

Komponenten erstellen

Hier mal eine kleine Anleitung zu erstellen eigener Komponenten.
Da dies meine erste Komponente ist hab ich mir gedacht schreib ich gleich mal eine Anleitung wie ich sie gemacht habe.

Da ich nun die Delphi Version 2005 PE habe und diese keine Komponenten für eine Datenbank Anbindung für MySQL hat hab ich mich mal im Netz umgeschaut und nix gefunden. Das einzige was ich gefunden habe ist eine Unit (mysql.pas) welche wir auch unbedingt brauchen um diese Komponente zu schreiben. Dieses Unit kann man unter <http://www.fichtner.net/delphi/mysql.delphi.phtml> kostenlos downloaden.

Benötigt wird auch die libmysql.dll welche bei der Installation von MySQL dabei ist. Diese sollte im System32 Ordner liegen.

So jetzt starten wir aber:

Delphi IDE auf und auf Datei **neu Package für Delphi - Win32** anklicken.
Jetzt habt ihr auf der rechten Seite in der Projektverwaltung ein Package mit dem Namen: **Package1.bpl**.

Dieses werden wir gleich mal speichern und den Namen Test vergeben.
Das sollte dann so aussehen:



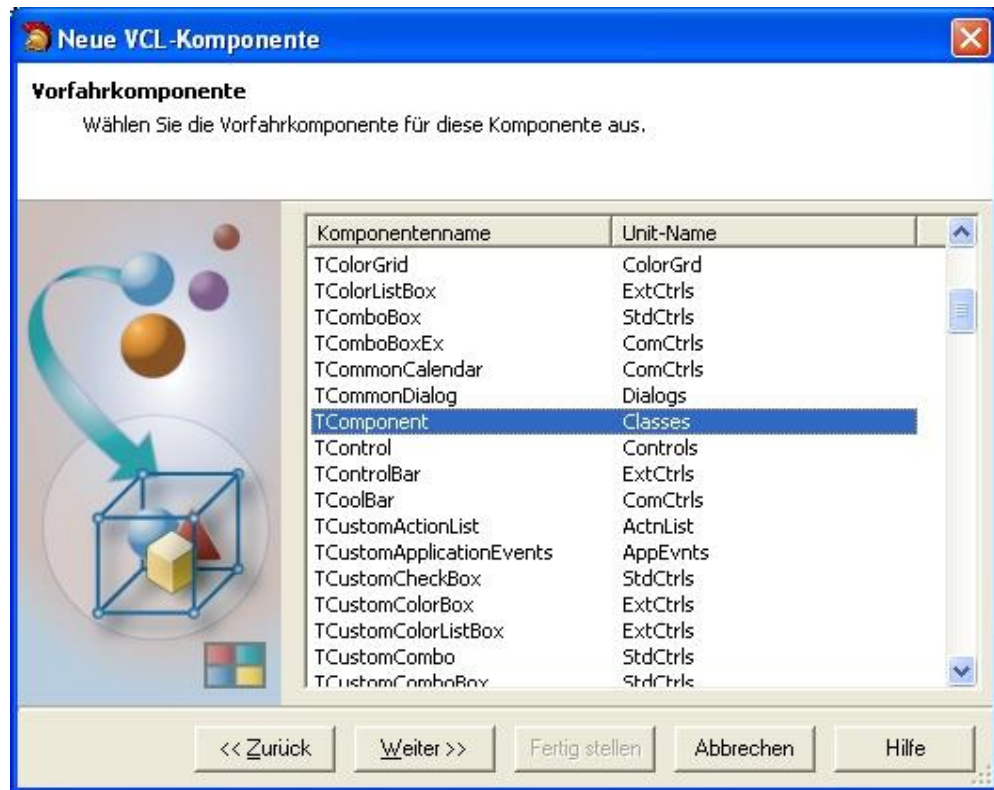
So nun weiter:

Als nächstes klicken wir wieder auf **Datei – Neu – weitere..**
Dann sehen wir folgendes:



