C# und MySQL /MariaDB

Als erstes benötigt man die MySql.Data.dll, die man als Verweis hinzufügen muss.
Dazu öffnen wir die Packet-Manager-Konsole durch den Tastenkürzel **ALT + T + N + O** und geben dort ein:
[crayon-67683518be3cf483646185/]
und bestätigen das Konsolenfenster mit Enter.

2. Namespace hinzufügen:
[crayon-67683518be3d6654444888/]
3. StringBuilder erzeugen. Alternativ geht natürlich ein gewöhnlicher String:
[crayon-67683518be3d8875459368/]
4. Connection String erzeugen. Port kann evtl. ein anderer sein:
[crayon-67683518be3da900430990/]

> *In der Regel umschließt man Datenbankverbindungen in einem try/catch. Dies wird hier von meiner Seite aber aus Gründen der besseren Übersicht nicht getan. Solltet Ihr das so verwenden wollen, empfiehlt es sich definitiv einen try/catch Block um rum zu verwenden*

Methode um ein SQL Befehl auszuführen
[crayon-67683518be3db122384328/]
Beispiel:
[crayon-67683518be3dd242844204/]
Methode um einen Befehl aus der Datenbank zu erhalten
[crayon-67683518be3df585007840/]

Beispiel:
[crayon-67683518be3e1536901010/]

Google Chrome – Benutzername und Passwort auslesen

stand heute funktioniert die genannte Methode noch.

Ich möchte mit dieser Anleitung nicht erreichen, dass du damit die Passwörter anderer ausliest, sondern lediglich für eigene Zwecke einsetzt. Ich übernehme daher auch keine Haftung was ihr damit macht. Bitte, tue dir selbst den Gefallen und mache das nicht, denn das kann dein ganzes Leben verändern. Negativ natürlich!

Ich stelle hier 2 Klassen zur Verfügung. Die erste, von mir geschriebene für die Prozesse, die andere ist die Decodierung des Passwortes von einem Autor, vor dem ich meinen Hut ziehe.

Die Datei, welche sämtliche gespeicherte Anmeldedaten enthält findet ihr hier:
[crayon-67683518bea33261362242/]
Dabei lässt sich diese Datei „Login Data“ mit Sqlite öffnen und bearbeiten. Lediglich die Passwortfelder sind als BLOB gekennzeichnet, lassen sich aber wie oben bereits erwähnt dekodieren.

Sobald man ein Objekt der Klasse Chrome erzeugt, muss man als Parameter, einen Dateinamen .db angeben, worauf diese mit denselben Dateinamen mit [Objekt].removeTemp(„Dateiname.db“) gelöscht werden kann.

Das Objekt der Klasse Chrome enthält als Eigenschaft unter anderem 3 ausgelesene Listen: Site, Username, Password

chrome Download

Backgroundworker

Wenn man aus einem Thread heraus eine Zeitaufwändige Operation durchführt, kann dies dazu führen, dass man den Eindruck bekommt, das Fenster wäre eingefroren. Der Grund ist ganz einfach, weil diese Aufgabe im ersten Thread stattfindet und solange dauert, bis es fertig ist. Jetzt stellt .Net den Threading Namespace zur Verfügung womit man für solche Aufgaben einen neuen Thread aufmachen könnte. Eben genau dies tut auch der Backgroundworker nur in einer stark vereinfachter Form.

Backgroundworker einbinden:

1. Namespace hinzufügen:
[crayon-67683518bebb8948427948/]
2. Backgroundworker deklarieren:
[crayon-67683518bebbc609604350/]
3. diesem Objekt events zuweisen:
[crayon-67683518bebbe482338382/]
4. die beiden Methoden implementieren:
[crayon-67683518bebbf425831245/]
5. worker asynchron ausführen lassen:
[crayon-67683518bebc1584021025/]

wünscht man einen Bericht über den derzeitigen Prozess, so muss man auch das event **ProgressChanged** hinzufügen, **ReportProgress(Int32)** in DoWorks ausführen lassen und in

WorkerReportsProgress das Resultat bekommen

Häufigste Problematik die man Anfangs hat ist, dass man nicht auf die Steuerelemente aus dem 1. Thread zugreifen kann. Abhilfe schafft dort
[crayon-67683518bebc3438990344/]

Polymorphie

Unter Polymorphie versteht man, wenn eine Klasse von der anderen erbt.

