STUDIA INFORMATICA 2001
Volume 22 Number 1 (43)

Henryk KRAWCZYK, Jerzy PROFICZ, Mariusz MATUSZEK
Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki

SERWERY APLIKACJI - NOWA FORMA DOSTARCZANIA USŁUG SIECIOWYCH*

Streszczenie. W pracy podano główne problemy projektowania serwerów aplikacji zapewniających właściwą realizację i utrzymanie różnych usług w środowisku sieciowym w oparciu o standard J2EE. Zaprezentowano architekturę oraz poszczególne komponenty takiego serwera oraz przedyskutowano metodologie tworzenia użytkowych aplikacji w takim środowisku.

APPLICATION SERVERS - A NEW RELIVERY FORM OF NETWORKED SERVICES

Summary. The paper gives basic problems of application server design, in order to achieve proper execution and maintenance of various user services in a networked environment. Elements of J2EE standards are described and the suitable server architecture and its components are presented. Developing methodology of networked user application is also discussed.

1. Wprowadzenie

Jednym z problemów prześladujących administratorów rozbudowanych korporacyjnych sieci komputerowych jest żądanie zapewnienia spójnego i jednolitego środowiska pracy na wszystkich komputerach w sieci. Dotychczasowa polityka instalowania aplikacji użytkowych na stacjach roboczych doprowadziła do sytuacji, w której dowolna zmiana w stosowanym oprogramowaniu pociąga za sobą konieczność aktualizacji dziesiątek, a nierzadko i setek komputerów, przy czym nie zawsze wystarczy wymiana oprogramowania, często w parze

* Pracę wykonano w ramach grantu KBN 8T11C 001 17.

idzie konieczność rozbudowy sprzętu. Aby zapobiec takiemu scenariuszowi i ułatwić prace administracyjne wymyślono koncepcję tzw. "cienkiego klienta" (ang. thin client). Koncepcja ta polega na zapewnieniu dostępu do wszystkich aplikacji użytkowych firmy przez niedrogie komputery wyposażone tylko w przeglądarkę internetową i niezbędne oprogramowanie systemowe. Same aplikacje zostały natomiast przeniesione na jeden bądź kilka komputerów dużej mocy, łatwiejszych w administrowaniu [3].

Poważnym kłopotem w takim rozwiązaniu jest dobór formatu danych. Przeglądarki internetowe zazwyczaj nie "rozumieją" danych wysyłanych przez aplikacje używane w firmach (systemy bazodanowe, SAP, itp.). Pojawiła się konieczność wprowadzenia warstwy pośredniej (ang. middleware), zapewniającej dwukierunkowe tłumaczenie danych pomiędzy aplikacjami koroporacyjnymi a przeglądarkami. Jednym z pomysłów na takie oprogramowanie są serwery aplikacyjne [4].

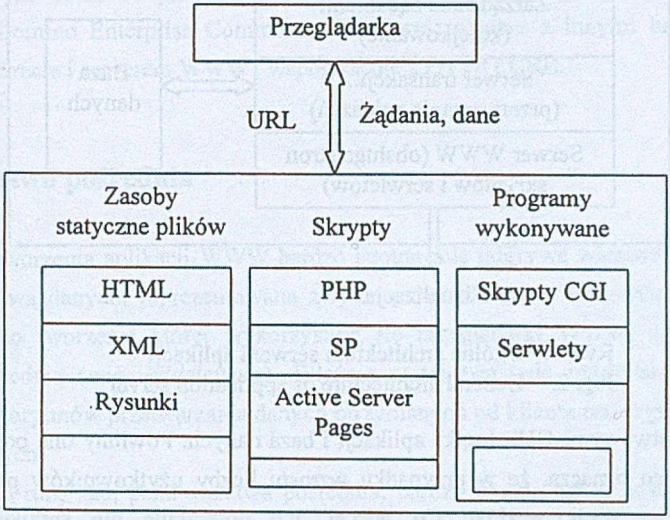
Serwer aplikacyjny to środowisko pracujące na komputerach o dość dużej mocy obliczeniowej, wykonujący wszelkie operacje konieczne do współpracy pomiędzy przeglądarką internetową na komputerze klienckim a aplikacją korporacyjną lub innym źródłem danych. Umożliwia on klientowi, wyposażonemu jedynie w przeglądarkę, dokonywanie np. operacji na bazach danych firmy. Zadaniem serwera aplikacyjnego jest pobieranie danych z różnych źródeł (bazy danych, systemy eksperckie, etc.) i tłumaczenie ich na postać dynamicznej publikacji sieciowej w języku HTML, jak również tłumaczenie informacji i komend użytkownika przekazanych przez przeglądarkę HTML na postać zrozumiałą dla warstwy danych.

