Państwowa Wyższa Szkoła Zawodowa w Krośnie

Laboratorium

Uwierzytelnianie użytkownika w aplikacjach internetowych

Celem ćwiczenia jest utworzenie mechanizmu logowania użytkownika w aplikacji internetowej.

Czynności wstępne

1. Uruchom Visual Studio.
2. Otwórz panel SQL Server Object Explorer i utwórz nową bazę danych o nazwie *UsersSQL*.
3. W bazie utwórz tabelę *Uzytkownicy* o następującej strukturze:

Login nvarchar(50), PK	**Haslo** nvarchar(50)	**Uprawnienia** char(1)
jan	12345	A
gosia	abcde	U

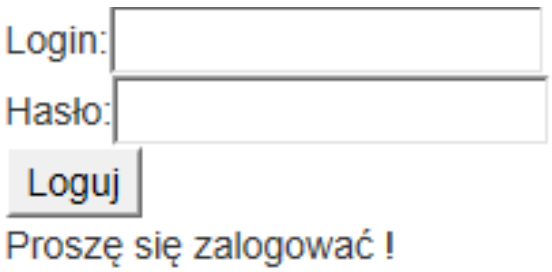
4. Wypełnij tabelę danymi jak w przykładzie powyżej. Będą one służyły do sprawdzania tożsamości użytkownika.

Przebieg ćwiczenia

1. Utwórz w Visual Studio nowy projekt w języku C# typu *ASP.NET Web Application* i nadaj mu nazwę.
2. W oknie *ASP.NET Templates* wybierz *Web Forms* i naciśnij OK.
3. W nowej aplikacji stworzymy własny mechanizm uwierzytelniania.
4. Otwórz plik default.aspx w widoku źródła (*Source*) i usuń całą zawartość wewnątrz znacznika *BodyContent*:

```
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
</asp:Content>
```

5. Zawartość tę zastąpimy własnym formularzem logowania. Umieść wewnątrz znacznika *BodyContent* dwa elementy typu *TextBox*, jeden *Button* oraz etykietę *Label*. Dodaj opisy (*Login, Hasło, Dalej, Proszę się zalogować*), aby strona wyglądała podobnie jak na rysunku poniżej:



6. Ustaw identyfikatory (ID) elementów typu TextBox jako *txtLogin*, *txtHaslo*, zaś *Label* jako *lKomunikat*.
7. Utwórz metodę obsługującą kliknięcie przycisku (*Button1_Click*).
8. Uzupełnij kod metody następująco:

```
protected void Button1_Click(object sender, EventArgs e)
  {
      if (CzyDobryLoginHaslo(txtLogin.Text, txtHaslo.Text))
      {
          lKomunikat.Text = "Witaj!";
      }
      else
      {
          lKomunikat.Text = "Spadaj!";
      }
  }
```

9. Sprawdzanie tożsamości na podstawie podanych loginu i hasła będzie realizowała metoda *CzyDobryLoginHaslo*. Na początku utwórz metodę (prywatną), która zwraca zawsze *true* (lub *false*). Uruchom aplikację i zweryfikuj, czy komunikat się wyświetla.
10. Następnie rozszerzymy kod metody *CzyDobryLoginHaslo*, aby sprawdzała dane użytkownika według danych z tabeli:

```
private bool CzyDobryLoginHaslo(string sLogin, string sHaslo)
   {
       bool bOk = false;

       try
       {
           SqlConnection cnUsers = new SqlConnection(@"Data
Source=(localdb)\MSSQLLocalDB;Initial Catalog=UsersSQL;Integrated
Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False");

           string sSQL = "SELECT * FROM Uzytkownicy WHERE Login='" + sLogin + "'
AND Haslo='" + sHaslo + "'";
           SqlDataAdapter daUsers = new SqlDataAdapter(sSQL, cnUsers);
           DataSet dsUsers = new DataSet();
           daUsers.Fill(dsUsers);

           bOk = (dsUsers.Tables[0].Rows.Count > 0);

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

```
        }
```

11. Uruchom aplikację i sprawdź, czy mechanizm logowania działa dla poprawnych i niepoprawnych danych logowania.
12. W polu Login wpisz następujący tekst: `jan' --`
13. Dlaczego aplikacja wpuszcza użytkownika mimo niepoprawnego hasła ? Wyjaśnij jakie polecenie jest przesyłane do serwera bazy danych.
14. W polu Login wpisz: `sdds' OR '1'='1' --`
15. Dlaczego aplikacja wpuszcza użytkownika mimo niepoprawnego loginu ? Wyjaśnij jakie polecenie jest przesyłane do serwera bazy danych.
16. Zmodyfikuj kod procedury *CzyDobryLoginHaslo* na następujący:

```
        protected bool CzyDobryLoginHasloParametr(string sLogin, string sHaslo)
        {
            bool bOk = false;

            try
            {
                SqlConnection cnUsers = new SqlConnection(@"Data
Source=(localdb)\MSSQLLocalDB;Initial Catalog=UsersSQL;Integrated
Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False");

                SqlParameter parLogin = new SqlParameter("PLogin", sLogin);
                SqlParameter parHaslo = new SqlParameter("PHaslo", sHaslo);

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

                sSelect.Parameters.Add(parLogin);
                sSelect.Parameters.Add(parHaslo);

                SqlDataAdapter daUsers = new SqlDataAdapter(sSelect);
                DataSet dsUsers = new DataSet();
                daUsers.Fill(dsUsers);

                bOk = (dsUsers.Tables[0].Rows.Count > 0);

                return bOk;
            }
            catch
            {
                return false;
            }

        }
```

17. Uruchom aplikację i sprawdź, czy poprzednie ataki nadal się powiodą. Na czym polega zabezpieczenie programu ?

Zadania dodatkowe

1. Do weryfikacji danych użyj procedury składowanej *SprawdzHaslo* o treści jak poniżej.

```
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
```

W celu jej wywołania dodaj do kodu metody *CzyDobryLoginHaslo* dodatkowy parametr wyjściowy:

```
SqlParameter ileParameter = new SqlParameter("@ile", 0);
ileParameter.Direction = ParameterDirection.Output;:
```

oraz użyć metody *ExecuteNonQuery* klasy *SQLCommand*.

2. Do weryfikacji danych użyj technologii *LINQ to SQL Classes* typu *Object Relational Mapping* (ORM). Dodaj ją do projektu (*Add -> New item*), a następnie przeciągnij tabelę *Uzytkownicy* do obszaru projektowego pliku *dbml*. W metodzie *CzyDobryLoginHaslo* użyj kodu:

```
        bool CzyDobryLoginHaslo(string sLogin, string sHaslo)
        {
            try
            {
                DataClasses1DataContext dcContext = new DataClasses1DataContext(@"Data
Source=(localdb)\MSSQLLocalDB;Initial Catalog=UsersSQL;Integrated
Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False");

                var wynik = from l in dcContext.Uzytkownicies
                            where (l.Login == sLogin) && (l.Haslo == sHaslo)
                            select l.Login;

                return (wynik.Count() > 0);

            }
            catch
            {
                return false;
            }
        }
```

3. Połącz technikę *Linq to SQL Classes* z użyciem procedury składowanej *SprawdzHaslo*. Przeciągnij ją na obszar projektowy pliku dbml, a następnie wywołaj ją w metodzie *CzyDobryLoginHaslo*.

4. Jakie inne usprawnienia związane z bezpieczeństwem można wykonać w aplikacji ?