STUDIA INFORMATICA 2002
Volume 23 Number 2B (49)

Michał ŚWIDERSKI
Politechnika Śląska, Instytut Informatyki

KONCEPCJA OBIEKTÓW SAMODZIELNIE KOMUNIKUJĄCYCH SIĘ Z BAZAMI DANYCH, OSADZANYCH W STRONACH WWW*)

Streszczenie. Artykuł omawia popularne implementacje fizyczne logicznego modelu trójwarstwowego oraz jedną z możliwych ich modyfikacji. Przedstawiona jest także aplikacja wykorzystująca zaproponowaną koncepcję.

CONCEPT OF OBJECTS INDIVIDUALLY ACCESSING DATABASE, EMBEDDED IN HTML PAGES

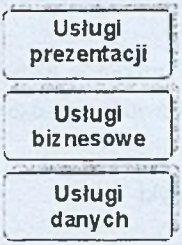
Summary. This article describes the most popular implementations of logical three-tier model and one of the possible modifications. There is also presented an application built upon the proposed implementation.

1. Wstęp

Jednym z głównych elementów projektowania aplikacji jest określenie architektury systemu, która definiuje, za jaką część funkcjonalności odpowiada dana część aplikacji oraz jak części aplikacji współdziałają ze sobą. Niniejsza praca skupia się na implementacjach fizycznych logicznego modelu trójwarstwowego.

Trójwarstwowy model logiczny dzieli aplikację na trzy logiczne komponenty [1].

*) Praca finansowana z funduszu Badań Statutowych Instytutu Informatyki w roku 2002



Rys. 1. Logiczny model trójwarstwowy
Fig. 1. Logical three-tier model

Usługi danych - łączą rekordy i utrzymują integralność bazy danych – np. nakładają ograniczenia na wartości danych wprowadzanych do tabel i wymuszają zachowanie relacji pomiędzy tabelami.

Usługi biznesowe - stosują reguły i logikę biznesową – np. dodają zgłoszenia do bazy danych, przeprowadzają autoryzację i nadają uprawnienia w systemie.

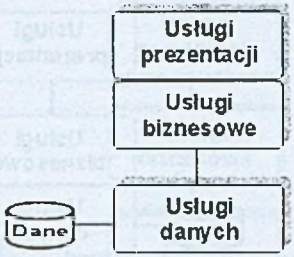
Usługi prezentacji – implementują interfejs użytkownika i obsługują wprowadzane przez niego dane.

2. Aplikacje trójwarstwowe

Model logiczny nie określa jednoznacznie implementacji fizycznej aplikacji, istnieje bowiem wiele sposobów implementacji logicznego modelu trójwarstwowego na fizycznych maszynach. Poniżej zostaną krótko scharakteryzowane najpopularniejsze implementacje fizyczne.

2.1. Fizyczna implementacja dwuwarstwowa z "grubym klientem" (fat client)

Częstą metodą tworzenia aplikacji jest fizyczna implementacja dwuwarstwowa z grubym klientem, gdzie zarówno usługi biznesowe, jak i prezentacji pracują po stronie klienta. W takiej implementacji serwer pracuje jedynie jako serwer bazodanowy.



Rys. 2. Implementacja dwuwarstwowa (gruby klient)
Fig. 2. Two-tier implementation (fat client)

By rozdzielić część biznesową od kodu prezentacji w samym kliencie, można umieścić kod odpowiedzialny za reguły biznesowe w odrębnym obiekcie COM lub klasie Java. Jeśli interfejs użytkownika i obiekt biznesowy pracują na komputerze klienta, jest to wciąż implementacja dwuwarstwowa. Rozdzielenie kodu umożliwi jednak powtórne użycie obiektu w innych zastosowaniach, wygodniejszą pielęgnację systemu oraz umożliwi łatwe przejście do trójwarstwowego modelu fizycznego opisanego poniżej.

Zalety rozwiązania:

- Narzędzia do tworzenia tego typu implementacji są bardzo zaawansowane i umożliwiają łatwe tworzenie tego typu aplikacji.
- Często sytuacją pożądaną jest, by dane związane z sesją były przechowywane po stronie klienta. Jest to sytuacja domyślana w tej implementacji.

