STUDIA INFORMATICA 2003
Volume 24 Number 3 (55)

Marek MITTMANN
Politechnika Śląska, Instytut Informatyki

IMPLEMENTACJA APLIKACJI WIELOWARSTWOWYCH W TECHNOLOGII .NET

Streszczenie. W artykule przedstawiono możliwości jednej z najnowszych technologii wspomagających tworzenie aplikacji wielowarstwowych. Omówione zostały aspekty związane z wykorzystaniem technologii .NET przy tworzeniu poszczególnych warstw aplikacji: od realizacji komunikacji z bazą danych, poprzez implementację powłoki reguł biznesowych, aż po budowanie interfejsu użytkownika.

Słowa kluczowe: technologia .NET, architektura wielowarstwowa.

IMPLEMENTATION OF MULTI-TIER APPLICATIONS WITH .NET TECHNOLOGY

Summary. The paper presents the one of newest technologies, which supports developing the multi-tier applications. There were explained some aspects of creating each of tiers with .NET technology: communication with database, business logic and user interface.

Keywords: .NET technology, multi-tier architecture.

1. Wstęp

Współczesny komputer, zarówno używany w pracy jak i ten domowy, jest coraz częściej podłączony do sieci komputerowej. Łączność z innymi komputerami daje użytkownikom wiele korzyści, o których jeszcze kilkanaście lat temu można było co najwyżej pomarzyć. Korzystanie z tych udogodnień nie byłoby jednak możliwe bez odpowiedniego oprogramowania. Przeciętny użytkownik Internetu lub firmowej sieci komputerowej nawet nie zdaje sobie sprawy z tego, jak może ono być skomplikowane. A to dlatego, że zazwyczaj ma kontakt jedynie z interfejsem aplikacji w postaci stron WWW lub prostego programu

okienkowego, który jest tylko niewielkim wycinkiem całości. Pozostałe elementy, niezbędne do działania, znajdują się na zdalnych maszynach.

Większość programów użytkowych przechowuje niezliczone ilości danych. W szczególności dotyczy to aplikacji pracujących w sieci, które muszą poradzić sobie z natłokiem informacji dla wielu użytkowników. Do ich przechowywania wykorzystuje się serwery baz danych. Serwer taki jest więc kolejnym po interfejsie użytkownika elementem współczesnego programu sieciowego. Następnym, chyba najważniejszym, składnikiem jest oprogramowanie działające pomiędzy bazą danych a warstwą interfejsu. To tu znajduje się cała logika aplikacji, odpowiedzialna między innymi za przetwarzanie danych. W przypadku, gdy rolę interfejsu pełnią strony WWW wyświetlane przez przeglądarkę, warstwa środkowa aplikacji wymaga do działania odpowiedniego serwera WWW [4-6]. Natomiast dla programów z interfejsem w postaci zwykłego programu konieczne jest użycie technologii realizującej komunikację pomiędzy rozproszonymi składnikami oprogramowania, np. DCOM lub CORBA [16].

Ze względu na podział na trzy odrębne składniki opisywane tu aplikacje nazywane są *aplikacjami trójwarstwowymi*. Więcej informacji na temat tej architektury można znaleźć w [17-19], a jeden ze sposobów implementacji tego typu aplikacji jest opisany w [18, 19].

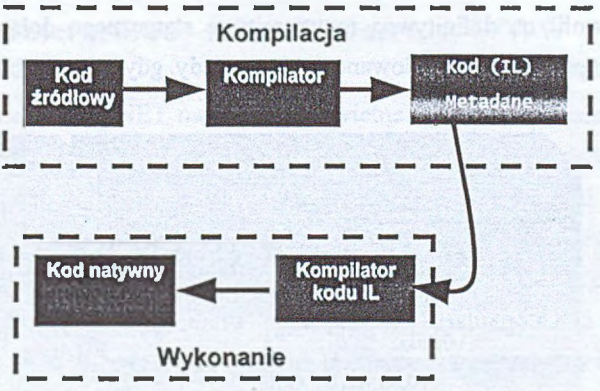
Implementowanie aplikacji trójwarstwowych nie jest rzeczą prostą. Jeżeli wziąć jeszcze pod uwagę stale rosnące wymagania użytkowników i potrzeby zmian wynikające z ciągłego postępu technologicznego oraz konieczność sprawnej obsługi wzrastającej liczby podłączonych komputerów, zadanie to wzrasta do rangi bardzo trudnego. Na szczęście producenci narzędzi dla programistów nie pozostają bezczynni na tym polu. Praktycznie każda licząca się w tej dziedzinie firma ma w swojej ofercie bardziej lub mniej kompleksowe narzędzia wspomagające budowanie oprogramowania wielowarstwowego.

