Obrona

Bezpieczne aplikacje Web w oparciu o ASP.NET2.0

Artur Żarski

stopień trudności

ASP.NET 20.0 jest technologią, dzięki której możliwe jest tworzenie dynamicznych stron internetowych. Wykorzystuje ona większość możliwości, jakie dostępne są wraz z platformą .NET Framework w związku z wykorzystaniem środowiska uruchomieniowego CLR.

Zastosowanie programowania opartego na zdarzeniach pozwoliło między innymi nad odseparowanie kodu aplikacji i warstwy definiującej wygląd. Dodatkowo możliwe jest tworzenie kodu naszej aplikacji w różnych językach .NET – C# oraz Visual Basic .NET. W najnowszej wersji ASP.NET 2.0 usprawniono w znaczący sposób wiele cech znanych z poprzednich wersji platformy, a także wprowadzono wiele nowych elementów, wspomagających proces szybkiego tworzenia aplikacji internetowych w szczególności mających związek z bezpieczeństwem. Tworzenie aplikacji, bez względu na to, czy piszemy aplikacje dla środowiska klient-serwer czy dla środowiska Web, zawsze wymaga zastanowienia się nad aspektami bezpieczeństwa. Tylko od nas zależy, czy nasza aplikacja będzie działać poprawnie i czy użytkownicy będą mogli się do niej włamać.

ASP.NET 2.0 jako technologia bardzo mocno wspiera programistów, aby tworzyć bezpieczne rozwiązania. Niniejszy artykuł ma na celu przedstawienie podstawowych zasad i mechanizmów tworzenia bezpiecznych aplikacji WWW. Na początku powinniśmy zastanowić się nad samym pojęciem bezpieczeństwa. Termin ów sam w sobie jest bardzo szerokim tematem i obejmuje bardzo wiele pojęć i zagadnień. Na rysunku pierwszym widać ogólny przekrój przez technologie oraz elementy związane z procesem bezpieczeństwa.

Jak widać na Rysunku 1, podstawowe technologie to:

- IIS jako serwer WWW
- ASP.NET jako platforma pisania aplikacji

Z artykułu dowiesz sie

- Artykuł przedstawia dostępne technologie platformy .NET, dzięki którym aplikacje Web mogą stać się bezpieczniejsze. Pokazane są aspekty, na które należy zwrócić uwagę przy tworzeniu i konfigurowaniu całego środowiska aplikacji

Co powinienes wiedzieć

- Przed przystąpieniem do lektury artykułu należy zapoznać się z podstawami bezpieczeństwa aplikacji. Ze względu na ograniczenie do platformy .NET należy znać podstawowe elementy tej platformy oraz orientować się w jej technologiach.

- Enterprise Services jako obiekty biznesowe klasy enterprise
- SQL Server jako baza danych

Wszystkie te technologie składają się z dwóch podstawowych elementów - autoryzacji oraz autentykacji. Pojęcia te bardzo się od siebie różnią, a mimo to często są mylone. Autentykacja to proces pobierania danych użytkownika i jego uprawnień. Natomiast autoryzacja to proces sprawdzania, czy użytkownik ma dostęp do pewnych zasobów.

Elementy technologii oraz jej elementów możemy przedstawić w tabeli

Bezpieczeństwo w ASP.NET

Bezpieczeństwo ASP.NET jest bardzo szerokim zagadnieniem, podobnie jak samo pojęcie bezpieczeństwa. Opiera się ono o mechanizmy, które są istotne dla całej platformy .NET. Składa się z bardzo wielu elementów, które mogą mieć wpływ na sposób działania naszej aplikacji WWW oraz decydować o tym, jak dużo czasu będziemy musieli poświęcić na oprogramowanie wszystkich mechanizmów oraz uniknięcie pewnych niechcianych zachowań. Poniżej przedstawiamy elementy mające wpływ na bezpieczeństwo aplikacji ASP.NET.

