

# Projektowanie aplikacji ADO.NET + ASP.NET

## Zestaw 4

Autentykacja, autoryzacja

29-11-2016

Liczba punktów do zdobycia: **10/40**

Zestaw ważny do: 06/13-12-2016

1. (**1p**) Zaprezentuj w praktyce przedstawiany na wykładzie mechanizm autentykacji **Windows**. Ściślej - przygotuj aplikację, w której użytkownik zostanie rozpoznany jako aktualnie zalogowany użytkownik systemu operacyjnego (formalnie: użytkownik który jest właścicielem procesu przeglądarki internetowej).

Naucz się różnicy między autentykacją w trybie podstawowym (przeglądarka zapyta o login i hasło) od uwierzytelniania w trybie zintegrowanym (użytkownik systemu/domeny zostanie rozpoznany automatycznie).

2. (**3p**) Naucz się korzystać z mechanizmu autentykacji **Forms**, opartego o ciastko autentykacji.

Zaimplementuj i użyj we własnej aplikacji takiego dostawcę usługi uwierzytelniania (**MembershipProvider**), który potwierdzi tożsamość użytkownika w bazie danych Microsoft SQL Server, w tabeli **USER**, w której zapisana będzie nazwa i email użytkownika, oraz tabeli **PASSWORD** gdzie zapisane będą skojarzone z użytkownikiem:

- skrót hasła, sól (salt), liczba rund haszowania oraz data ustawienia hasła (zgodnie ze schematem przedstawionym na wykładzie)
- (lub) skrót, sól i liczba rund w jednym - w wyniku użycia funkcji klucza pochodnego (*Key Derivation Function*), np. `bcrypt` (implementacji `bcrypt` nie ma w biblioteki standardowej, poszukaj gotowej implementacji w sieci)

Zbuduj formularz dodawania/rejestracji użytkownika, który po utworzeniu konta poprawnie zapisze dane do obu tabel (użytkownika i hasło).

Pokaż że użytkownicy poprawnie logują się do aplikacji.

Odpowiedz na pytania:

- (a) dlaczego po stronie serwera hasła użytkownika nie można przechować w postaci jawnej?
- (b) dlaczego niektóre funkcje skrótu (które?) są niewskazane w praktyce?
- (c) do czego potrzebna jest dodatkowa wartość (salt) przy wyliczaniu skrótu?
- (d) dlaczego hasła przechowuje się w osobnej tabeli i nie wystarczy do tego kolumna (kolumny) w tej samej tabeli w której przechowuje się listę użytkowników?
- (e) jakie mechanizmy ochrony przed atakami typu brute-force można zastosować w typowej aplikacji internetowej?

- (f) jak obsłużyć scenariusz w którym użytkownik zapomniał hasła i chce je w jakiś sposób odzyskać?
3. (1p) Poprzednie zadanie rozwiń o implementację usługi dostawcy ról (**RoleProvider**), gdzie role zapisane byłyby w tabeli ROLES, a powiązanie wiele-do-wielu użytkowników z rolami w tabeli USERSROLES.

Dostęp do zasobów można zabezpieczyć przez wskazanie ról użytkowników którzy mogliby do tych ról mieć dostęp. Pokaż, że można to robić zarówno dla pojedynczych zasobów (sekcja `location` w `web.config`) oraz całych podfolderów (osobny, zdegenerowany `web.config`).

4. (1p) Jak korzystać z informacji o rolach użytkowników w aplikacji?

Pokaż, że potrafisz zablokować dostęp do podglądu i edycji **wybranego** elementu tabeli (GridView/Listview) dla użytkowników będących w konkretnej roli.

Na przykład pole PESEL powinien widzieć każdy, a edytować tylko użytkownik będący w roli ADMINISTRATOR, zaś pole PENSJA powinien widzieć i edytować tylko użytkownik w roli PLACOWA.

5. (2p) Rozwiń przykład z wykładu demonstrujący technikę zastępowania modułu Forms Authentication przez moduł Session Authentication.

Pokaż jak zapamiętać więcej informacji o użytkowniku: email, imię, nazwisko, wiek. Pokaż jak skorzystać z tej dodatkowej informacji w aplikacji do zablokowania dostępu do jakiejś funkcji użytkownikowi.

Za <http://www.wiktorzychla.com/2014/11/forms-authentication-revisited-for-net.html>

6. (2p) Naucz się korzystać z biblioteki DotNetOpenAuth do autentykacji OAuth2 do wybranego dostawcy autentykacji (Google/Facebook/LiveID).

Za <http://www.wiktorzychla.com/2014/11/simple-oauth2-federated-authentication.html>

Szczegóły komunikacji z dostawcami <http://benfoster.io/blog/oauth-providers>

Wiktor Zychla