2. Technologia .NET

Jednym z najnowszych narzędzi programistycznych, wspierającym konstruowanie aplikacji trójwarstwowych dla systemu operacyjnego Windows, jest pakiet Visual Studio.NET Microsoftu [1, 2]. Razem z nim producent systemu Windows zaczął wdrażać liczne rozwiązania wspomagające działanie programów rozproszonych, wśród których przede wszystkim należałoby wymienić nową wersję ASP (*ang. Active Server Pages*) – technologii do tworzenia aktywnych stron WWW [1, 4-6] oraz ADO (*ang. ActiveX Data Objects*) – interfejsu do komunikacji z bazami danych [1, 6].

Jakby tego było mało, wprowadzono następcę Windows API (*ang. Application Programming Interface*) – nową platformę uruchomieniową z obiektowym interfejsem,

nazwaną .NET Framework. Jej obecność jest niezbędna do funkcjonowania aplikacji skompilowanych za pomocą Visual Studio.NET. Programy napisane dla tej platformy nie są kompilowane do postaci języka maszynowego procesora, lecz do specjalnego kodu pośredniego, nazywanego IL (*ang. Intermediate Language*). Technika ta jest podobna do zastosowanej w Javie, lecz w tym przypadku aplikacje nie są uruchamiane w takiej postaci. W chwili wywołania programu IL środowisko uruchomieniowe .NET, nazwane przez Microsoft CLR (*ang. Common Language Runtime*) [4], wczytuje kod pośredni i w locie przekompilowuje go na język maszynowy procesora (*tzw. kod natywny – ang. native code*). Rozwiązanie to jest nieco skomplikowane (rys. 1), lecz ma kilka istotnych zalet. Po pierwsze, CLR ma większą kontrolę nad kodem i jest w stanie wykryć fragmenty niepoprawne lub naruszające zasady bezpieczeństwa (dlatego programy te nazywane są zarządzanymi – *ang. managed*). Po drugie łatwiejsze jest tworzenie aplikacji wieloskładnikowych i to złożonych z komponentów napisanych w różnych językach. Ten nowy sposób budowania komponentów, oparty na CLR ma zastąpić stosowaną dotychczas w Windows technologię COM+. Trzeba tu wspomnieć, że pojawiła się wreszcie nadzieja na wyeliminowanie bałaganu z niezarejestrowanymi kontrolkami, zagubionymi bibliotekami typów i nieporządkiem w rejestrze systemowym, a to dzięki temu, że wszystkie informacje potrzebne do używania komponentu są zintegrowane z nim. Tak ulokowane informacje są nazywane metadanymi, a wraz z kodem komponentu tworzą podzespół (*ang. assembly*) [1].



Rys. 1. Uruchamianie programu w CLR
Fig. 1. Execution of application in CLR

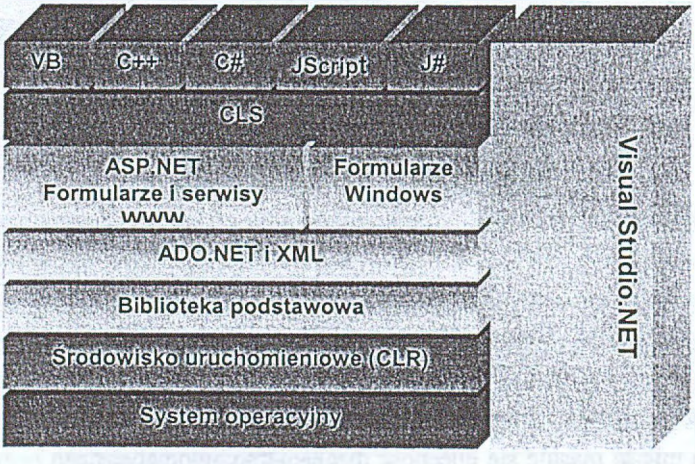
Spośród cech platformy .NET, które należałoby jeszcze wymienić, na jednym z pierwszych miejsc plasuje się obecność mechanizmu automatycznego oczyszczania pamięci (*ang. Garbage Collector*). Dzięki niemu programista jest zwolniony z obowiązku usuwania nieużywanych obiektów i zaalokowanych bloków pamięci. Środowisko uruchomieniowe robi