Tabela 1.

IIS	
Autoryzacja	Autentykacja
Uprawnienia NTFS	Autoryzacja Windows
Restrykcje IP	Dostęp anonimowy
	Certyfikaty
	Dostęp podstawowy
ASP.NET	
Autoryzacja	Autentykacja
URL	Windows
Pliki	Formularze
Role .NET	Passport
	Własne
Enterprise Services	
Autoryzacja	Autentykacja
Role COM+	RPC
Uprawnienia NTFS	
SQL Server	
Autoryzacja	Autentykacja
Użytkownicy	Windows
Uprawnienia do obiektów	SQL Server
Role bazy danych	
Role użytkownika	
Role aplikacyjne	

CAS - Code Access Security

ASP.NET jest składnikiem platformy .NET Framework i rządzi się podobnymi prawami, jak każda aplikacja .NET. Ogólnie rzecz biorąc, proces wygląda tu tak, że *Common Language Runtime* (CLR) zezwala kodowi aplikacji wykonywać tylko te operacje, do których wykonania ma on uprawnienia. Jeśli chcielibyśmy narzucić jakąś politykę bezpieczeństwa, musimy posiłkować się CAS (*Code Access Security*). CAS jest mechanizmem bezpieczeństwa CLR wymuszającym politykę bezpieczeństwa i umożliwia:

- Definiowanie, co może robić kod
- Definiowanie, jaki program może wywoływać dany kod
- Identyfikowanie kodu

Do głównych zadań CAS należą między innymi:

- Ograniczenia narzucone kodowi w zakresie zdefiniowanych zasad bezpieczeństwa
- Definiowanie uprawnień do zasobów systemowych
- Żądanie określonych uprawnień z kodu
- Przydzielanie uprawnień ładowanemu kodowi w zakresie zasad bezpieczeństwa
- Konfigurowanie przez administratora zasad bezpieczeństwa dla grup kodu

Dla wielu programistów .NET CAS nie jest znany i nie mają oni świadomości jego możliwości. Może to stanowić pewien problem, ponieważ każda aplikacja wykorzystuje CAS, a w zasadzie podlega jego regułom. Dlaczego tak się dzieje? Może to wynikać z faktu, że większość naszych stron WWW, które piszemy, uruchamiamy lokalnie, na lokalnym użytkowniku, etc. W momencie uruchomienia aplikacji Web w rozproszonym środowisku, coś nagle przestaje działać.

Uwierzytelnianie w ASP.NET 2.0

Znane jest już nam pojęcie autentykacji. Zobaczmy teraz, jak wygląda proces uwierzytelniania. W ASP.NET wyróżniamy następujące sposoby uwierzytelniania: systemu Windows, za pośrednictwem formularza oraz przy użyciu usługi Passport. Możliwy jest też jeszcze jeden podział wg trudności uwierzytelniania. Wyróżnić możemy następujące jego sposoby:

- Proste – zintegrowane oraz formularze
- Średniozaawansowane – szyfrowanie hasła, wykorzystanie sygnatur (hash) (np. NTLM, Digest)
- Zaawansowane – certyfikaty, Kerberos

W przypadku autoryzacji, wykorzystywany jest mechanizm Role Based Security, czyli zabezpieczenia bazujące na rolach. Polegają one na sprawdzeniu tożsamości użytkownika i pozwoleniu bądź odrzuceniu żądania do zasobu.

Zintegrowana autoryzacja

Najbardziej popularne i najczęściej stosowane są mety uwierzytelniania zintegrowanego oraz poprzez formularze. Pierwszy z nich bazuje na ustawieniach serwera IIS i największa część pracy wymagana jest po stronie jego konfiguracji. Jak wyglądają poszczególne ustawienia w uwierzytelnianiu zintegrowanym, przedstawia Tabela nr 2. Kolumna rodzaj uwierzytelnienia to konkretne sposoby, natomiast kolumna zachowanie określa, jak ten proces wygląda.

