Państwowa Wyższa Szkoła Zawodowa
im. Stanisława Pigonia
w Krośnie

Kierunek: Informatyka
Specjalność: Technologie internetowe i bazy danych
Przedmiot: Administrowanie baz danych
Imię i nazwisko: Adrian Kut
Numer albumu: 23821

Temat:

Uwierzytelnianie użytkownika w aplikacjach internetowych

Opis aplikacji

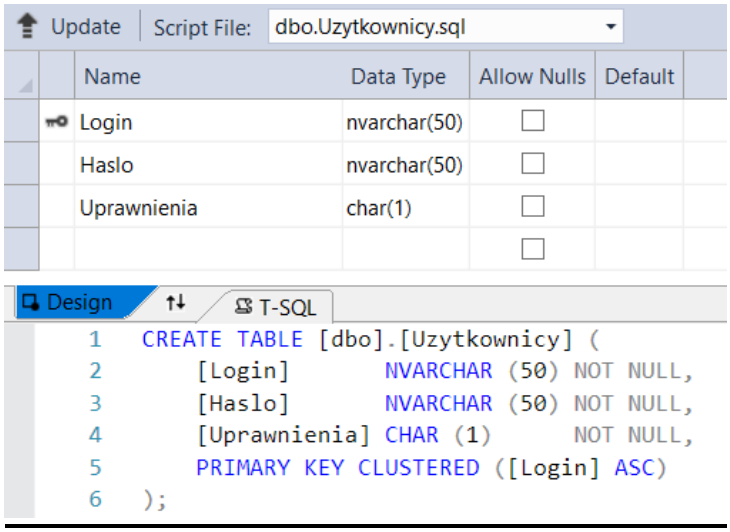
Aplikacja została napisana w taki sposób, że do zalogowania się można użyć dwóch przycisków. Jeden z nich pozwala na użycie funkcjonalności LINQ, zaś drugi zwykłego połączenia z bazą danych przy pomocy SQLConnection. Początkowo możliwe było zalogowanie się podając nieprawidłowe dane, wpisując w polu Login na przykład ''jan' --'' co powodowało zakomentowanie pozostałej części skryptu SQL i finalnie zalogowanie się do aplikacji. Dopiero dodanie poniższych parametrów do skryptu, pozwoliło na uniknięcie tego błędu.

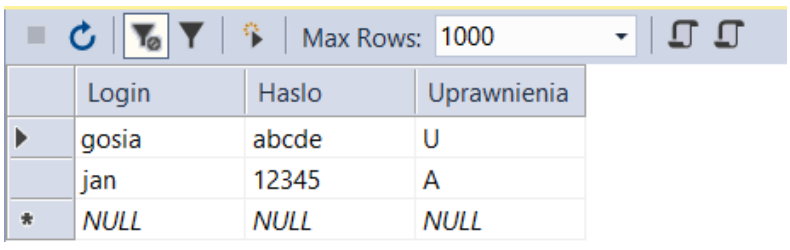
```
SqlParameter parLogin = new SqlParameter("PLogin", sLogin);
SqlParameter parHaslo = new SqlParameter("PHaslo", sHaslo);
        sqlCommand.Parameters.Add(parLogin);
        sqlCommand.Parameters.Add(parHaslo);
```

Do ostatecznej weryfikacji danych posłużyła procedura składowa

```
CREATE PROCEDURE [dbo].[SprawdzHaslo]
(
@log varchar(50),
@pass varchar(50),
@ile int OUTPUT
)
AS
SELECT @ile = count(*) FROM Uzytkownicy WHERE Login=@log AND Haslo=@pass
RETURN
```

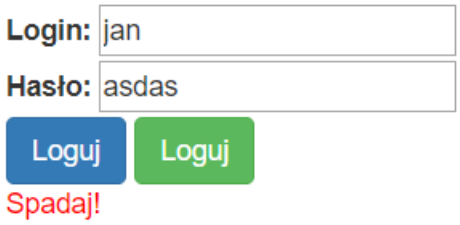
Tabela Użytkownicy

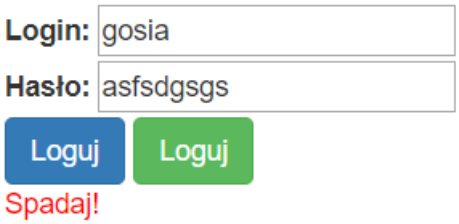




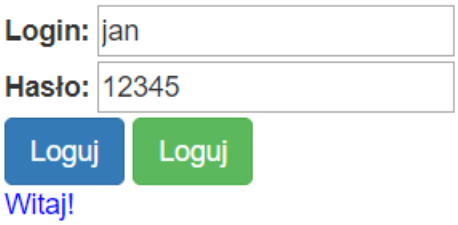
Widok aplikacji

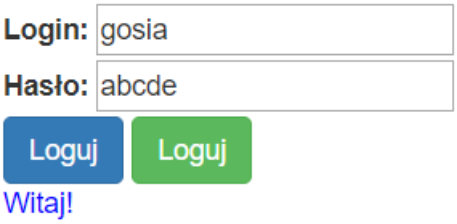
Po wprowadzeniu błędnych danych nie jest możliwe zalogowanie się do aplikacji, niezależnie od wyboru przycisku.



