parameter zetten samen. Er bestaan ook zogenaamde anonieme parameters:

```
cmd.CommandText = "UPDATE mijnTabel SET kolom1 = ?, kolom2 = ?;";  
cmd.Parameters.AddWithValue("naam1", waarde1);  
cmd.Parameters.AddWithValue("naam2", waarde2);
```

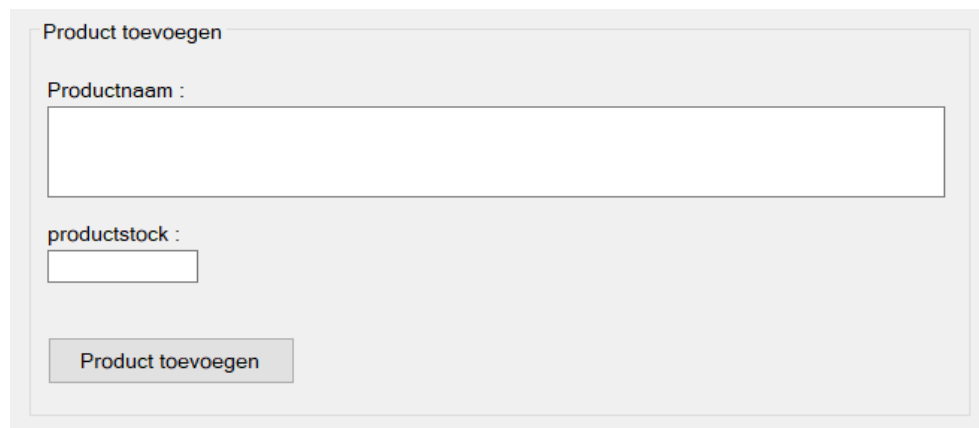
De '?' in de commandtext worden aangevuld met de parameters in de volgorde waarin ze zijn gedefinieerd. Anonieme parameters krijgen dus een volgnummer of index toegewezen.

Het delete-commando is een commando dat geen gegevens teruggeeft aan jouw toepassing. Er is dus geen reader nodig. Om een dergelijk commando uit te voeren gebruiken we de ExecuteNonQuery()-method. Deze method gebruik je ook bij create-, alter-, drop-, insert-, update-query's enz. Dus query's zonder return-waarde.

#### b. Programmeeropdrachten

In deze opdracht ga je producten toevoegen aan de tabel 'producten' van de database 'produductenDB'.

Voeg een nieuwe tab aan de tabcontrol toe. Geef deze tab het opschrift 'toevoegen product' en plaats er de volgende controls op:



The screenshot shows a Windows Forms application window titled "Product toevoegen". Inside the window, there are two text boxes. The first is labeled "Productnaam :" and is empty. The second is labeled "productstock :" and is also empty. Below these text boxes is a button labeled "Product toevoegen".

Wanneer je op de knop 'product toevoegen' klikt dan moet er op de MySQL-server een query uitgevoerd worden die aan de tabel producten een product toevoegd met de gegeven naam en productstock. Voor de beschikbaarheid van het product wordt de default waarde genomen.

Het toevoegen moet *SQL-injectie veilig* en *idiot-proof* zijn.