Formularze

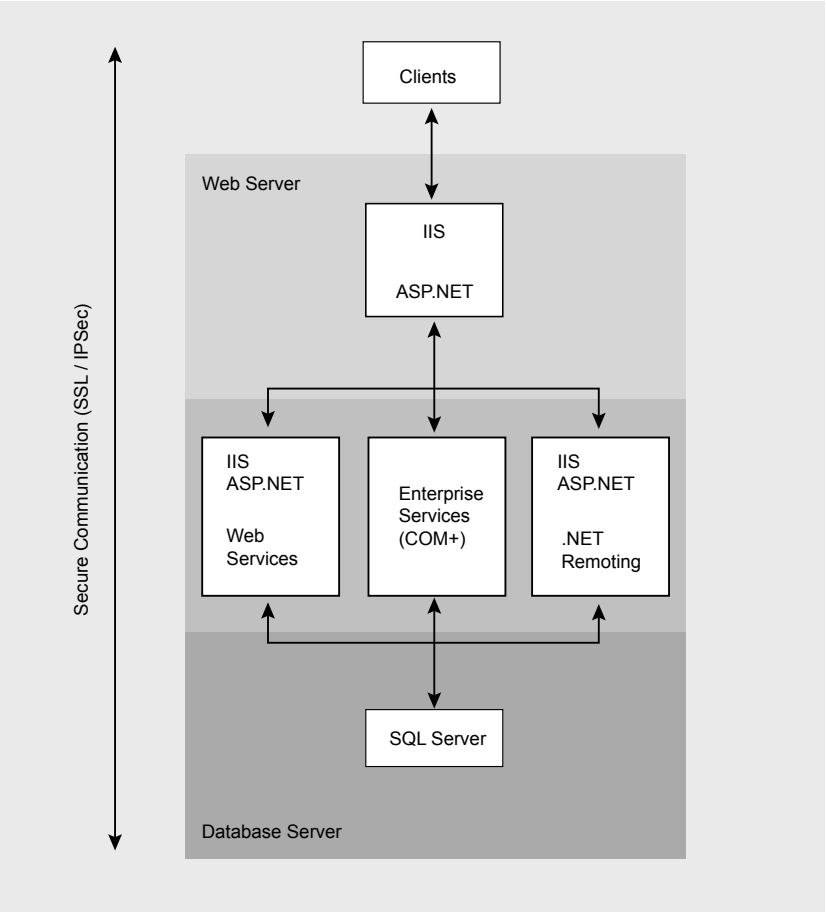
Autoryzacja na poziomie formularzy jest zupełnie odmiennym procesem. O ile w przypadku uwierzytelniania zintegrowanego nie wprowadzaliśmy nazwy użytkownika i hasła, to w przypadku formularzy musimy te wartości podać. Dopiero na ich podstawie system da nam dostęp do zasobów i zaloguje lub odrzuci zgłoszenie. Logowanie zwykle odbywa się na specjalnej stronie WWW, na której podajemy nazwę użytkownika i hasło. Zaletą tego typu autoryzacji jest możliwość uwierzytelniania do dowolnej listy użytkowników. Ta lista może być przechowywana w bazie danych, może to być dowolny serwer LDAP lub dowolne inne źródło – plik XML, lista użytkowników trzymana w pliku Web.config, etc.

Przykładowa lista użytkowników w pliku Web.config może wyglądać jak na Listingu 1.

W przypadku uprawnień przechowywanych w bazie danych może to być (i najczęściej tak właśnie jest) zwykła tabela, w której mamy dwa pola: użytkownik i hasło.