Serwer aplikacji nie jest pojedynczym programem, lecz zestawem komponentów, które pracując razem tworzą środowisko. Burzliwy rozwój tej dziedziny rynku oprogramowania oraz niezależne prace prowadzone przez firmy, takie jak: IBM, Sun, Netscape, Oracle czy Microsoft doprowadziły do powstania wielu różniących się znacznie technik realizacji serwerów aplikacyjnych. Do najpopularniejszych należą: PHP, JSP, ASP, Enterprise Java Beans, servlety, CORBA, DCOM [1, 2].

W pracy przedstawiono zarówno architekturę serwerów aplikacji i ich podstawowe własności, jak i metodę opracowywania aplikacji użytkowych pracujących na tego typu serwerach. Rozważania oparto na własnych doświadczeniach w projektowaniu i implementowaniu aplikacji webowych jak i organizacji serwera aplikacji dla potrzeb naukowo - badawczych katedry [6, 7].

2. Architektura serwerów aplikacji

Chcąc opisać architekturę serwerów aplikacji przedstawimy najpierw architekturę serwera WWW (rys. 1). Każdy serwer WWW współpracuje z przeglądarką, która spełniając żądanie użytkownika generuje odpowiednie żądanie dostarczenia strony, której adres określa URL. Może to być również żądanie opisane w PHP. Serwer WWW wybiera odpowiednią stronę

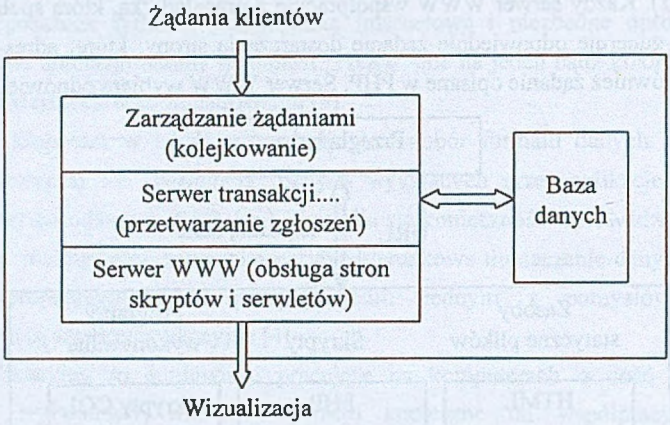


Rys. 1. Architektura Serwera WWW
Fig. 1. Architecture of WWW server

(plik HTML) lub wykonuje odpowiednie skrypty generujące dokumenty kod HTML. Skrypty podczas wykonania generują (kreują) odpowiednie procesy. W dokumencie HTML może być wstawiony kod Javy (Java Serwer Pages, PSP, Java Script), który jest następnie kompilowany do odpowiednich serwletów. Stanowią one alternatywę do skryptów. Zbudowane są z komponentów i dostarczają obiekty i metody. Tak więc obliczenia na serwerze WWW realizowane są jako model klient - serwer.

Serwer aplikacji jest zestawem odpowiednich komponentów, które kreują środowisko obliczeniowe. Jego architekturę przedstawia rys. 2. Każdy taki serwer składa się z serwera WWW, serwera transakcyjnego oraz komponentu zarządzającego nadchodzącymi żądaniami od klientów. Ten ostatni komponent ma za zadanie odpowiednio kolejkować zgłoszenia i od jego strategii działania w dużej mierze zależy wydajność przetwarzania informacji. Komponent transakcyjny przyjmuje zgłoszenia i przetwarza je, pobiera z bazy danych niezbędne dane i przygotowuje do wyświetlenia wyniki przetwarzania. Serwer WWW przejmuje te wyniki i prezentuje je użytkownikowi. W serwerze aplikacji wbudowuje się

również mechanizmy do obsługi zdalnego administrowania i zdalnej pracy. Umożliwia to budowanie obiektów, które są niezależne od platformy oraz mogą być przenoszone między węzłami oraz zdalnie wykonywane. Zakłada się, że serwery aplikacji realizują zadania w



Rys. 2. Ogólna architektura serwera aplikacji
Fig. 2. General architecture of application server

modelu trójwarstwowym: GUI, logika aplikacji i baza danych. Powinny one posiadać cechy skalowalności, co oznacza, że w przypadku wzrostu liczby użytkowników pracę serwera rozdziela się na wiele węzłów. Co więcej, ich dołączenie nie sprawia kłopotów (przezroczystość).

