Verslag database koppelen met applicatie v1.3– Team Smokey

Laatst bijgewerkt: 22-5-2014  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Projectbegeleider: Sietse Dijks

Klas: RIO4-APO1F, RIO4-APO1E

Periode: 4

Project: Project FIFA

Inhoud

[Keuzes 2](#_Toc388516808)

[SQLite 2](#_Toc388516809)

[XML 2](#_Toc388516810)

[MySQL 2](#_Toc388516811)

[SQL 2](#_Toc388516812)

[SQL Compact 2](#_Toc388516813)

[CSV 2](#_Toc388516814)

[Beste keuze 3](#_Toc388516815)

[Hoe koppel je een database met C#? 3](#_Toc388516816)

[Connection Strings 3](#_Toc388516817)

[Code SQL 4](#_Toc388516818)

# Keuzes

Wij hebben opgezocht welke databases er gekoppeld kunnen worden aan onze c# applicatie en wat wij dachten dat voor onze doeleinde de beste keuze zou zijn. Over de beste keuze komen we later op terug in dit document.

De volgende databases kunnen worden gebruikt voor onze applicatie:

### SQLite

Erg handig voor beginnende programmeurs omdat er geen server voor nodig is. SQLite is eenvoudig uit te lezen en er is geen installatie nodig. SQLite databases zijn enorm klein, een grote database kan minder dan 500KB zijn. SQLite kent maar een bepaald aantal datatypes, namelijk:

* NULL
* INTEGER
* REAL
* TEXT
* BLOB

XML

XML is handig om data uit te lezen via code. Je kunt in XML al je elementen en attributen een eigen naam geven. XML is echter ook enorm inefficiënt vergeleken bij bijvoorbeeld SQLite, omdat databases veel groter worden dan nodig vanwege alle tags.

### MySQL

Handig voor een applicatie, maar veel te ingewikkeld voor wat wij op het moment nodig hebben. MySQL biedt niet veel mogelijkheden, en ook is mySQL niet zo stabiel als andere opties.

### SQL

SQL is enorm handig om te gebruiken in C#, want er zijn genoeg tutorials voor om te volgen, het is eenvoudig te gebruiken en heeft veel functies. Zodra de grootte van een database in SQL groter word dan 15 a 20mb, word de database traag en duurt het lang om er uit te lezen of naar toe te schrijven.

### SQL Compact

SQL compact is goed voor kleinere databases, zoals één die wij nodig hebben. SQL compact is een goede keuze om te gebruiken in een applicatie zoals die van ons. Een database in SQL Compact is meestal maar rond de 5mb. SQL compact werkt slecht met meerdere gebruikers.  
Access

Acces is eenvoudig te gebruiken voor ongevorderde gebruikers omdat het heel eenvoudig in elkaar zit. De nadelen zijn dat het een slechte beveiliging heeft en slechte data kwaliteit. Ook word Acces enorm traag als er heel veel data aanwezig is.

### CSV

In CSV wordt alles gescheden met een komma, op die manier is het eenvoudig te onderscheiden wat bij elkaar hoort. Alleen is het wat onoverzichtelijk voor een gebruiker. Ook is het enorm eenvoudig (net als bij SQL) om data uit de database te lezen. Een probleem is dat alle computersystemen op één na een bepaald teken gebruiken als regeleinde, terwijl dat ene afwijkende systeem (MsWindows) twee tekens voor een regeleinde gebruikt. Dat kan tot conversieverschillen leiden.

# Beste keuze

De beste keuze lijkt ons SQL, omdat onze database niet groot word, en een SQL database word pas sloom zodra de database te groot word. Ook kan internet ons veel hulp bieden omdat SQL één van de meest gebruikte databases is.

# Hoe koppel je een database met C#?

Een database koppelen met C# is vrij eenvoudig. Eerst moet er een connectie geopend worden met de database. Vervolgens kunnen we gegevens uit de database halen, en inladen in een DataGridView, als laatst sluiten we de connectie.

# Connection Strings

Om te kijken welke connection string je nodig hebt voor een database is <http://www.connectionstrings.com/> een enorm handige website om op te kijken, daar staan alle databases met de benodigde code erbij. Ook is in Visual Studio 2012 de benodigde connection string al te zien.

# 

# Code SQL

Omdat wij SQL gaan gebruiken voor onze software is hieronder aangegeven hoe je een SQL connectie opent en daar data uit haalt vanuit C# en Visual Studio.

using System;

using System.Data.SqlClient;

class Program

{

static void Main()

{

//

// First access the connection string.

// ... This may be autogenerated in Visual Studio.

//

string connectionString = ConsoleApplication1.Properties.Settings.Default.ConnectionString;

//

// In a using statement, acquire the SqlConnection as a resource.

//

using (SqlConnection con = new SqlConnection(connectionString))

{

//

// Open the SqlConnection.

//

con.Open();

//

// The following code uses an SqlCommand based on the SqlConnection.

//

using (SqlCommand command = new SqlCommand("SELECT TOP 2 \* FROM Dogs1", con))

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Console.WriteLine("{0} {1} {2}",

reader.GetInt32(0), reader.GetString(1), reader.GetString(2));

}

}

}

}

}