to automatycznie. Kolejną znaczną innowacją jest zastosowanie metajęzyka XML jako standardowego formatu przechowywania i wymiany danych [1, 2, 15].

Kiedy pojawiają się rozwiązania takie jak platforma .NET, dla autorów oprogramowania istotne jest, aby było możliwe używanie starych komponentów i bibliotek. Opisywane środowisko gwarantuje taką możliwość. Dotyczy to w szczególności komponentów COM, w tym kontrolek ActiveX. Stare komponenty z powodzeniem działają w aplikacjach zarządzanych. Oczywiście, ze względu na różnice w sposobie dostępu do funkcji interfejsów w COM i CLR konieczne jest wygenerowanie kodu adaptacyjnego. Jednak nie stanowi to żadnego problemu, gdyż robi się to za pomocą prostych w obsłudze narzędzi [1-15].

Korzystanie w pisanych aplikacjach ze wszystkich innowacji wprowadzonych w .NET, wymaga wsparcia ze strony używanego języka programowania. Dlatego istniejące języki musiały zostać rozszerzone o nowe konstrukcje składniowe. Powstały też zupełnie nowe, dedykowane specjalnie dla opisywanej platformy. Jednym z wprowadzonych języków jest C# [7-12], uznawany za jeden z podstawowych do pisania programów dla .NET. Natomiast zmianom poddano Visual Basic [14] oraz Visual C++ [13].

Microsoft opublikował zasady, które muszą spełniać języki zgodne z .NET oraz kompilatory dla nich. Specyfikacji tej nadano nazwę CLS (*ang. Common Language Specification*). Dzięki CLS możliwe jest łączenie kodu napisanego w różnych językach i używanie wspólnych narzędzi programistycznych oraz wspólnej biblioteki standardowych klas. Podział kompilacji na dwa etapy (z tekstu źródłowego do IL, a potem z IL do kodu natywnego) pozwolił na definitywną rezygnację ze statycznego dołącznia standardowych bibliotek do kodu programu. Są ładowane dopiero wtedy, gdy są potrzebne [1, 2, 15].



Rys. 2. Platforma .NET
Fig. 2. .NET Framework

3. Dostęp do baz danych – ADO.NET

Platforma .NET zawiera zestaw klas do obsługi baz danych. Klasy te zostały zaprojektowane pod kątem tworzenia wydajnych i skalowalnych aplikacji o architekturze wielowarstowowej. Nadają się zarówno do wykorzystania przy tworzeniu aplikacji internetowych, jak i programów nie korzystających z serwera WWW. Komunikacja z serwerami danych może być relizowana zarówno za pomocą języka SQL jak i XML. ADO.NET zapewnia pełną obsługę rozłącznego trybu dostępu do danych, który polega na przetwarzaniu danych, które nie są połączone ze źródłem [1, 4-6].

Obsługa ADO.NET jest dla programisty dość wygodna i nie nastręczająca nazbyt wiele problemów. Do przechowywania i przetwarzania danych pobranych ze źródła należy wykorzystać obiekt *DataSet*. Wymiana danych ze źródłem odbywa się za pośrednictwem następujących komponentów:

- *OleDbConnection* lub *SqlConnection* - realizują połączenie,
- *OleDbDataAdapter* lub *SqlDataAdapter* - przesyłają dane,
- *OleDbCommand* lub *SqlCommand* – służą do wykonywania poleceń na źródle.

