Parameters

Denne artikel beskriver hvorfor parameters er gode.

Den forudsætter lidt kendskab til C# og ADO.NET.

Der findes en tilsvarende artikel med VB.NET.

Skrevet den **18. Feb 2010** af **arne_v** I kategorien **Programmering / C#** | ⭐⭐⭐☆☆

Historie:
V1.0 - 12/11/2005 - original
V1.1 - 14/11/2005 - fix manglende ' i eksempel
V1.2 - 18/10/2010 - smårettelser

Problemerne

Alle der har programmeret op mod en database kender et eller flere af problemerne.

1) uheldige single quotes

Eksempel:

```
sql = "INSERT INTO tt VALUES(" + id + ",'" + name + "')";
```

hvis navnet er Hansen så virker det fint:

```
INSERT INTO tt VALUES(123,'Hansen')
```

men hvis navnet er O'Toole så giver det fejl:

```
INSERT INTO tt VALUES(123,'O'Toole')
```

2) single quotes med vilje (kendt som SQL injection)

Eksempel:

```
sql = "SELECT * FROM myusers WHERE un = '" + username + "' AND pw = '" + password + "'";
```

(efterfulgt af et test på om der blev fundet nogle records)

det virker fint for den pæne bruger som indtaster:
arne
hemmeligt

```
SELECT * FROM myusers WHERE un = 'arne' AND pw = 'hemmeligt'
```

men det returnerer forkert OK for den ondsinded cracker som indtaster:
arne

x' OR 'x' = 'x

SELECT * FROM myusers WHERE un = 'arne' AND pw = 'x' OR 'x' = 'x'

3) dato formater

Den giver altid problemer med input til databasen.

Hvilket format skal man bruge:
dd mm yyyy (dansk)
mm dd yyyy (US)
yyyy mm dd (sorterings rigtigt)
?

Skal værdierne:
sættes i ''
sættes i ##
konverteres med en funktion
?

Styres formatet af:
operativ system sprog version
database sprog version
styre system sprog indstilling
database sprog indstilling
?

Løsningen

En begynder løsning på de 2 første problemer er at fordoble
alle single quotes:

sql = "INSERT INTO tt VALUES(" + id + ",'" + name.Replace("'","''") + "')";

INSERT INTO tt VALUES(123,'Hansen')

INSERT INTO tt VALUES(123,'O''Toole')

sql = "SELECT * FROM myusers WHERE un = '" + username.Replace("'","''") + "' AND pw = '" +
password.Replace("'","''") + "'";

SELECT * FROM myusers WHERE un = 'arne' AND pw = 'hemmeligt'

SELECT * FROM myusers WHERE un = 'arne' AND pw = 'x'' OR ''x'' = ''x'

og det virker, men det er ikke super godt:
* det er ikke kønt
* det er nemt at glemme
* forskellige databaser kan have forskellige andre tegn som også kan misbruges
* det løser ikke dato problemet

Dato problemet undlader man ofte helt at løse. Man hardkoder SQL sætningerne
med formatet til det system man udvikler på. Enten med en DateTime ToString eller
ved noget banal string manipulation.

Og så får man problemet når man skal have det til at køre på en anden maskine.

Den rigtige løsning som løser alle problemerne er at bruge parameters fremfor at sætte værdier ind i selve SQL strengen.

Med parameters skriver man bare en placeholder alle de steder i ens SQL hvor der skal indsættes værdier og så sætter man de værdier og ADO.BNET håndterer alle problemerne.

Det lyder måske lidt mystisk, men lad os tage nogle eksempler.

Kode eksempler

Eksemplerne vil bruge SQLServer som database, men alle eksemplerne virker lige så godt med Access eller MySQL, man skal bare rette fra SqlXxxx til OleDbXxxx eller MySqlXxxx (og måske tilrette nogle data typer som kan hedde noget forskelligt).

De data som køres på er:

```
CREATE TABLE tt (
    id INTEGER PRIMARY KEY,
    name VARCHAR(50)
)
GO

CREATE TABLE myusers (
    un VARCHAR(32) PRIMARY KEY,
    pw VARCHAR(32)
)
GO

INSERT INTO myusers VALUES('arne', 'hemmeligt')
GO

CREATE TABLE dtest (
    i INTEGER PRIMARY KEY,
    d DATETIME
)
GO
```

Først INSERT med single quotes:

