Database: Data > add new datasource

ALGEMEEN

---

een file "App.config"

- <appSettings>
  - Voor alles wat je opvraagt een key en een value <add key="..." value=""/>
    key vb select/delete/update/insert
- <connectionStrings>: 2 connections:
  - <add name="..."
    connectionstring="server=localhost;user
    id=root;Password="e=mc**2";database=classicmodels"
    providerName="MySql.Data.MySqlClient"/>

```
<configuration>
    <appSettings>
        <add key="select_customers" value="select * from customers"/>
        <add key="select_customer" value="select * from customers
            where customerNumber = @customerNumber" />
        <add key="delete_customer" value="delete from customers
            where customerNumber = @customerNumber"/>
        <add key="update_customer"
            value="update customers set customerNumber=@customerNumber,
            customerName=@customerName,contactLastName=@contactLastName,
            contactFirstName=@contactFirstName,phone=@phone,
            addressLine1=@addressLine1,addressLine2=@addressLine2,
            city=@city,state=@state,postalCode=@postalCode,country=@country,
            salesRepEmployeeNumber=@salesRepEmployeeNumber,
            creditLimit=@creditLimit"/>
        <add key="insert_customer"
            value="insert into customers (customerNumber,customerName,
            contactLastName,contactFirstName,phone,addressLine1,addressLine2,
            city,state,postalCode,country,salesRepEmployeeNumber,creditLimit)
            values (@customerNumber,@customerName,@contactLastName,
            @contactFirstName, @phone,@addressLine1,@addressLine2,@city,@state,
            @postalCode,@country,@salesRepEmployeeNumber,@creditLimit)"/>
    </appSettings>
    <connectionStrings>
        <add
            name="classicmodels"
            connectionString="server=localhost;user id=gast;
                Password=gast;database=classicmodels"
            providerName="MySql.Data.MySqlClient" />
        <add
            name="Winkel.Properties.Settings.classicmodelsConnectionString"
            connectionString="server=localhost;User Id=root;password=&amp;quot;
                e=mc**2&amp;quot;;Persist Security Info=True;database=classicmodels"
            providerName="MySql.Data.MySqlClient" />
    </connectionStrings>
</configuration>
```