Pisanie kodu do obsługi danych jest wspomagane licznymi narzędziami, w które wyposażone jest Visual Studio, takich jak kreatory i edytory właściwości.

4. Aplikacje Internetowe – ASP.NET i serwisy WWW

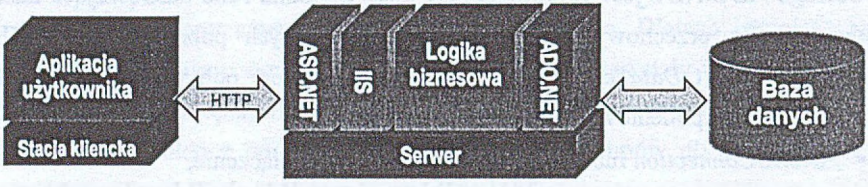
Za pomocą Visual Studio.NET można robić świetne aplikacje trójwarstwowe, które nie wykorzystują serwera WWW. Możliwość wykonywania funkcji komponentów, które znajdują się na zdalnej maszynie, pozwala na wykonanie niewielkim nakładem pracy modułów pełniących rolę serwera aplkacji. Wystarczy jeszcze tylko połączyć się z bazą za pośrednictwem ADO.NET i system o architekturze trójwarstwowej już jest gotowy.

Jednak największa siła .NET tkwi w możliwości wykonywania aplikacji, które wykorzystują serwer WWW i technologię ASP.NET do realizacji warstwy logiki biznesowej oraz przeglądarkę stron internetowych jako powłokę interfejsu użytkownika. Na rysunku 3 przedstawiona jest przykładowa struktura takiego rozwiazania. Udostępnianie stron WWW w Internecie realizuje serwer IIS. Wykonywaniem skryptów, które znajdują się na stronach, zajmuje się środowisko ASP.NET. Strony te, oprócz procedur skrytptowych, mogą zawierać specjalne kontrolki, które oferują dużo większą funkcjonalność niż standardowe elementy HTML. Przetwarzanie tych dodatkowych komoponentów odbywa się po stronie serwera. Do przeglądarki na komputerze użytkownika trafiają zwykłe teksty HTML oraz skrypty

JavaScript, czyli standardowe składniki stron WWW, dzięki czemu po tej stronie nie jest wymagane żadne dodatkowe oprogramowanie [1, 4-6].

Pakiet Visual Studio ma dużo składników ułatwiających projektowanie takich aplikacji:

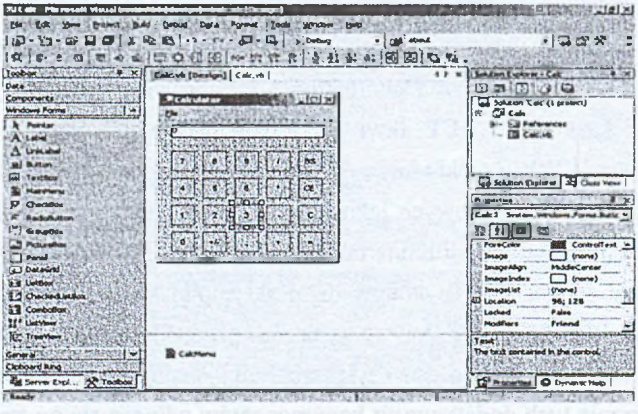
- wizualne projektowanie stron WWW zawierających skrytpty i kontrolki ASP,
- wizualne generowanie kodu do pobierania, edytowania i wyświetlania danych,
- tworzenie dodatkowych kontrolek ASP,
- edytory plików XML i HTML,
- tworzenie serwisów – komponentów, których funkcje można wywoływać poprzez Internet.



Rys. 3. Architektura internetowych aplikacji trójwarstwowych
Fig. 3. Architecture of multi-tier web applications

5. Projektowanie interfejsu użytkownika

Pakiet Visual Studio w wersji 7 (rys. 4) wreszcie doczekał się usprawnień, czyniących z niego narzędzie doskonale nadające się do projektowania i implementowania interfejsu użytkownika w języku innym niż Visual Basic.