TestPrep1.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
```

```
namespace TestParam
{
    public class TestClass
    {
        public static void Main(string[] args)
        {
            SqlConnection con = new SqlConnection("Server=ARNEPC3;Integrated
Security=SSPI;Database=Test");
            con.Open();
            SqlCommand cmd = new SqlCommand("INSERT INTO tt VALUES (@id,
@name)", con);
            cmd.Parameters.Add("@id", SqlDbType.Int);
            cmd.Parameters.Add("@name", SqlDbType.VarChar, 50);
            cmd.Parameters["@id"].Value = 123;
            cmd.Parameters["@name"].Value = "Hansen";
            cmd.ExecuteNonQuery();
            cmd.Parameters["@id"].Value = 124;
            cmd.Parameters["@name"].Value = "O'Toole";
            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
}
```

Bemærk at vi ikke sætter '' omkring placeholder når det er en String.

Og en gang mere.

TestPrep2.cs:

```

using System;
using System.Data;
using System.Data.SqlClient;

namespace TestParam
{
    public class TestClass
    {
        public static bool IsValid(string un, string pw)
        {
            SqlConnection con = new SqlConnection("Server=ARNEPC3;Integrated
Security=SSPI;Database=Test");
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT * FROM myusers WHERE un =
@un AND pw = @pw", con);
            cmd.Parameters.Add("@un", SqlDbType.VarChar, 50);
            cmd.Parameters.Add("@pw", SqlDbType.VarChar, 50);
            cmd.Parameters["@un"].Value = un;
            cmd.Parameters["@pw"].Value = pw;
            SqlDataReader rdr = cmd.ExecuteReader();
            bool res = rdr.Read();
```

```
            rdr.Close();
            con.Close();
            return res;
        }
        public static void Main(string[] args)
        {
            Console.WriteLine(IsValid("anonymous", ""));
            Console.WriteLine(IsValid("arne", "hemmeligt"));
            Console.WriteLine(IsValid("arne", "x' OR 'x' = 'x"));
        }
    }
}
```

Og til sidst dato.

TestPrep3.cs:

```
using System;
using System.Threading;
using System.Data;
using System.Data.SqlClient;

namespace TestParam
{
    public class TestClass
    {
        public static void Main(string[] args)
        {
            SqlConnection con = new SqlConnection("server=ARNEPC3;Integrated
Security=SSPI;database=Test");
            con.Open();
            SqlCommand ins = new SqlCommand("INSERT INTO dtest VALUES (@i,
@d)", con);
            ins.Parameters.Add("@i", SqlDbType.Int);
            ins.Parameters.Add("@d", SqlDbType.DateTime);
            for(int i = 0; i < 10; i++)
            {
                ins.Parameters["@i"].Value = i;
                DateTime dt = DateTime.Now;
                ins.Parameters["@d"].Value = dt;
                ins.ExecuteNonQuery();
                Thread.Sleep(1000);
            }
            SqlCommand sel = new SqlCommand("SELECT * FROM dtest WHERE d >
@d", con);
            sel.Parameters.Add("@d", SqlDbType.DateTime);
            DateTime cut = DateTime.Now.AddSeconds(-5);
            sel.Parameters["@d"].Value = cut;
            SqlDataReader rdr = sel.ExecuteReader();
            while(rdr.Read())
            {
```

```
                int i = (int)rdr[0];
                DateTime dt = (DateTime)rdr[1];
                Console.WriteLine(i + " " + dt);
            }
            rdr.Close();
            con.Close();
        }
    }
}
```

Stored Procedures

Det er ikke nødvendigt at bruge stored procedures for at bruge parameters.

Men hvis man bruger stored procedures så er syntaxen for parameters den samme.

Konklusion

Det er en selvfølge at man bruger parameters når man skal igang med seriøs brug af ADO.NET.

Eksemplerne i denne artikel skulle gerne have givet et indblik i hvordan man bruger parameters.

Kommentar af skwat d. 15. Nov 2005 | 1

SMUKT intet mindre, denne artikel burde være skrevet for længe siden.

Kommentar af thomasso (nedlagt brugerprofil) (nedlagt brugerprofil) (nedlagt brugerprofil) d. 17. Dec 2005 | 2

God og velskrevet artikel om området.
Var rimelig bekendt med parameters i forvejen, men rart at få opfrisket det igen.

Kommentar af mysitesolution d. 05. Jul 2007 | 3

God artikel... Hvis man bruger meget databaser i C# er dette også værd at kigge på
http://en.wikipedia.org/wiki/Language_Integrated_Query

Kommentar af innercitydk d. 05. Dec 2005 | 4

Super artikel Arne. Jeg har ikke rigtig kunne finde "ordentligt" rundt i det før nu :D

Kommentar af scarface335 d. 05. Sep 2006 | 5

Kommentar af fufan d. 01. Jul 2006 | 6