DATAREADERS

"DataStorage" waar je al het lezen en opslaan inzet

- -Imports
- -Datamembers: ConnectionStringSettings en DbProviderFactory
- -Constructor
- -GetConnection
- -CreateCommand
- -CreateParam
- -InsertProduct
- -functie die een lijst van de dingen die je opvraagt teruggeeft
- -void insertIets(Iets blah)

```
using System.Data.Common;
using System.Configuration;
using System.Data;

private ConnectionStringSettings connStringSet;
private DbProviderFactory factory;

public DataStorage()
{
   This.connStringSet = ConfigurationManager.ConnectionStrings["classicmodels"]; ;
   factory = DbProviderFactories.GetFactory(connStringSet.ProviderName);
}

protected DbConnection GetConnection()
{
   DbConnection connection = factory.CreateConnection();
   connection.ConnectionString = connStringSet.ConnectionString;
   return connection;
}

private DbCommand CreateCommand(DbConnection conn, DbTransaction trans, string text)
{
   DbCommand command = conn.CreateCommand();
   command.Transaction = trans;
   command.CommandText = text;
   return command;
}

private DbParameter CreateParam(DbCommand command, string parameter, string value)
{
   DbParameter param = command.CreateParameter();
   param.ParameterName = parameter;
   param.DbType = DbType.String;
   param.Value = value;
   return param;

}

public void InsertProduct(Product p)
{
```

```
   // verbinding met de gegevensbank
   DbConnection conn = GetConnection();
   // commando-object aanmaken
   DbCommand command = CreateCommand(conn, ConfigurationManager.AppSettings["insert_products"]);
   // zoekopdracht uitvoeren
        try  {
      command.Parameters.Add(CreateParam(command, "@productCode", p.ProductCode));
      command.Parameters.Add(CreateParam(command, "@productName", p.ProductName));
      command.Parameters.Add(CreateParam(command, "@productLine", p.ProductLine));
      command.Parameters.Add(CreateParam(command, "@productScale", p.ProductScale));
      command.Parameters.Add(CreateParam(command, "@productVendor", p.ProductVendor));
      command.Parameters.Add(CreateParam(command, "@productDescription", p.ProductDescription));
      command.Parameters.Add(CreateParam(command, "@quantityInStock", p.Quantity));
      command.Parameters.Add(CreateParam(command, "@buyPrice", p.Price));
      command.Parameters.Add(CreateParam(command, "@MSRP", p.Msrp));
      conn.Open();
      command.ExecuteNonQuery();
   }
   catch (Exception e) { Console.WriteLine(e.StackTrace); }
   finally { conn.Close(); }

public List<Product> SelectProducts()
{
   List<Product> producten = new List<Product>();

        Idem vorig

   try
   {
      DbDataReader reader = command.ExecuteReader();
      while (reader.Read())
      {
         Product p = new Product();
         p.ProductCode = reader["productCode"].ToString();
         p.ProductName = reader["productName"].ToString();
         p.ProductLine = reader["productLine"].ToString();
         p.ProductScale = reader["productScale"].ToString();
         p.ProductVendor = reader["productVendor"].ToString();
         p.ProductDescription = reader["productDescription"].ToString();
         p.Quantity = reader.GetInt32(6);
         p.Price = reader.GetDouble(7);
         p.Msrp = reader.GetDouble(8);
         producten.Add(p);
      }
   }

        idem vorig
      }

      public void insertOrder(Order ord)
{
        // verbinding met de gegevensbank
   DbConnection conn = GetConnection();
   conn.Open();
   DbTransaction trans = conn.BeginTransaction();
   try
   {
```

```
        // commando-object aanmaken
        DbCommand comOrders = CreateCommand(conn,trans,
                        ConfigurationManager.AppSettings["insert_orders"]);

        comOrders.Parameters.Add(CreateParam(comOrders, "@orderNumber", ord.Number));
        comOrders.Parameters.Add(CreateParam(comOrders, "@orderDate", ord.Ordered));
        comOrders.Parameters.Add(CreateParam(comOrders, "@requiredDate", ord.Required));
        comOrders.Parameters.Add(CreateParam(comOrders, "@shippedDate", ord.Shipped));
        comOrders.Parameters.Add(CreateParam(comOrders, "@status", ord.Status));
        comOrders.Parameters.Add(CreateParam(comOrders, "@comments", ord.Comments));
        comOrders.Parameters.Add(CreateParam(comOrders, "@customerNumber", ord.CustomerNumber));
        comOrders.ExecuteNonQuery();

        foreach (OrderDetail detail in ord.Details)
        {
           DbCommand comOrderDetails = CreateCommand(conn, trans,
                        ConfigurationManager.AppSettings["insert_orderdetails"]);
           comOrderDetails.Parameters.Add(CreateParam(comOrderDetails,
                        "@orderNumber",ord.Number));
           comOrderDetails.Parameters.Add(CreateParam(comOrderDetails,
                        "@productCode",detail.ProductCode));
           comOrderDetails.Parameters.Add(CreateParam(comOrderDetails,
                        "@quantityOrdered",detail.Quantity));
           comOrderDetails.Parameters.Add(CreateParam(comOrderDetails,
                        "@priceEach",detail.Price));
           comOrderDetails.Parameters.Add(CreateParam(comOrderDetails,
                        "@orderLineNumber",detail.OrderLineNumber));
           comOrderDetails.ExecuteNonQuery();
        }

        trans.Commit();
      }
        idem vorig
        }
```

Voor alles wat je opvraagt een klasse (vb product/order/...)
- zoek in de databank (.xsd file) welke parameters elke klasse moet hebben
- getters en setters voor alles (insert snippet)

Je hoofdprogramma: "Program.cs"