Wady rozwiązania:

- Umieszczenie reguł biznesowych po stronie klienta zazwyczaj oznacza wzmożony ruch w sieci, ponieważ dane muszą być przesłane do klienta, by podjąć decyzje biznesowe.

2.2. Fizyczna implementacja dwuwarstwowa z "grubym serwerem" (fat server)

W fizycznej implementacji dwuwarstwowej z grubym serwerem logika biznesowa i usługi danych są zaimplementowane na serwerze bazodanowym. W tej implementacji logika biznesowa jest zazwyczaj tworzona jako procedury (ang. stored procedures) i wyzwalacze (ang. triggers) wewnątrz bazy danych. Wiele aplikacji pracujących w przemyśle korzysta z podobnego scenariusza.

Rys. 3. Implementacja dwuwarstwowa (gruby serwer)
Fig. 3. Two-tier implementation (fat server)

Zalety rozwiązania:

- Główną zaletą takiej implementacji jest wydajność. Logika biznesowa pracuje w tej samej przestrzeni procesu co kod mający dostęp do danych, więc dane nie muszą być ani przenoszone, ani kopiowane, co sprawia, że ruch w sieci pomiędzy klientem i serwerem jest zmniejszony.

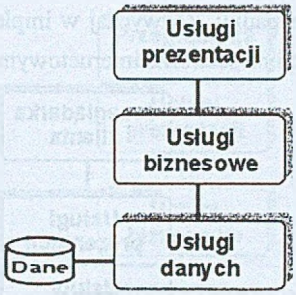
Wady rozwiązania:

- Główną wadą jest ograniczony zestaw narzędzi do tworzenia takich aplikacji, ponieważ procedury zazwyczaj mogą być pisane tylko w języku obsługiwanym przez bazę danych.

2.3. Fizyczna implementacja trójwarstwowa

Fizyczna implementacja trójwarstwowa, zwyczajowo określana jako "model trójwarstwowy", często traktowana jest błędnie jako jedyna implementacja fizyczna logicznego modelu trójwarstwowego.

W tej implementacji logika biznesowa pracuje w oddzielnym procesie, który może być skonfigurowany tak, by pracował na serwerze bazodanowym lub na innym serwerze. Tym, co odróżnia fizyczną implementację trójwarstwową od poprzednich implementacji, jest fakt wykonywania usługa danych, biznesowych i prezentacji przez osobne procesy pracujące najczęściej na różnych komputerach. Wiele poważnych aplikacji finansowych jest zbudowanych w ten sposób.



Rys. 4. Implementacja trójwarstwowa
Fig. 4. Three-tier implementation

Zalety rozwiązania:

- Niezależność aplikacji od bazy danych. Większość fizycznych implementacji trójwarstwowych korzysta z kilku baz danych wykorzystując tylko standardowe komendy SQL.
- Niektóre wersje implementacji oferują niezależność od języka programowania.
- W niektórych przypadkach fizyczna implementacja trójwarstwowa jest bardziej skalowalna niż inne implementacje. Jeśli kod logiki biznesowej bardzo obciąża procesor lub pamięć fizyczną, można umieścić te procesy na jednym lub wielu serwerach oddzielnych od baz danych, tak by zmniejszyć obciążenie serwera. Potencjalny wzrost skalowalności zmniejszany jest jednak przez dodatkowy koszt przesyłania danych do usług biznesowych.
- Istnieje możliwość dostępu do wielu baz danych zawierających tylko częściowe dane, jednakże takie rozwiązanie może wprowadzić ogromne komplikacje w obsłudze danych i nie jest często stosowane w przemyśle.

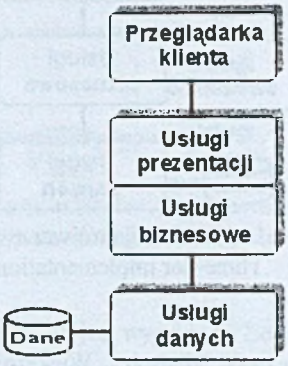
Wady rozwiązania:

- Aplikacje tego typu zazwyczaj wymagają więcej obsługi niż poprzednie implementacje.
- Zazwyczaj wartość współczynnika ceny do wydajności aplikacji jest gorsza od wartości współczynnika dla implementacji dwuwarstwowej z wykorzystaniem grubego serwera.