11. Voeg een nieuwe tab toe aan de tabcontrol. Geef deze tab het opschrift 'Beheer tabel producten'.

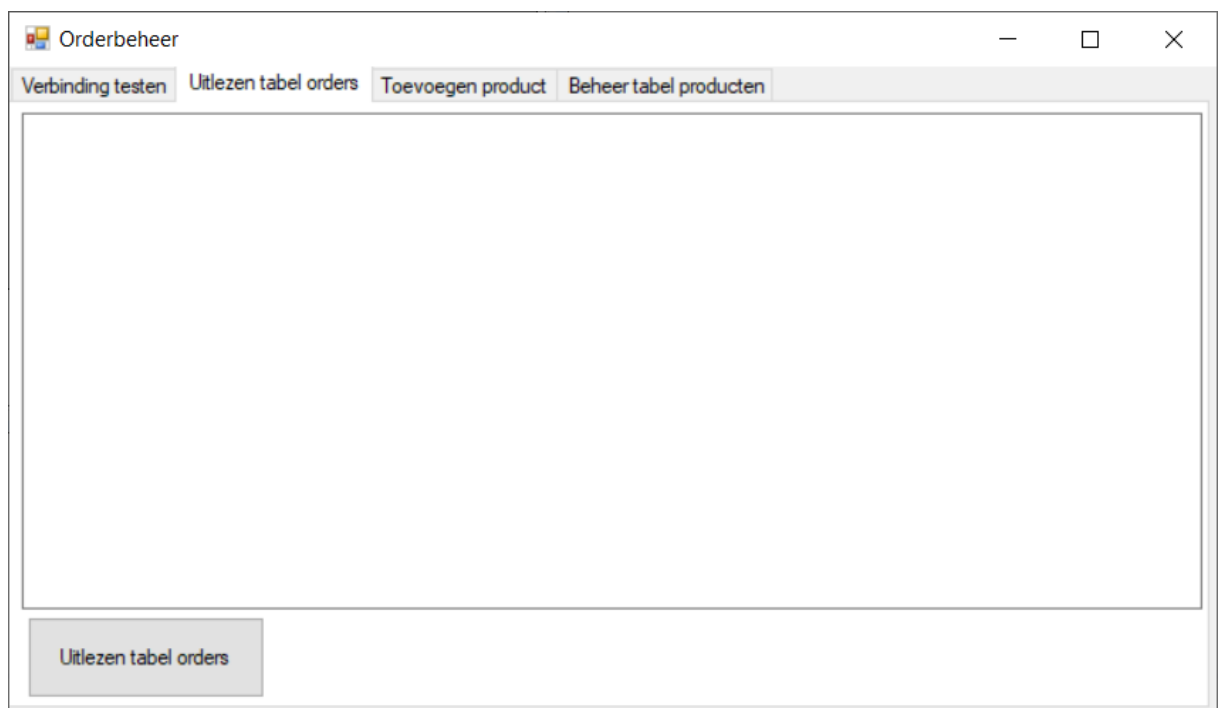
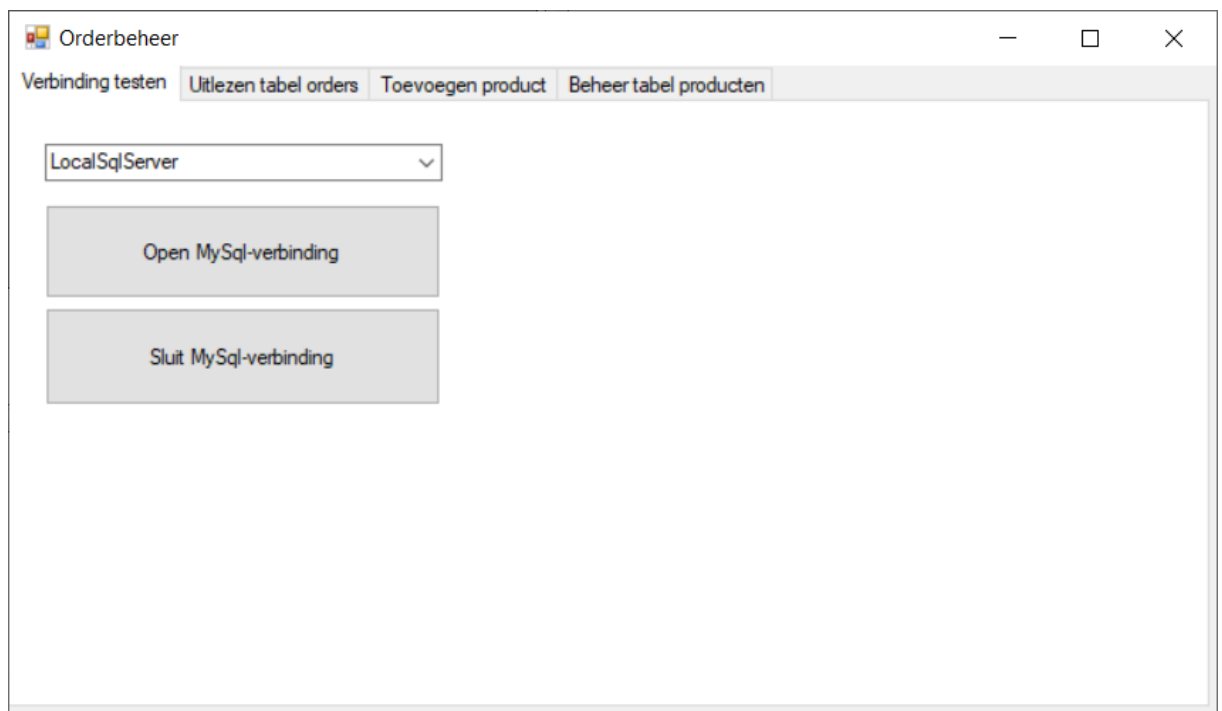
Voeg aan deze een datagridview en twee knoppen toe. Een knop krijgt het opschrift "Lees tabel producten". De andere knop krijgt het opschrift "Verwijder product".

Wanneer je op de knop "Lees tabel producten" klikt dan wordt er kortstondig verbinding gemaakt met de database om de productentabel in te lezen. De ingelezen zaken worden getoond in het datagridview.

Wanneer je in de datagridview een record (rij) selecteerd en je drukt op de knop 'Verwijder product' dan wordt de vraag gesteld of je zeker bent dat het record mag verwijderd worden. Bij bevestiging wordt het betreffend record verwijderd en de inhoud van het datagridview vernieuwd.

In de voorbeeldcode voor de itprodb-database zijn er twee methods voorzien. Een maakt gebruik van de reader. De andere van een MySqlAdapter.

## Screenshots



Orderbeheer

Verbinding testen Uitlezen tabel orders Toevoegen product Beheer tabel producten

Product toevoegen

Productnaam :

productstock :

Product toevoegen

Orderbeheer

Verbinding testen Uitlezen tabel orders Toevoegen product Beheer tabel producten

Lees tabel producten Verwijder product