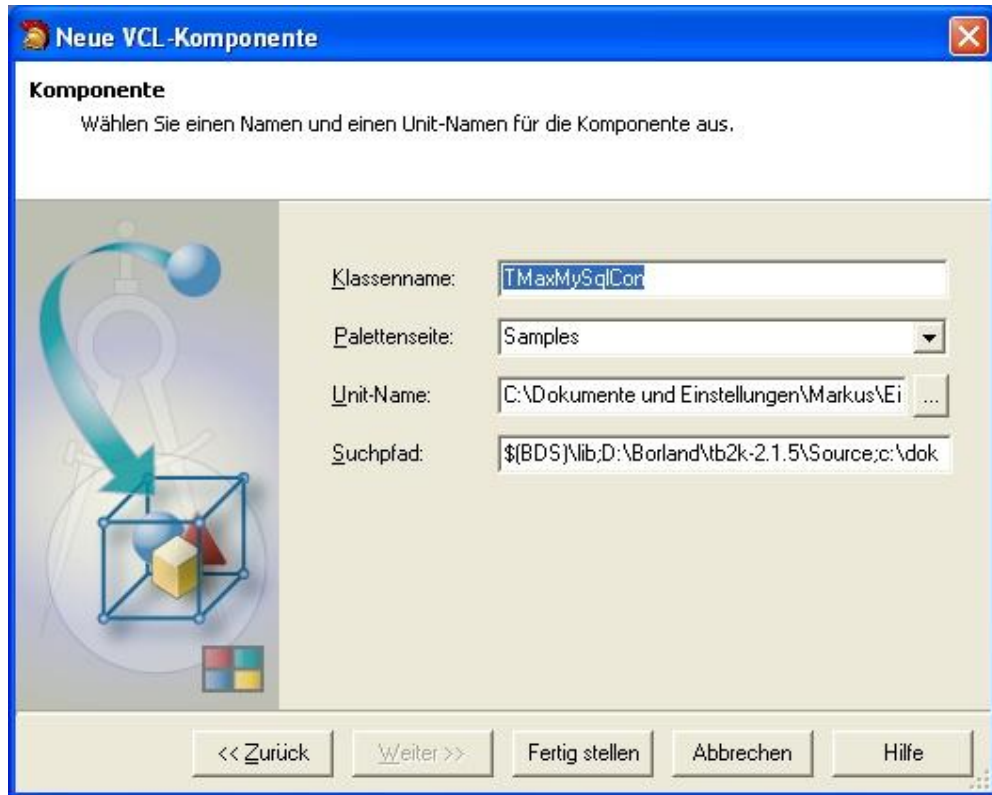
Hier wählen wir unter **Delphi-Projekte – Delphi-Dateien die Komponente** aus und klicken auf OK.

Und noch ein Fenster erscheint:



Hier nun ein bisschen runterscrollen und **TComponent** auswählen und auf Weiter klicken. Wir wählen TComponent aus da es für unsere Komponente keinen geeigneten Vorfahren gibt und TComponent der Vorfahre aller Komponenten ist!

Und nun das letzte Fenster:



In diesen vergeben wir den Klassennamen **TMaxMySQLCon** und klicken auf Fertig stellen.

Als Grundgerüst erhalten wir nun folgendes Unit:

```
unit Component1;
```

```
interface
```

```
uses
```

```
  SysUtils, Classes;
```

```
type
```

```
  MaxMySQLCon = class(TComponent)
```

```
    private
```

```
      { Private-Deklarationen }
```

```
    protected
```

```
      { Protected-Deklarationen }
```

```
    public
```

```
      { Public-Deklarationen }
```

```
    published
```

```
      { Published-Deklarationen }
```

```
  end;
```

```
procedure Register;
```

```
implementation
```

```
procedure Register;
```

```
begin
```

```
  RegisterComponents('Samples', [MaxMySQLCon]);
```

```
end;
```

```
end.
```

Jetzt klicken wir auf Speichern und vergeben für das Unit den Namen MaxMySQLCon.

So als erstes müssen wir noch die **MySql.pas** (Welche wir ja vorher runtergeladen haben in Den Ordner kopieren in dem unser Projekt gespeichert ist!!!

Dann klicken wir in der Projektverwaltung mit der rechten Maustaste auf Test.bpl und dann auf hinzufügen. Danach auf Durchsuchen klicken und dann suchen wir unseren Ordner und wählen die Datei mysql.pas aus. Dann auf Öffnen und dann auf OK.

Wenn das erledigt ist müssen wir dieses Unit auch in unser Projekt einbinden!