2.4. Implementacja internetowa

Internet wprowadził nowy zwrot w logicznym modelu trójwarstwowym: możliwość rozdzielenia usług prezentacji na przeglądarkę klienta i serwera internetowego, który jest właściwie odpowiedzialny za formatowanie stron, które ogląda użytkownik. Przeglądarka jest odpowiedzialna za wyświetlanie tych stron i ewentualne pobieranie dodatkowego kodu potrzebnego stronie. Dobór miejsca umieszczenia usług biznesowych podlega takim samym

zasadom jak w poprzednim przypadku. Zazwyczaj w implementacji internetowej umieszcza się usługi biznesowe i prezentacji na serwerze internetowym.



Rys. 5. Implementacja internetowa
Fig. 5. Internet implementation

Zalety rozwiązania:

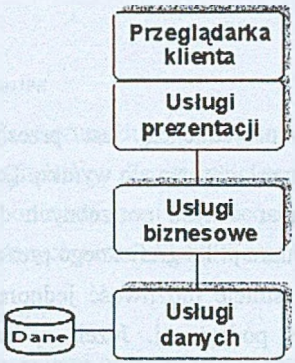
- Każdy klient wyposażony w standardową przeglądarkę może korzystać z aplikacji, ponieważ cała wymagana od klienta funkcjonalność dostarczana jest przez standardową przeglądarkę.
- Łatwość zarządzania aplikacją. Zmiana oprogramowania na serwerze internetowym automatycznie pociąga za sobą zmianę oprogramowania na wszystkich klientach. Zarządzanie kodem na kilku serwerach jest łatwiejsze niż zarządzanie kodem na wielu klientach.

Wady rozwiązania:

- Podstawowe implementacje internetowe nie są aplikacjami wysokowydajnymi. (Można jednak zwiększyć wydajność takich aplikacji umieszczając logikę biznesową w procedurach umieszczanych w bazie danych)

3. Obiekty samodzielnie komunikujące się z bazą danych

Implementacja internetowa opisana powyżej jest bardzo uniwersalna i szeroko stosowana, jednakże dla szczególnej klasy zastosowań może okazać się, że wprowadzenie pewnych modyfikacji może zwiększyć wydajność aplikacji i jej niezawodność. Jedną z możliwych modyfikacji jest połączenie modelu internetowego z modelem grubego klienta.



Rys. 6. Połączenie implementacji internetowej z grubym klientem
Fig. 6. Combination of internet and fat client implementations

W proponowanej implementacji przeglądarka nie służy już tylko jako aplikacja prezentująca wynik pracy usług prezentacji pracujących na serwerze internetowym, ale przejmuje na siebie część lub całość pracy związanej z przetwarzaniem danych otrzymanych od usług biznesowych. Sytuację taką można osiągnąć poprzez wstawienie w treść strony HTML skryptu lub skompilowanego obiektu klienta, który potrafiłby implementować interfejs, obsługiwać żądania użytkownika oraz generować wyniki przetwarzania danych.

Wybór między zastosowaniem skryptu lub osadzanego obiektu zależy od zadań spełnianych przez konkretną aplikację, jednak w bardziej zaawansowanych przypadkach obiekt wykazuje większą stabilność, większą wydajność oraz większe możliwości funkcjonalne, dlatego właśnie ten przypadek będzie dalej omawiany [2].

Obiekt klienta powinien funkcjonować na podstawie danych zawartych bądź w treści strony, bądź też pobranych z obiektu biznesowego. Druga z możliwości wydaje się jednak zasadniejsza, ponieważ zakładając, że obiekt korzysta z danych zawartych w treści strony, zakładamy jednocześnie, że strona ta musi być formatowana po stronie serwera internetowego, co wymusza z kolei rozbudowę warstwy prezentacji na tymże serwerze.