DataAdapter en DataSet

Enkel program.cs en AdapterDatastorage.cs

- AdapterDatastorage
  - Dataleden: private DbDataAdapter customerAdapter;
  - private void setColumnEnVersion(ref DbParameter parm, string parameter,
    DbType type, string column, DataRowVersion version)
    {
    parm.ParameterName = parameter;
    parm.DbType = type;
    parm.SourceColumn = column;
    parm.SourceVersion = version;
    }

```
-private DbCommand createCommand(DbConnection conn, string text)
        {
        DbCommand command = conn.CreateCommand();
        command.CommandText = text;
        return command;
}
-public AdapterDatastorage()
        {
        ConnectionStringSettings connString
                = ConfigurationManager.ConnectionStrings["classicmodels"];
        DbProviderFactory factory = DbProviderFactories.GetFactory(connString.ProviderName);
        customerAdapter = factory.CreateDataAdapter();
        DbConnection connection = factory.CreateConnection();
        customerAdapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;
        connection.ConnectionString = connString.ConnectionString;

        customerAdapter.SelectCommand = createCommand(connection,
                ConfigurationManager.AppSettings["select_customers"]);
        DbCommand commando = createCommand(connection,
                ConfigurationManager.AppSettings["delete_customer"]);
        zetDeleteParameters(ref commando);
        customerAdapter.DeleteCommand = commando;
        commando = createCommand(connection,
                ConfigurationManager.AppSettings["update_customer"]);
        zetUpdateParameters(ref commando);
        customerAdapter.UpdateCommand = commando;
        commando = createCommand(connection,
                ConfigurationManager.AppSettings["insert_customer"]);
        zetInsertParameters(ref commando);
        customerAdapter.InsertCommand = commando;
}
- private void zetInsertParameters(ref DbCommand commando)
{
        DbParameter param = commando.CreateParameter();
        setColumnEnVersion(ref param, "@customerNumber",
                DbType.Int32, "customerNumber", DataRowVersion.Current);
        commando.Parameters.Add(param);
        param = commando.CreateParameter();
        setColumnEnVersion(ref param, "@customerName",
                DbType.String, "customerName", DataRowVersion.Current);
        commando.Parameters.Add(param);
                etc...
}
- private void zetUpdateParameters(ref DbCommand commando)
{
        DbParameter param = commando.CreateParameter();
        setColumnEnVersion(ref param, "@customerNumber",
                DbType.Int32, "customerNumber", DataRowVersion.Original);
        commando.Parameters.Add(param);
                etc...
        }
- private void zetDeleteParameters(ref DbCommand commando)
{
        DbParameter customerNumber = commando.CreateParameter();
        setColumnEnVersion(ref customerNumber, "@customerNumber",
                DbType.Int32, "customerNumber", DataRowVersion.Original);
        commando.Parameters.Add(customerNumber);      }
```

```
- public DataSet getCustomers()
    {
            DataSet customers = new DataSet();
            customerAdapter.Fill(customers,"customers");
            return customers;
    }
    - public void updateCustomers(DataSet customers)
    {
            customerAdapter.Update(customers,"customers");
    }
```

Stored Procedures

Een stored procedure is een functie of procedure die bewaard wordt in een gegevensbank
en die door clientapplicaties kan opgeroepen worden.
Deze functie of procedure haalt informatie op uit de gegevensbank of manipuleert zijn data.

stored procedure eerst toevoegen in de mysql terminal

```
        ConnectionStringSettings connString = ConfigurationManager.ConnectionStrings["classicmodels"];
        DbProviderFactory factory = DbProviderFactories.GetFactory(connString.ProviderName);
    DbConnection connection = factory.CreateConnection();
    connection.ConnectionString = connString.ConnectionString;

    DataSet ds = new DataSet();

    connection.Open();

    try
    {
      Console.WriteLine("Blah blah");
      DbCommand command = connection.CreateCommand();
      command.CommandText = ConfigurationManager.AppSettings["getAmount"];
      command.CommandType = CommandType.StoredProcedure;
      DbParameter param = factory.CreateParameter();
      param.DbType = DbType.Int32;
      param.ParameterName = "code";
      param.Value = 114;
      command.Parameters.Add(param);
      DbParameter param2 = factory.CreateParameter();
      param2.DbType = DbType.Double;
      param2.ParameterName = "totaal";
      param2.Direction = ParameterDirection.Output;
      command.Parameters.Add(param2);

      command.ExecuteNonQuery();
      Console.WriteLine(command.Parameters["totaal"].Value);

    }
```