Also hinein in unseren **uses Bereich** (Dialogs auch gleich rein):

uses

 SysUtils, Classes, **mysql,Dialogs**;

Wie ihr vielleicht von and. Datenbank-Komponenten wisst kann man unter den Eigenschaften den Datenbanknamen, Port, Usernamen usw. eingeben. Das wollen wir natürlich für unsere Komponente auch anbieten!

Daher muss man wissen in welchen Bereich man diese Eigenschaften setzt.

Solche Eigenschaften müssen immer in den **published Bereich** hinein.

Also schreiben wir mal folgendes hinein:

published

 { **Published-Deklarationen** }

property Host : **String Read** FHost **Write** FHost;

property User : **String Read** FUser **Write** FUser;

property Port : Integer **Read** FPort **Write** FPort;

property Password : **String Read** FPass **Write** FPass;

property DB : **String Read** FDatenbank **Write** FDatenbank;

So hier nun mal eine Erklärung was das da oben soll:

property Host : **String Read** FHost **Write** FHost;

property Host: bedeutet nur das wir in den Eigenschaften der Komponente Host stehen haben.

String : Den Wert den wir dort eingeben können ist vom Typ String.

Read FHost Wir können aus der Variable FHost den Wert lesen.

Write FHost; Wir können in die Variable FHost den Wert schreiben.

Aber wo ist die Variable FHost??

Die geben wir im **Privat Teil** hinein.

private

 { **Private-Deklarationen** }

 FHost : **String**;

 FUser: **String**;

 FPort: Integer;

 FPass : **String**;

 FDatenbank : **String**;

So das bedeutet wir können nun durch die Eigenschaften der Komponente schon mal in unsere Variablen einige Werte setzen aber das genügt ja noch lange nicht.

Also nun wollen wir einige Funktionen von der Unit mysql.pas nutzen.
Also brauchen wir ein Objekt aus diesem Unit.

Nun schreiben wir noch in den Private Teil folgendes:

_MySql : Pmysql;

Verbindung : Boolean; Die brauchen wir auch noch.

Jetzt wollen wir unsere erste function für unsere Komponente schreiben und zwar zum Verbinden zu Datenbank!

Daher auf zum **Public Teil!**

Im Public Teil kommen immer alle Funktionen unserer Komponente!!

```
public
  { Public-Deklarationen }
  function Connect():boolean;
```

Deklariert haben wir sie nun und jetzt auf zur eigentlichen function:

```
function TMaxMySQLCon.Connect():boolean;
begin
  if Connected = false then //Diese Funktion folgt noch etwas weiter unten!!!!
  begin
    _mysql:= mysql_init(nil); //Objekt erzeugen
    if _mysql=nil then //Wenn es nicht erzeugt werden kann Fehlermeldung
    begin
      ShowMessage('Kann Verbindungsobjekt nicht initialisieren');
      result:= false; //Rückgabewert unserer function
      exit; //Raus aus der function
    end;

    //Verbindung herstellen versuchen
    //hier werden unsere Privat Variablen gebraucht und umgewandelt.
    //Und unser Objekt _mysql wird auch an die function mysql_real_connect übergeben.
    //Diese function kommt aus dem Unit mysql.pas

    if mysql_real_connect(_mysql,PChar(Fhost),PChar(FUser),PChar(FPass),
      + PChar(FDatenbank), FPort,nil,0) = nil then
    begin //Wenn der Rückgabewert nicht nil ist hat die Verbindung geklappt
      result:= false;
      ShowMessage(mysql_error(_mysql)); //Fehlermeldung wenn Verbindung nicht
      exit; //zustande gekommen ist. Gleichzeitig wird die
    end; //function mysql_error (mysql.pas) unser Objekt
      //_mysql übergeben und es wird eine
      // Sql-Fehlermeldung gebracht.

      Verbindung:= true; //Hier setzten wir die Variable auf true damit wir später
      //wissen ob wir schon eine Verbindung aufgebaut haben.

      result:= true; //Rückgabewert unserer Function
    end
  else
    result:=true; //Es besteht noch eine Verbindung (Connected=true)
  end;
```

Und nun auf der nächsten Function unserer Komponente:

Hier testen wir ob wir noch verbunden sind!