Przykładami serwerów aplikacji są:

1. WebSphere Application Server (IBM), który realizuje aplikacje B2B, wykorzystuje serwlety Javy, obiekty Java Beans, Java Server Pages oraz Enterprise Java Beans, obsługuje dokumenty w standardzie XML, zapewnia przetwarzanie transakcyjne (DBZ) i współpracuje z innymi aplikacjami (SHP), umożliwia określanie reguł dostępu przez administratora.
2. NetDynamic Application Server (Sun Microsystem), wykorzystujący aplikacje Javy, które wykonywane są na 1 lub wielu węzłach i oparte na standardzie CORBY, obsługuje EJB, umożliwia tworzenie wielu konfiguracji i dodawanie nowych komponentów (PAC), integruje te komponenty (JOF - Java Object Framewwork), dostarcza gotowych obiektów do wykorzystania przez programistów, udostępnia cienki serwer (CGI) do komunikacji z serwerem WWW, współpracuje z bazą Oracle, Informix, Sybase, MSSQL, zapewnia ochronę stron WWW, ACL (Access Control List) oraz udostępnia regularny dostęp do bazy danych.

3. Lotus Domino Application Server (wykorzystywany między innymi przez firmę PROKOM), wspierajacy technologie MIME, SMTP, LPAP, Corba, IIOP, Java, Java Script, DOM (Domino Object Model), współpracuje z Microsoft Internet Information Server, dostarcza obiekty obecne po stronie serwera. Umożliwia tworzenie aplikacji (Lotus Domino Designer), programowanie w środowisku WYSIWYG, w języku skryptowym Lotus Script, zapewnia połączenie z bazami Lotusa za pomocą pakietu DECS (Domino Enterprise Communication Services) oraz z innymi bazami danych: Sybase, Oracle i serverem WWW , współpracuje z HTTP i URL.

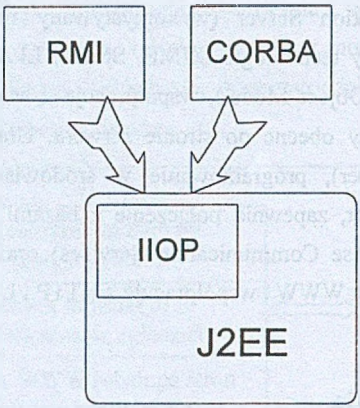
3. Warstwa pośrednia

Podczas tworzenia aplikacji WWW bardzo istotną rolę odgrywa warstwa znajdująca się między warstwą danych, reprezentowaną zwykle przez relacyjną bazę danych, a warstwą prezentacji, do tworzenia której wykorzystuje się przeglądarki WWW, zwaną przez to warstwą pośrednią (*ang. middleware*). Właśnie w tej warstwie implementowanych jest większość algorytmów przetwarzania danych otrzymanych od klienta oraz zgromadzonych w warstwie danych.

Z powodu roli, jaką pełni warstwa pośrednia, bardzo istotną rzeczą jest, aby była ona osadzona w odpowiednim środowisku programistycznym. W klasycznych systemach warstwa ta była realizowana poprzez implementację skryptów CGI, pisanych w dowolnym języku, jak również stron dynamicznych, np. PHP czy ASP + Visual Basic. Jednak powyższe rozwiązania słabo wspierają podejście obiektowe ze względu na rozproszenie obiektów, jak również nie ułatwiają rozłożenia obliczeń na kilka serwerów.

Platforma CORBA (*ang. Common Object Request Broker Architecture*) jak również zaproponowane w języku Java RMI (*ang. Remote Method Invocation*) umożliwiają realizację rozproszenia obiektów na poszczególne serwery. Umożliwia to podejście obiektowe do tworzenia implementacji warstwy pośredniej aplikacji WWW, co w połączeniu z wprowadzeniem przetwarzania wielowątkowego może znacznie ułatwić tworzenie warstwy pośredniej.

Powyższe rozwiązania ułatwiają wprowadzenie skalowalności do warstwy pośredniej, jednak nie przedstawiają żadnego rozwiązania wprost. Projektant sam musi stworzyć odpowiednie mechanizmy w zależności od rozwiązywanego problemu, co powoduje dodatkowy narzut pracy. Rozwiązaniem jest zastosowanie serwera aplikacyjnego, który wprowadza pewne ustalone sposoby zapewnienia skalowalności na więcej niż jeden serwer – komputer.



Rys. 3. Rola RMI i CORBA w standardzie J2EE
Fig. 3. Role of RMI and CORBA in J2EE standard

Każdy z producentów serwerów aplikacji może przedstawić własne rozwiązanie umożliwiające rozłożenie obiektów zdalnych na serwerach warstwy pośredniej, jak również dostarczyć własne sposoby tworzenia takich obiektów – API (*ang. Application Programmer Interface*). Większość proponowanych rozwiązań była zgodna ze standardem CORBA lub RMI, jednak dodatkowe funkcje nie były ustandaryzowane.