Passport authentication

Jeszcze jednym sposobem autentykacji jest użycie usługi Microsoft .NET Passport. Usługę tę najczęściej można znaleźć na stronach firmowych Microsoft. Niewątpliwą jej zaletą jest to, że posiadając konto w usłudze Passport, możemy mieć jednocześnie dostęp do różnych witryn (np. nasza własna witryna oraz witry-



Rysunek 1. *??*

Tabela 2. *Sposoby zintegrowanego uwierzytelniania*

Sposób uwierzytelnienia	Zachowanie
Dostęp anonimowy	Nie chronimy nic – każdy użytkownik ma dostęp
Basic authentication	Login i hasło jawnie przesyłane w nagłówku HTTP Wykorzystanie tylko z SSL
Digest authentication	Przesyłanie hasha hasła
Windows Integrated authentication	challenge-response, dwufazowe, NTLM
Mapowanie certyfikatów	Bardzo uniwersalne, obsługiwane przez wszystkie systemy Możliwość mapowania certyfikatu na konto użytkownika techniką jeden-do-jednego oraz jeden-do-wielu Możliwość przechowywania certyfikatów na bezpiecznych nośnikach
Kerberos	Wymaga Active Directory oraz Windows XP/2000 u klienta

na na stronach MSDN). Niestety ma ona jedną wadę, a jest nią sposób implementacji usługi - do tego celu musimy wykorzystać specjalny SDK.

ASP.NET 2.0 i elementy dla programisty

Wraz z ASP.NET 2.0 programiści dostali całkiem spory zasób kontrolek, które pozwalają w bardzo prosty i szybki sposób zaimplementować proces logowania i autoryzacji w systemie. Na rysunku 2 widać zestaw nowych kontrolek, które są dostępne w Visual Studio.

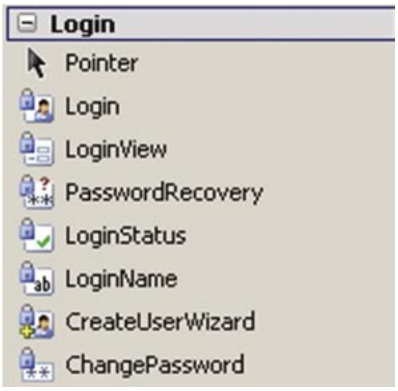
Najważniejsze z tych kontrolek to:

- CreateUserWizard – uruchamia wizarda, dzięki któremu możemy stworzyć użytkownika
- Login – czyli logowanie
- LoginStatus – kontrolka odpowiadająca za stan zalogowania – określa, czy użytkownik został zalogowany czy nie.
- PasswordRecovery – możliwość odzyskania hasła na podstawie jakieś tajemniczego pytania
- ChangePassword – kontrolka umożliwiająca w bardzo prosty sposób zmianę hasła.

Co najistotniejsze z punktu widzenia programisty, to fakt, że pozbywamy się czasu na implementacje tego fragmentu aplikacji. Po prostu przenosimy kontrolkę na formularz, a następnie konfigurujemy. Cała praca związana z tworzeniem przycisków, etykiet, podłączaniem do źródeł danych etc. jest już zrobiona. Na rysunku 3 przedstawione zostały niektóre kontrolki związane z logowaniem i użytkownikami.

Web Site Administration Tool

- Bardzo istotnym elementem z punktu widzenia bezpieczeństwa aplikacji ASP.NET jest narzędzie Web Site Administration Tool. Jest to narzędzie, dzięki któremu mamy możliwość utworzenia użytkowników, grupy użytkowników, serwera pocztowego, dostawców usług uwierzytelnienia, etc. Rysunek 4 przedstawia główny ekran tego narzędzia. Narzędzie to jest bardzo proste w obsłudze i bardzo intuicyjne – aby np. stworzyć nowego użytkownika, wystarczy w menu użytkownicy wybrać opcję „New”, a następnie podać odpowiednie informacje, a użytkownik zostanie założony.