Daher wieder eine Deklaration in den **Public Teil**:

```
function Connected():boolean;
```

```
function TMaxMySQLCon.Connected():boolean;
```

```
begin
```

```
  try
```

```
  begin
```

```
    if _MySQL = nil then      //Wenn Objekt nicht vorhanden raus
```

```
    begin                    // ist notwendig da sonst Exception kommt
```

```
      result:= false;
```

```
      exit;
```

```
    end;
```

```
    if Verbindung = false then  //Globale Var die in Connect und disconnect gesetzt wird
```

```
    begin                    // ist notwendig da sonst Exception kommt
```

```
      result := false;
```

```
      exit;
```

```
    end;
```

```
    if _MySQL<>nil then
```

```
    begin
```

```
      if mysql_ping(_MySQL) = 0 then //0 bedeutet hat funktioniert
```

```
      result:= true           // Wir haben den Server angepingt (function aus Mysql.pas)
```

```
    else
```

```
      result:= false;
```

```
    end
```

```
    else
```

```
      result:=false;
```

```
    end;
```

```
  except
```

```
    result:=false;
```

```
  end;
```

```
end;
```

Nun können wir schon verbinden und feststellen ob wir verbunden sind.

Man muss aber auch die Verbindung trennen könne und das kommt jetzt mit ..

Der nächsten Function:

Daher wieder was in den Public Teil:

```
function disconnect():boolean;
```

```
function TMaxMySQLCon.disconnect( ):boolean;
```

```
begin
```

```
  if Connected then           //Unsere function testet ob wir verbunden sind.
```

```
  begin
```

```
    mysql_close(_mysql);      //hier wird die Verbindung geschlossen bzw. gelöst.
```

```
    Verbindung:= false;      //Unserer Variablen teilen wir mit das wir nicht mehr verbunden
```

```
    result:= true;           //sind.
```

```
  end
```

```
  else
```

```
    result:= false;          //Wenn wir nicht verbunden sind könne wir auch nichts trennen.
```

```
end;
```

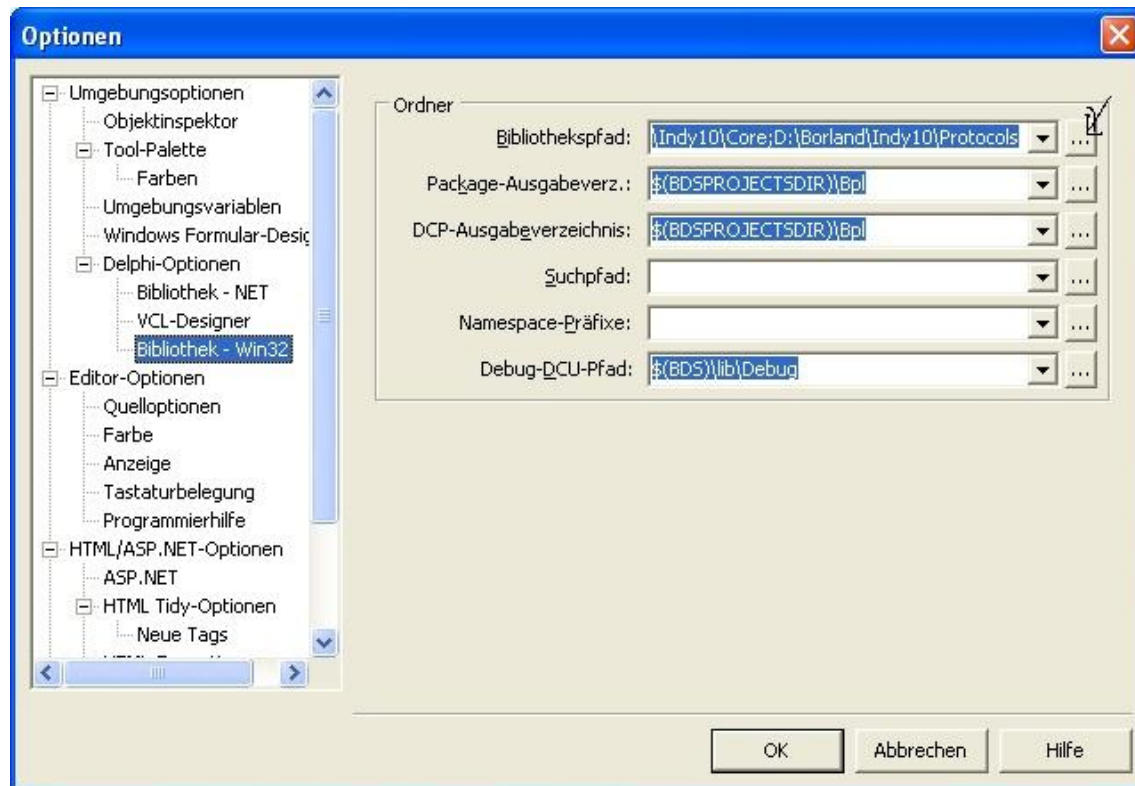
So jetzt wollen wir die Komponente erstellen.