W świetle poczynionych założeń koncepcja obiektów samodzielnie komunikujących się z bazami danych, osadzanych w stronach WWW, przedstawia się następująco:

- Aplikacja powinna korzystać z ogólnie stosowanych technologii internetowych i współpracować z popularnymi przeglądarkami.
- W treści strony umieszczony jest obiekt, który na podstawie danych pobranych bezpośrednio z obiektu biznesowego obsługuje interfejs użytkownika oraz modyfikuje swoje działanie.

3.1. Zalety rozwiązania

3.1.1. Wydajność aplikacji

- Dla pewnej klasy zadań korzystniejsze jest przesłanie danych potrzebnych do wygenerowania wyniku niż przesłanie samego wyniku. Dzieje się tak często w przypadku wykresów, dla których rozmiar danych potrzebnych do jego wykreślenia może być kilkakrotnie mniejszy od rozmiaru pliku graficznego prezentującego wykres.
- W proponowanej koncepcji istnieje możliwość jednorazowego załadowania danych i zapamiętania ich w pamięci podręcznej. Jeżeli aplikacja byłaby zmuszona pobrać dodatkowe dane, może pobrać tylko brakujące, a nie wszystkie, jak w klasycznym przypadku, co w znaczący sposób może zmniejszyć ruch w sieci.
- Skompilowany obiekt pracujący w przeglądarce może pracować wydajniej niż analogiczny kod umieszczany zazwyczaj jako skrypt po stronie serwera internetowego.
- Przeniesienie usług prezentacji z serwera internetowego do przeglądarki może w znaczący sposób odciążyć serwer internetowy.

3.1.2 Niezawodność i bezpieczeństwo

- Skompilowany obiekt podlega awariom znacznie rzadziej niż interpretowany skrypt [3].
- Technologie obiektów osadzanych umożliwiają podpisywanie kodu, dzięki czemu użytkownik ufający danemu producentowi oprogramowania może być pewien, że aplikacja będzie działać prawidłowo.

3.1.3. Zaawansowany interfejs użytkownika

- Dzięki umieszczeniu w ciele strony skompilowanego obiektu możliwe jest tworzenie interfejsu użytkownika o funkcjonalności odpowiadającej aplikacjom systemowym.
- Przechowywanie danych w pamięci podręcznej umożliwia zapis danych wejściowych i wynikowych na komputerze klienta w różnej formie.

3.1.4. Możliwość rozbudowy

- Proponowana koncepcja może być dowolnie łączona z innymi implementacjami modelu trójwarstwowego. Obok siebie może więc funkcjonować klasyczny model internetowy oraz proponowany przez autora.

3.2. Wady rozwiązania

3.2.1. Wymagania programowe

Proponowana implementacja narzuca duże wymagania programowe zarówno na serwer internetowy, jak i na przeglądarkę. Zazwyczaj podstawowa konfiguracja serwera internetowego nie wystarcza, ponieważ musi on mieć możliwość odpowiedzi na żądania obiektu i przesyłania danych żądanych przez obiekt klienta jako strumień bitowy.

3.2.2. Ograniczona przenośność

W związku z dużymi wymaganiami programowymi przenośność podobnego rozwiązania może być ograniczona.

3.2.3. Utrudnione zarządzanie aplikacją

Każdorazowa zmiana w kodzie obiektu pociąga za sobą konieczność pobrania nowej jego wersji z serwera internetowego.

4. Praktyczna realizacja koncepcji

Do implementacji proponowanej koncepcji użyto serwera internetowego IIS w wersji 5.0 pracującego pod kontrolą systemu Windows 2000 Professional. Obiekt osadzany w ciele strony został zaimplementowany jako kontrolka ActiveX pobierana przez przeglądarkę z serwera internetowego. Zastosowanie technologii ActiveX ogranicza uniwersalność rozwiązania, ponieważ nie wszystkie przeglądarki internetowe potrafią ją obsłużyć, jednak zapewnia stabilność aplikacji oraz dużą wydajność. Spowodowane jest to faktem, że kod kontrolki ActiveX kompilowany jest do postaci binarnej, podczas gdy bardziej uniwersalne rozwiązania stosują kod w postaci bajtowej, co niekorzystnie wpływa na wydajność i stabilność tych rozwiązań. Technologia dostępu do bazy danych zostanie opisana poniżej.