Rysunek 2. *??*

Praca z danymi

Praca z danymi w przypadku aplikacji ASP.NET odbywa się na podobnych zasadach, jak w przypadku zwykłych aplikacji okienkowych. Po-

Listing 1. *???*

```
<authentication mode="Forms">
<forms name="frmLog" loginUrl="/logowanie.aspx">
     <credentials passwordFormat="SHA1">
          <user name="Artur"
password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>
          <user name="Karol"
          password="BA56E5E0366D003E98EA1C7F04ABF8FCB3753889"/>
     </credentials>
</forms>
</authentication>
```

Listing 2. *???*

```
  <connectionStrings>
    <add name="DD_cnx"
        connectionString="Data Source=ARTURZ02\SQLEXPRESS;Initial Catalog=DeveloperDays;Persist Security Info=True;User
                    ID=dd_user; Password=haslo;"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Listing 3. *???*

```
<connectionStrings>
  <add name="Pubs" connectionString="Server=localhost;Integrated Security=True;Database=Pubs"
    providerName="System.Data.SqlClient" />
  <add name="Northwind" connectionString="Server=localhost;Integrated Security=True;Database=Northwind"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

trzebne nam jest źródło danych, połączenie, zestaw zapytań oraz miejsce, do którego trafią wyniki naszego zapytania. Źródłem danych może być dowolny element: baza danych, plik, obiekt, etc. Istotnym elementem jest sposób podłączenia się do naszego źródła. Najczęściej wykorzystuje się klasę *SqlConnection* i podaje jako parametr *ConnectionString*. *ConnectionString* jest niczym innym, jak tylko zestawem informacji, które pozwalają na identyfikację serwera, identyfikację bazy danych oraz parametry użytkownika.

Nasz kod do połączenia z bazą danych może wyglądać jak na Listingu 5.

W ten sposób zapisane dane nie są jednak bezpieczne. Dlatego też ConnectionString można przenieść do pliku Web.config. Wtedy w pliku tym pojawi się sekcja podobna do sekcji z Listingu 2.

Przenosząc jawnie zapisany ciąg połączenia do pliku Web.config, zrobiliśmy pierwszy krok, aby zabezpieczyć się przed najprostszymi sposobami włamania do serwera. Niestety w przypadku bardziej zaawansowanych użytkowników również i taki sposób zapisu nie chroni nas przed niepowołanym dostępem. Dlatego też wraz z ASP.NET mamy możliwość szyfrowania *ConnectionString* w plikach Web.config. Do tego celu wykorzystujemy polecenie dostępne w ramach .NET Framework: aspnet_regiis. Jeśli nasz plik web.config wygląda tak jak na Listingu 3

To po wykonaniu polecenia:

```
aspnet_regiis -pe
      "connectionStrings"
      -app "/NaszaAplikacja"
```

Dostaniemy wynik jak na Listingu 4.

Gdzie w polu CipherValue będzie zaszyfrowany nasz ciąg.