Daher mit der rechten Maustaste auf unser Projekt in der Projektverwaltung klicken und

Auf Compilieren klicken.

Wenn das geklappt hat dann auf Installieren klicken.

Danach müssen wir noch unter Tools – Optionen – klicken und dann wie im Bild ersichtlich auf die 3 Punkte klicken.



Dann im nächsten Fenster wieder auf die Punkte und unseren Projekt-Ordner auswählen.

Und die Komponente sollte unter Samples zu sehen sein.

Hier nun das komplette Unit auf einmal:

```
unit MaxMySqlCon;
```

```
interface
```

```
uses
```

```
SysUtils, Classes, mysql, Dialogs;
```

```
type
```

```
TMaxMySqlCon = class(TComponent)
```

```
private
```

```
{ Private-Deklarationen }
```

```
_MySql : Pmysql;
```

```
FHost : String;
```

```
FUser: String;
```

```
FPort: Integer;
```

```
FPass : String;
```

```
FDatenbank : String;
```

```
Verbindung : Boolean;
```

```
protected
```

```
{ Protected-Deklarationen }
```

```
public
```

```
{ Public-Deklarationen }
```

```
function Connect():boolean;
```

```
function disconnect():boolean;
```

```
function Connected():boolean;
```

```
published
```

```
{ Published-Deklarationen }
```

```
property Host : String Read FHost Write FHost;
```

```
property User : String Read FUser Write FUser;
```

```
property Port : Integer Read FPort Write FPort;
```

```
property Password : String Read FPass Write FPass;
```

```
property DB : String Read FDatenbank Write FDatenbank;
```

```
end;
```

```
procedure Register;
```

```
implementation
```

```
procedure Register;
```

```
begin
```

```
RegisterComponents('Samples', [TMaxMySqlCon]);
```

```
end;
```

```

{ *****Verbindung aufbauen*****}
function TMaxMySqlCon.Connect():boolean;
begin
  if Connected = false then
    begin
      _mySql:= mysql_init(nil); //Objekt erzeugen
      if _mySql=nil then
        begin
          ShowMessage('Kann Verbindungsobjekt nicht initialisieren');
          result:= false;
          exit;
        end;
      //Verbindung herstellen
      if
mysql_real_connect(_mySql,PChar(Fhost),PChar(FUser),PChar(FPass),PChar(FDatenbank),
+
          FPort,nil,0) = nil then
        begin
          result:= false;
          ShowMessage(mysql_error(_mySql));
          exit;
        end;
        Verbindung:= true;
        result:= true;
      end
      else
        result:=true; //Es besteht noch eine Verbindung (Connected=true)
      end;

{ *****noch Verbunden??*****}
function TMaxMySqlCon.Connected():boolean;
begin
  try
    begin
      if _MySql=nil then //Wenn Objekt nicht vorhanden raus
        begin // ist notwendig da sonst Exception kommt
          result:=false;
          exit;
        end;
      if Verbindung=false then //Globale Var die in Connect und disconnect gesetzt wird
        begin // ist notwendig da sonst Exception kommt
          result:= false;
          exit;
        end;
      if _MySql<>nil then
        begin
          if mysql_ping(_MySql)=0 then //0 bedeutet hat funktioniert
            result:= true
          else
            result:= false;
          end
        end
    end
  end

```

```

    else
        result:=false;
    end;
except
    result:=false;
end;

end;

{ *****Verbindung trennen***** }
function TMaxMySQLCon.disconnect():boolean;
begin
    if Connected then
        begin
            mysql_close(_mysql);
            Verbindung:=false;
            result:= true;
        end
    else
        result:= false;
    end;
end.

```

**Hoffe ich konnte es einiger maßen verständlich erklären.
Markus**