4.1. Komunikacja obiektu z bazą danych

Do komunikacji obiektu ActiveX klienta z bazą danych wykorzystano mechanizm Remote Data Service (RDS), który wykorzystuje zbiór komponentów dostarczanych z biblioteką Microsoft Data Access Components (MDAC) [4].

RDS pozwala aplikacjom na dostęp do źródeł OLE DB, pracujących na odległych komputerach. Użycie RDS może być przedstawione na przykładzie:

A. Użytkownik wprowadza adres strony startowej aplikacji.

B. Żądanie użytkownika jest tłumaczone na żądanie HTTP i wysłane przez przeglądarkę do serwera WWW.

C. Użytkownik otrzymuje stronę HTML z umieszczoną kontrolką obiektu, która zawiera kontrolkę RDS Data Control.

D. Użytkownik za pomocą interfejsu kontrolki obiektu określa rodzaj zadania, które ma być wykonane, kod obiektu decyduje, jakie dane należy pobrać z serwera, a RDS Data Control komunikuje się z serwerem poprzez standardowe protokoły, takie jak: HTTP i DCOM, zgłaszając żądanie.

E. Komponent RDS umieszczony na serwerze sprawdza uprawnienia i ewentualnie wykonuje żądanie użytkownika i tworzy jako wynik obiekt ADO Recordset, który jest zmieniany w strumień binarny i wysyłany do klienta.

F. Na komputerze klienta obiekt Recordset jest odtwarzany wewnątrz komponentu RDS. Kontrolka obiektu operuje na standardowych strukturach danych i generuje wynik.

G. Ewentualne zmiany w obiekcie Recordset mogą być przesłane z powrotem do bazy danych na serwerze poprzez kontrolkę RDS.

Technicznie proces komunikacji wygląda to następująco:

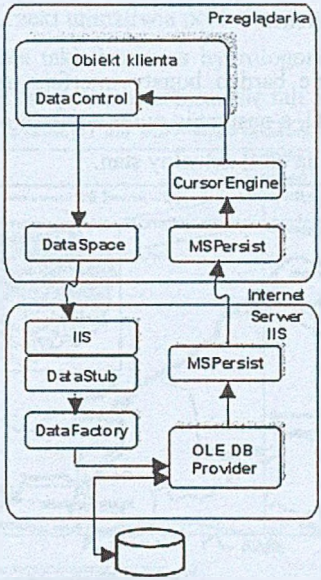
A. Żądanie generowane przez RDS.DataControl konwertowane jest na żądanie HTTP przez obiekt RDS.DataSpace i wysyłane do serwera.

B. Na serwerze komponent RDS ADISAPI mapuje żądania HTTP na metody wywołujące i sterujące obiektem RDS.DataFactory.

C. RDS.DataFactory tworzy obiekt Recordset i pobiera do niego dane poprzez bezpośrednie odwołania do źródła OLE DB.

D. Pozyskany obiekt Recordset jest szeregowany w strumień binarny poprzez OLE DB Persistence provider (MSPersist).

E. Ze strumienia w obiekcie Cursor Engine odtwarzany jest obiekt Recordset, który następnie przesyłany jest do RDS.DataControl.



Rys. 7. Komunikacji z bazą danych za pomocą RDS
Fig. 7. Communication with data base using RDS