Listing 4. *???*

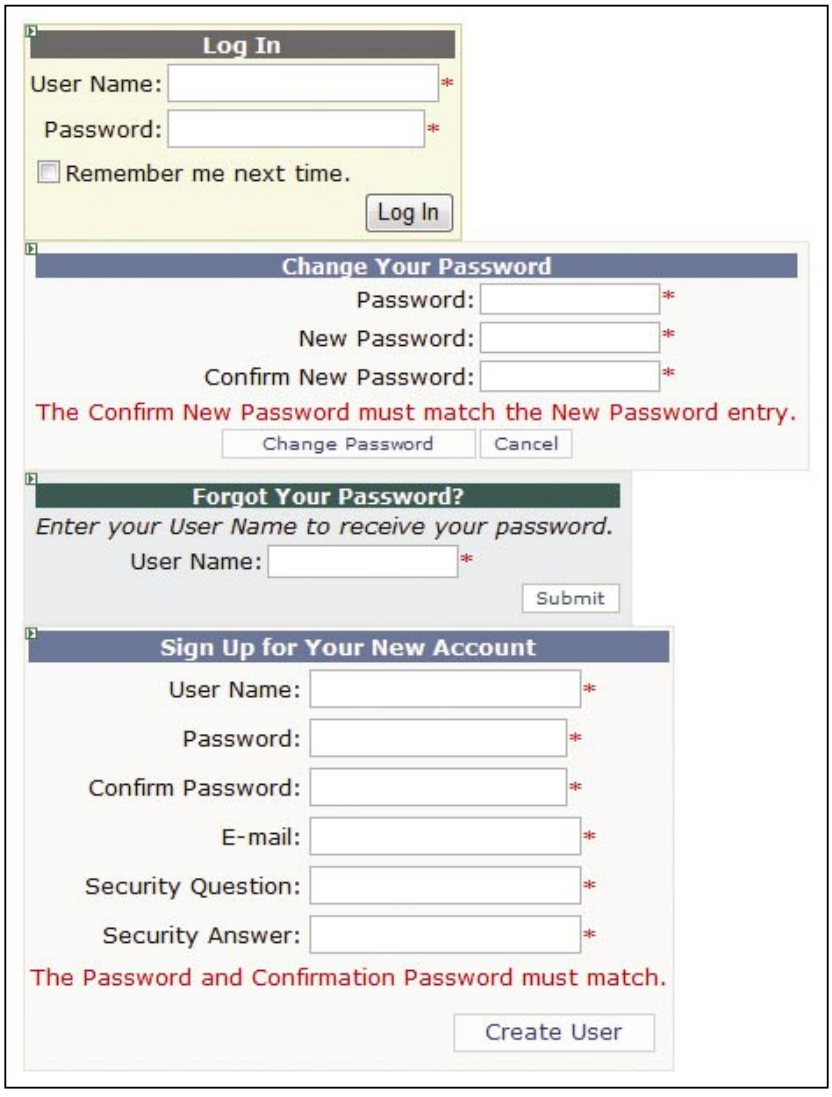
```
<connectionStrings>
  <EncryptedData>
    <CipherData>
      <CipherValue>AQAAANCMndjHoAw
           </CipherValue>
    </CipherData>
   </EncryptedData>
</connectionStrings>
```

Listing 5. *???*

```
SqlConnection cnxPolaczenie =
      New SqlConnection(
            „Data Source=
      ARTURZ02\SQLEXPRESS;
Initial Catalog=DeveloperDays;
      Persist Security Info=
            True;User ID=dd_user;
      Password=haslo;”);
```

Serwer WWW

Oczywiście nie tylko sama aplikacja musi być bezpieczna. Ważne jest też, aby i serwer WWW był bezpieczny. Co może sprawić, ze nasza aplikacja będzie działać bezpiecznie? Oczywiście szyfrowanie transmisji danych przy wykorzystaniu protokołu SSL. SSL (*Secure Socket Layer*) jest protokołem sieciowym używanym do bezpiecznych połączeń internetowych. Został opracowany przez firmę Netscape i powszechnie go przyjęto jako standard szyfrowania na WWW. Normalnie strony z serwerów oraz formularze do serwera są przesyłane przez sieć otwartym tekstem, który stosunkowo łatwo przechwycić (szczególnie w sieci lokalnej). Jeśli serwer używa protokołu SSL do komunikacji z przeglądarką, wówczas informacja w obie strony (między serwerem WWW i przeglądarką) jest przesyłana przez sieć w sposób zaszyfrowany, co odszyfrować jest już stosunkowo trudno. SSL realizuje szyfrowanie, uwierzy-