Dabei nimmt eine Klasse die Rolle der Elternklasse und die anderen, die einer Kindklasse an.

Bedingung:

1. Eigenschaft/Methode muss gleich heißen
2. Die Kindklasse erbt die Elternklasse (LKW : PKW)
3. die Methode der Eltern muss virual sein
4. die Methode des Kindes muss override sein (Das heißt die Elternmethode wird überschrieben/ ergänzt)

das base.[Methodenname] implementiert die Methode aus der Elternklasse in die Kindklasse
[crayon-67683518bede3500800786/]
erstellen wir nun ein Objekt von PKW und lassen die Methode aufrufen, bekommen wir als Ausgabe

[crayon-67683518bede7021115246/]
tun wir dasselbe mit der LKW Klasse, bekommen wir als Ausgabe:
[crayon-67683518bede8330098673/]
sealed – Klasse versiegeln

Generell kann jede Klasse von einer anderen Erben. Möchte man aber vermeiden, dass von einer Klasse geerbt werden soll, nutzt man den Ausdruck sealed. Dies ist Sinnvoll, wenn man weiß, dass man in der Vererbung in der letzten Instanz angekommen ist.
[crayon-67683518bede9442050438/]
abstract – Abstrakte Klassen

Abstrakte Klassen sind Klassen, die reine vererbbare Klassen sind. Das bedeutet man kann aus der Klasse kein Objekt mehr erzeugen.
[crayon-67683518bedeb740249050/]

Eigene Mini Programmiersprache schaffen mithilfe von Regulären Ausdrücken

Filtert nur die Ausdrücke aus, die mit[beginnen und mit] enden.

Match liefert nur den erstgefundenen Wert,

MatchCollection dagegen alle gefundene.
[crayon-67683518befc5522604229/]

Nun könnte man über ein switch/case eine Abfrage erstellen.

Z.B. case FELD1:

tue dies oder jenes

Quellen:

http://www.regexr.com/ – Online Editor

http://www.mycsharp.de/wbb2/thread.php?threadid=41009 – Tutorial

http://www.dotnetperls.com/regex-match – Tutorial

Strukturen – struct

Strukturen sind ähnlich wie Klassen, weisen im Gegensatz zu Ihnen aber folgende Unterschiede auf:

Die Methoden/Konstruktoren haben keine Eigenschaftswerte und Namen

Auf Strukturen kann deutlich schneller zugegriffen werden

Strukturen sind Werttypen und keine Verweistypen

Können nicht erben/vererben

Können keine Konstruktoren ohne Parameter haben

Kleines Beispiel:

[crayon-67683518bf140817924603/]

Das Objekt x greift direkt auf die Variablen Vorwahl, Nummer zu

Das Objekt y dagegen geht den besseren und schnelleren Weg über den Konstruktor v,n und instanziert dann die Variablen dementsprechend.

Textdateien schreiben / lesen

Textdateien schreiben:

Der erste Parameter gibt den Pfad der Textdatei an. Wird nur ein Dateiname angegeben, wird die Datei im selben Ordner erstellt, wo das Programm derzeit läuft. Der 2. Parameter gibt an, wenn es diese Datei schon gibt, ob er die dann überschreiben soll (true) oder neu erstellen soll (false)
[crayon-67683518bf2cf820473194/]
Eine Textdatei auslesen funktioniert auf ähnliche Weise, nur mit dem StreamReader. Mit dieser Möglichkeit wird der Inhalt der Textdatei in den string abc iniatilisiert.
[crayon-67683518bf2d3280263826/]
Möchte man eine ganze Textdatei in einem Rutsch auslesen, geht dies auch ganz einfach mit:
[crayon-67683518bf2d4065409787/]
und schreiben mit:
[crayon-67683518bf2d6828645414/]
Um auch mit deutschen Umlauten zu arbeiten, muss man die Codierung auf UTF-8 stellen:

```
using (sw = new StreamWriter(@exportdatei, false, System.Text.Encoding.Default))
```