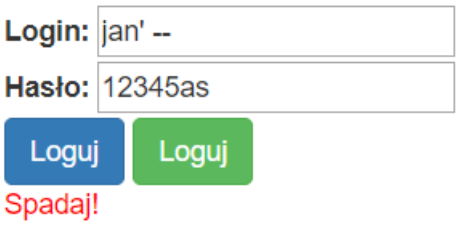


Po wprowadzeniu prawidłowego loginu i hasła, otrzymujemy stosowny komunikat, który informuję nas o prawidłowym połączeniu z bazą „zalogowaniem się”. Oba przyciski działają.





Aplikacja została odpowiednio zabezpieczona przed tego typu wstrzykiwaniem kodu.



Inne możliwe zabezpieczenia

- Zabezpieczenie witryny za pomocą protokołu HTTPS
- Ukrycie tekstu wprowadzanego w polu „Hasło:’’ poprzez zmianę typu tego pola → „password”
- Wymuszanie posiadania silnego hasła
- Wprowadzenie uwierzytelniania dwuetapowego

Kod

LINQ ze zwykłym połączeniem do bazy

```
  var wynik = from l in dcContext.Uzytkownicies
                                where (l.Login == sLogin) && (l.Haslo == sHaslo)
                                select l.Login;
                    return (wynik.Count() != 0);
```

LINQ z obsługą procedury składowej

```
int? x = 0;
                    DataClasses1DataContext dcContext = new
DataClasses1DataContext(@connectionString);
                    dcContext.SprawdzHaslo(sLogin, sHaslo,ref x);
                    if (x > 0)
                    {
                        return true;
                    }
                    else
                    {
                        return false;
                    }
```

Finalna wersja kodu aplikacji

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Uwierzytelnianie
{
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        protected void Button1_Click(object sender, EventArgs e)
        {

            komunikat("button1");
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
            komunikat("button2");
        }
        private void komunikat(string who)
        {
            if (CzyDobryLoginHaslo(txtLogin.Text, txtHaslo.Text, who))
            {
                lKomunikat.ForeColor = System.Drawing.Color.Blue;
                lKomunikat.Text = "Witaj!";
            }
            else
            {
```

```
                lKomunikat.ForeColor = System.Drawing.Color.Red;
                lKomunikat.Text = "Spadaj!";
            }
        }
        private bool CzyDobryLoginHaslo(string sLogin, string sHaslo, string who)
        {
            string connectionString = "Data Source=.;Initial Catalog=UserSQL;Integrated
Security=True";
            bool bOk = false;
            try
            {
                    //LINQ TO SQL
                if (who.Equals("button1"))
                {
                     int? x = 0;
                    DataClasses1DataContext dcContext = new
              DataClasses1DataContext(@connectionString);
                    dcContext.SprawdzHaslo(sLogin, sHaslo,ref x);
                    if (x > 0)
                    {
                        return true;
                    }
                    else
                    {
                        return false;
                    }
                }
                    //ZWYKŁE ŁĄCZENIE SIĘ Z BAZĄ
                if (who.Equals("button2"))
                {

                    SqlConnection cnUsers = new SqlConnection(@connectionString);
                    SqlParameter parLogin = new SqlParameter("PLogin", sLogin);
                    SqlParameter parHaslo = new SqlParameter("PHaslo", sHaslo);
                    SqlParameter ileParametr = new SqlParameter("@ile", 0);
                    ileParametr.Direction = ParameterDirection.Output;

                    string SQLString = "SELECT * FROM Uzytkownicy WHERE Login=@PLogin AND
Haslo=@PHaslo";
                    SqlCommand sqlCommand = new SqlCommand(SQLString, cnUsers);

                    sqlCommand.Parameters.Add(parLogin);
                    sqlCommand.Parameters.Add(parHaslo);
                    sqlCommand.Parameters.Add(ileParametr);

                    sqlCommand.Connection.Open();
                    sqlCommand.ExecuteNonQuery();

                    SqlDataAdapter daUsers = new SqlDataAdapter(sqlCommand);
                    DataSet dsUsers = new DataSet();
                    daUsers.Fill(dsUsers);
                    bOk = (dsUsers.Tables[0].Rows.Count > 0);


                    return bOk;
                }
                return false;
            }
            catch
            {
                return false;
            }
        }
    }
}
```