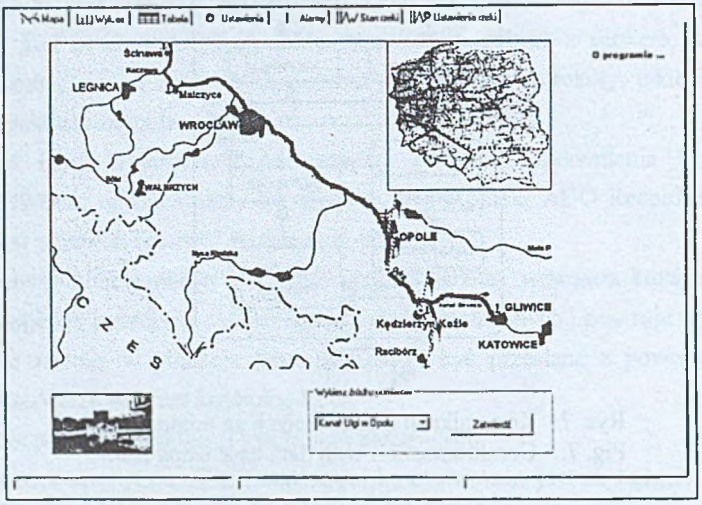
4.2. Zadania obiektu klienta

Obiekt klienta zbudowany zgodnie z zaproponowaną koncepcją wchodzi w skład systemu monitorującego stan rzeki Odry. Do zadań obiektu należy:

- umożliwienie wyboru punktu pomiarowego na hierarchicznym menu map Polski oraz udostępnienie w formie graficznej ostatnich pomiarów na tejże mapie;
- umożliwienie określenia, z jakiego okresu interesują użytkownika pomiary;
- prezentacja pomiarów w formie:
  - konfigurowalnych wykresów dwu- i trójwymiarowych,
  - tabelarycznej,
  - wydruków na drukarce;
- zapis pomiarów na dysku klienta;
- wykrywanie stanów awaryjnych w punktach pomiarowych;
- wykreślanie profilu rzeki w danym momencie czasowym.

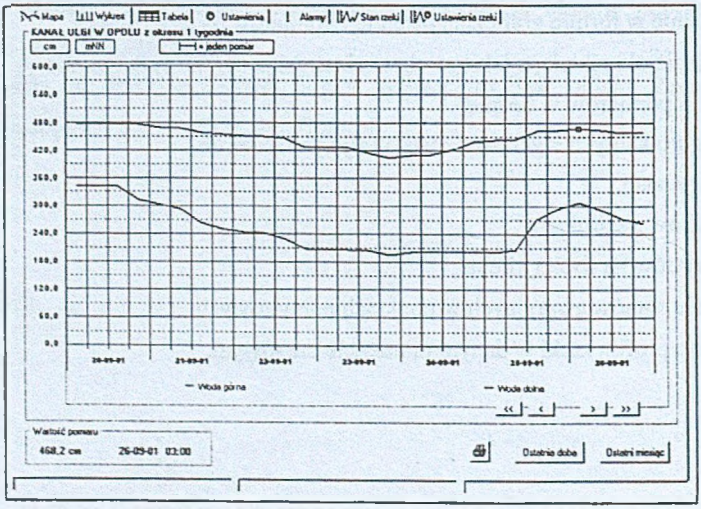
4.3. Prezentacja obiektu klienta

Obiekt klienta implementuje bardzo bogaty interfejs użytkownika. Zakładka startowa umożliwia wybór obszaru Polski, a następnie punktu pomiarowego. Z punktem pomiarowym stowarzyszona jest jego fotografia oraz aktualny stan.



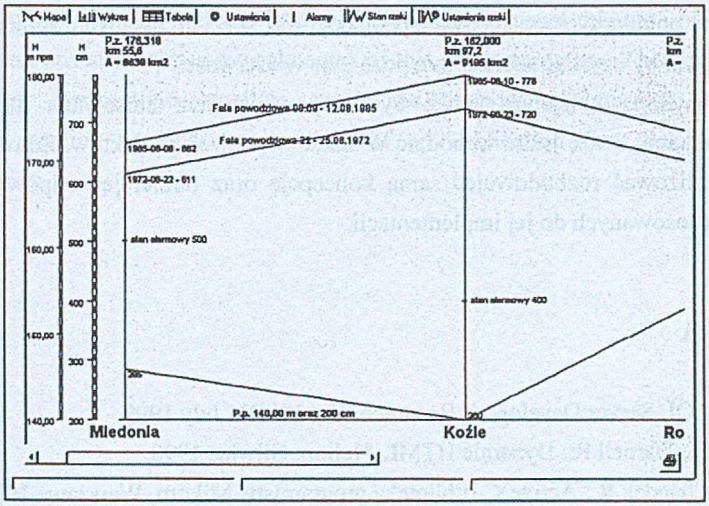
Rys. 8. Zakładka startowa obiektu klienta
Fig. 8. Client's object startup tab

Zakładka prezentująca pomiary umożliwia obserwację pomiarów z zadanego przez użytkownika okresu. Dostępne są funkcje skalowania wykresu, wycinania serii, wyboru sposobu prezentacji na wykresie i wiele innych.