Rys. 4. Visual Studio.NET
Fig. 4. Visual Studio.NET

Poza zmianami w postaci dodatkowych narzędzi wizualnych na uwagę zasługuje nowa biblioteka klas. Dzięki wprowadzonym innowacjom pisanie programów za pomocą tego środowiska jest dużo łatwiejsze i mniej czasochłonne.

6. Podsumowanie

Nawet najwięksi przeciwnicy rozwiązań pochodzących od producenta systemu Windows muszą zgodzić się ze stwierdzeniem, że przynajmniej niektóre z nowości wdrażanych w ramach technologii .NET są tym, co jest obecnie potrzebne programistom. Oczywiście nie jest to jedyny tego typu produkt dostępny na rynku. Konkurencja ma do zaoferowania wiele ciekawych alternatyw. Jednak technologia „Giganta z Redmond" wypada stosunkowo korzystnie w porównaniach z innymi produktami. Na pewno .NET jest świetnym rozwiązaniem do tworzenia aplikacji wielowarstwowych, co miał na celu pokazać niniejszy artykuł.

Zamieszczone tu zagadnienia są jedynie skrótową prezentacją niektórych cech technologii .NET. Zapoznanie się z nimi powinno wskazać potencjalne możliwości wykorzystania i ułatwić poszukiwanie dalszych informacji.

LITERATURA

1. Francis B., Ullman C., Short S., Conard J., Ramachandran R., Schenken J., Harvey B., Hollis B., Glynn J., Dengler P.: Wprowadzenie do .NET. MIKOM, Warszawa 2002.
2. Platt D.: Microsoft .NET – podstawy. RM, Warszawa 2001.
3. Burton K.: NET CLR. Księga eksperta. Helion, Gliwice 2003.
4. Worley S.: ASP.NET. Vademecum profesjonalisty. Helion, Gliwice 2003.
5. Payne C.: ASP.NET dla każdego. Helion, Gliwice 2002.
6. Esposito D.: Tworzenie aplikacji za pomocą ASP.NET oraz ADO.NET. RM, Warszawa 2002.
7. Visual C# .NET. Encyklopedia. Microsoft Press, Wydanie polskie - Helion, Gliwice 2003.
8. Mueller J. P.: Visual C#.NET. Developer's Handbook. SYBEX, San Francisco 2002.
9. Gunnerson E.: Programowanie w języku C#. MIKOM, Warszawa 2001.
10. Troelsen A.: Język C# i Platforma .NET. MIKOM, Warszawa 2002.
11. Sharp J., Jagger J.: Microsoft Visual C#.NET. RM, Warszawa 2002.

12. Microsoft C# - specyfikacja języka. Microsoft Press, Wydanie polskie - RM, Warszawa 2001.
13. Templan J., Olsen A.: Microsoft Visual C++.NET. RM, Warszawa 2002.
14. Visual Basic .NET. Encyklopedia. Microsoft Press, Wydanie polskie - Helion, Gliwice 2003.
15. Microsoft Developers Network (MSDN). Dokumentacja w postaci elektronicznej, http://msdn.microsoft.com.
16. Eddon G., Eddon H.: Inside Distributed COM. Microsoft Press, Redmond 1998.
17. N-tier Computing Architecture. Artykuł zamieszczony w serwisie internetowym Delaware.gov, http://www.state.de.us/ois/arch/n-tier.html.
18. Mittmann M.: Projekt i realizacja mechanizmu kontroli dostępu do danych w programie KS-SWD. Praca dyplomowa magisterska, Instytut Informatyki, Politechnika Śląska 2001.
19. Mittmann M.: Wielowarstwowa architektura dostępu do danych z programowalnymi widokami. Studia Informatica, Vol. 23, no. 3 (50), Gliwice 2002, s. 347-353.

Recenzent: Prof. dr hab. inż. Tadeusz Wieczorek

Wpłynęło do Redakcji 9 kwietnia 2003 r.

Abstract

The paper presents the one of newest technologies, which supports creating and developing the multi-tier applications. There were explained some aspects of creating each of tiers with .NET technology: communication with database, business logic and user interface. Figures 1 and 2 present the .NET Framework. Figure 3 shows architecture of typical multi-tier web application based on .NET.

Adres

Marek MITTMANN: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, mittmann@star.iinf.polsl.gliwice.pl .