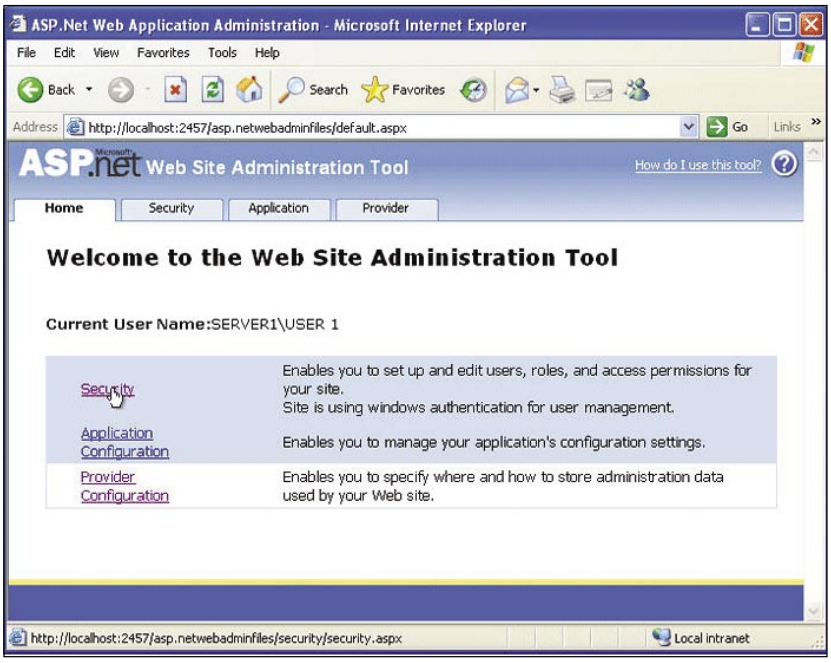


Rysunek 3. *??*

telnienie serwera (ewentualnie użytkownika również) i zapewnienie integralności oraz poufności przesyłanych informacji. W momencie nawiązania połączenia z bezpieczną (stosującą protokół SSL) stroną www następuje ustalenie algorytmów oraz kluczy szyfrujących, stosowanych następnie przy przekazywaniu danych między przeglądarką a serwerem WWW.

Podsumowanie

Bezpieczeństwo aplikacji nie jest już wyłącznie domeną administratorów. Szczególnego znaczenia nabierają te słowa w przypadku tworzenia aplikacji WWW. ASP.NET 2.0 daje szerokie możliwości zabezpieczenia się przed niepowołanym dostępem. Oczywiście nieuniknione jest wsparcie ze strony systemu operacyjnego oraz serwera WWW, niemniej jednak sama platforma .NET oferuje całkiem spore możliwości. Niniejszy artykuł bardzo ogólnie przedstawił podstawowe elementy, aby pokazać czytelnikom, jak zabrać się za aspekty bezpieczeństwa i bezpiecznego tworzenia aplikacji WWW. Nikt nie powinien twierdzić, że tworzenie bezpiecznych aplikacji jest banalnie proste, ale jednocześnie nie można popadać w druga skrajność. Umiejętne użycie gotowych mechanizmów pozwoli nam na bardziej efektywną i twórczą pracę bez konieczności skupiania się na najmniejszych detalach związanych z implementacją naszej aplikacji (jak np. zmiana lub przypomnienie zapomnianego hasła) ●



Rysunek 4. ??

W Sieci

- Oficjalna strona ASP.NET – *http://www.asp.net/*
- Bezpieczeństwo aplikacji ASP.Net przygotowane przez grupę Patterns & Practices – *http://msdn2.microsoft.com/en-us/library/ms998372.aspx*
- Fragment książki – ASP.NET 2.0 security – *http://www.awprofessional.com/articles/article.asp?p=351414&rl=1*
- Książka Developing More-Secure Microsoft® ASP.NET 2.0 Applications – *http://www.microsoft.com/mspress/books/9989.aspx*
- Podstawy SSL – *http://pl.wikipedia.org/wiki/Transport_Layer_Security*
- SSL teoria – *http://muflon.photosite.pl/doc/progzesp/ssl.html*