Rys. 9. Zakładka wykresu obiektu klienta
Fig. 9. Client's object chart tab

Zakładka prezentująca profil rzeki umożliwia porównanie całościowego stanu rzeki z jego stanem podczas powodzi. Wykres taki dostarcza hydrologom informacji ogólnych (ryzyko powodzi) i szczegółowych (szybkość przemieszczania się fali powodziowej).



Rys. 10. Zakładka profilu rzeki
Fig. 10. River profile tab

4.4. Ocena implementacji

Zaimplementowany system działa zgodnie z oczekiwaniami. Do szczególnie wyraźnych zalet należą:

- Stabilność – obiekt klienta działa bez zarzutu mimo bogatego interfejsu i dużej ilości obliczeń przez niego przeprowadzanych.
- Niskie obciążenie sieci:
  - obiekt przechowuje dane w pamięci podręcznej, przez co nie musi ich wielokrotnie pobierać z serwera,
  - w wielu sytuacjach potwierdza się teza, że oszczędniej jest pobrać dane do wykreślenia wykresu niż gotowy wykres w formie pliku graficznego. Dla rzeczywistego profilu rzeki obiekt klienta pobiera **4,3 kB** danych, natomiast odpowiadający mu plik graficzny w formacie gif zajmuje **15,8 kB**, zatem proporcja wynosi ponad **3,6** na korzyść proponowanego rozwiązania.

Do wad należą:

- Wysokie wymagania programowe, tj: jako serwer internetowy może być wykorzystywany jedynie Microsoft IIS, jako przeglądarka jedynie Internet Explorer.
- Pokaźnych rozmiarów obiekt klienta, który należy pobrać z serwera internetowego.

5. Wnioski

Standardowa implementacja modelu trójwarstwowego dla sieci internet jest wystarczająca dla większości popularnych zastosowań, jednakże dla specjalizowanych aplikacji warto ją modyfikować, tak by uzyskać szczególnie pożądane właściwości.

Przedstawiona koncepcja może być z powodzeniem stosowana do budowania wydajnych aplikacji, może także wchodzić w skład większych projektów. Zauważone wady można zneutralizować rozbudowując samą koncepcję oraz dobierając odpowiedni zestaw technologii zastosowanych do jej implementacji.

LITERATURA

1. Microsoft SQL Server Developer's Resource Kit, MSDN July 1999.
2. Campbell B., Darnell R.: Dynamic HTML. Helion, Gliwice 1998.
3. Lalani S., Chandak R.: ActiveX. Biblioteka programisty. Mikom, Warszawa 1997.
4. Microsoft Data Access Components 2.5 SDK Beta - Technical Articles, MSDN July 1999.

Recenzent: Dr inż. Arkadiusz Sochan

Wpłynęło do Redakcji 8 kwietnia 2002 r.

Abstract

This paper describes different implementations of the logical three-tier model giving both advantages and disadvantages of each option.

At first, it introduces a physical two-tier implementation with fat clients, where the business logic and presentation services run on the client side (see Fig.2). A primary advantage of this fat client implementation is that the tools that support it are powerful and well established. The second implementation is that with a fat server, where business logic and presentation services are deployed from the server database (see Fig.3). In this implementation, business logic is mainly written as stored procedures and triggers within the database. The third described implementation is the physical three-tier implementation (see Fig.4) that offers an advantage of database independence. Then, we move to Internet

implementation (see Fig.5) which a key advantage is that anybody who has a browser client can access these applications.

From a combination of the Internet and fat client implementations we derive a new one (see Fig.6), which we take a closer look at. The paper discusses this case more thoroughly, showing numerous strengths and weaknesses of this implementation. The key advantages of the concept are: performance, stability, security and advanced user interface.

The last section presents an example of an application built upon the proposed implementation.