Projektowanie aplikacji ASP.NET
Zestaw 7

Autentykacja, autoryzacja

2024-11-19

Liczba punktów do zdobycia: **12/60**
Zestaw ważny do: 2024-12-10 (zestaw wspólny do wykładów 7 i 8 dlatego termin ważności jest dłuższy)

1. (**1p**) Zaprezentuj w praktyce przedstawiany na wykładzie mechanizm autentykacji zintegrowanej (`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).

   Uwaga! Windows 10 Home, w przeciwieństwie do większości innych edycji systemu, nie ma domyślnie aktywnej konfiguracji uwierzytelniania zintegrowanego. Żeby je aktywować, należy posłużyć się wskazówkami zawartymi w notatce `https://stackoverflow.com/a/48197001/941240`.

2. (**3p**) Naucz się korzystać z mechanizmu autentykacji opartego o ciastko autentykacji. Wykorzystaj moduł **Forms** lub **SessionAuthenticationModule** dla .NET.Framework lub wbudowane middleware autentykacji dla .NET.Core.

   Zaimplementuj i użyj we własnej aplikacji taki mechanizm usługi uwierzytelniania, który potwierdzi tożsamość użytkownika w bazie danych (Microsoft SQL Server lub PostgreSQL), 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) lub **PBKDF2** (implementacja jest w bibliotece standardowej, klasa `Rfc2898DeriveBytes`)

   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.

   W przypadku wersji dla .NET.Framework można (ale nie trzeba) architekturę klas odpowiedzialnych za weryfikację pary login/hasło oprzeć o omówiony na wykładzie mechanizm (**MembershipProvider**).

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. (**2p**) Poprzednie zadanie rozwiń o implementację usługi ról, 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 pojedyńczych zasobów (sekcja `location` w `web.config`) oraz całych podfolderów (osobny, zdegenerowany `web.config`). W przypadku MVC pokaż jak wymagać określonych ról użytkownika za pomocą atrybutu **Authorize**.

   W przypadku .NET.Framework można skorzystać z architektury (**RoleProvider**).

4. (**3p**) Zaimplementuj logowanie OAuth2/OpenIDConnect do autentykacji OAuth2 do wybranego dostawcy autentykacji (Google/Facebook/Azure).

   Za
   `https://www.wiktorzychla.com/2014/11/simple-oauth2-federated-authentication.html`
   oraz
   `https://developers.google.com/identity/protocols/oauth2/web-server?hl=en` oraz
   `https://learn.microsoft.com/pl-pl/aspnet/core/security/authentication/social/google-l`

5. (**3p**) Dodaj uwierzytelnianie dwuskładnikowe do aplikacji. Formalnie - wykorzystaj bibliotekę Otp.NET lub równoważną, przechowuj w bazie danych klucz główny użytkownika, a na serwerze sprawdzaj czy kod przesłany na formularzu logowania jest identyczny z tym wygenerowanym na serwerze.

   Do testów użyj którejkolwiek aplikacji klienckiej TOTP (np. Google Authenticator, Microsoft Authenticator). Każda z tych aplikacji posiada wygodny tryb rejestrowania klucza głównego za pomocą kodu QR, ale do zaliczenia tego zadania nie trzeba generować kodów QR w swojej aplikacji, wystarczy tryb rejestrowania przez wpisanie kodu do aplikacji autentykującej.

Wiktor Zychla