Obecnie dominującym standardem tworzenia warstwy pośredniej z użyciem serwera aplikacyjnego jest zaproponowana przez firmę Sun Microsystems, oparta na języku Java, platforma J2EE (*ang. Java 2 Enterprise Edition*). Dzięki zastosowaniu protokołu RMI-IIOP (*ang. Remote Method Invocation – Inter Internet ORB Protocol*) jest on zgodny ze standardami CORBA i RMI, a dodatkowe funkcje są jasno opisane w dostępnej publicznie specyfikacji, dzięki czemu może on być implementowany przez wielu producentów.

4. Serwery J2EE

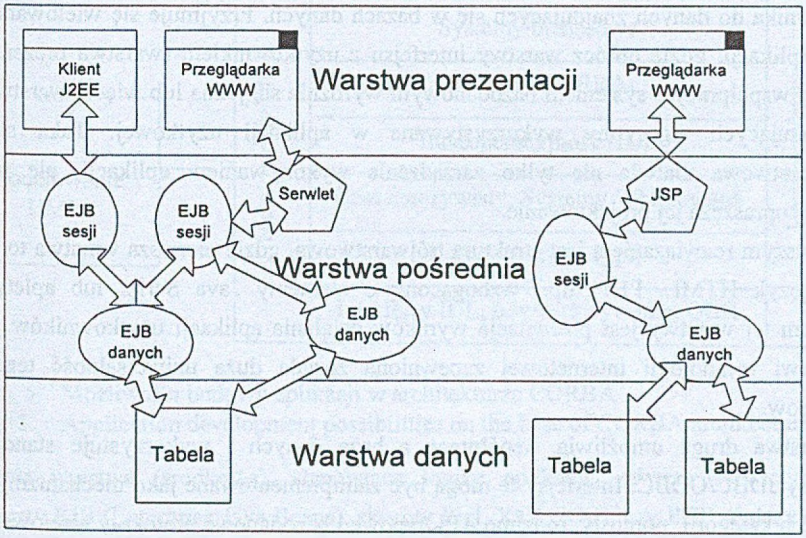
Podstawowym elementem standardu J2EE jest architektura EJB (*ang. Enterprise Java Beans*), której mechanizmy odpowiedzialne są za skalowalność w warstwie pośredniej. Obiekty EJB mogą być zapisywane, odtwarzane i przenoszone pomiędzy komputerami bez dodatkowych zabiegów ze strony programisty przez serwer aplikacji. Dzięki temu uzyskuje się skalowalność oraz przezroczystość ze względu na lokalizację danego obiektu.

Wyróżniamy dwa rodzaje obiektów EJB: danych (*ang. enetity EJB*) i sesji (*ang. session EJB*). Obiekty danych umożliwiają dostęp do źródeł danych, np. relacyjnej bazy, natomiast obiekty sesji EJB umożliwiają śledzenie pojedynczego połączenia – przeprowadzanie sesji z

klientem. Obiekty sesji odbierają żądanie klienta - może one być wykonane bezpośrednio przez obiekt sesji lub może on skorzystać z metod obiektów danych.

Obiekty EJB sesji mogą mieć stan lub być bezstanowe. W pierwszym przypadku dla każdej sesji – każdego klienta musi istnieć osobny obiekt, natomiast w drugim wystarczy jeden, ewentualnie powielony na kilku komputerach. Obiekty EJB danych mogą bezpośrednio odzwierciedlać dane przechowywane w bazie – np. wiersze tabeli. Wtedy ich odczyt i zapis są zarządzane bezpośrednio przez serwer aplikacji lub mogą być odzwierciedleniem bardziej złożonej struktury. Wówczas procedury zachowywania i ładowania muszą być zaimplementowane przez programistę.

Na styku warstwy pośredniej z warstwą prezentacji używane są zwykle mechanizmy umożliwiające dynamiczne tworzenie stron WWW. Zwykle są to serwlety lub strony JSP. Serwlet jest to obiekt zwykle osadzony w serwerze HTTP, każde żądanie wysłane do tego serwera zamieniane jest na wołanie metody tego obiektu. Wynikiem wykonania takiej metody jest dokument wysyłany do przeglądarki WWW. Strony JSP (*ang. Java Server Pages*) również umożliwiają obsługę żądań HTTP z tą różnicą, że wykonywany kod programu jest osadzony w dokumencie HTML lub XML. Ponadto użytkownik może korzystać z oprogramowania umożliwiającego bezpośrednią komunikację z obiektami sesji (patrz rys. 4).



Rys. 4. Warstwy serwera aplikacji J2EE
Fig. 4. Layers of J2EE server application

Oprócz architektury EJB oraz mechanizmów tworzenia dokumentów (serwlety i JSP) ważną rolę odgrywają inne biblioteki zawarte w standardzie J2EE. JMS (*ang. Java*

Messaging Service) umożliwia przesyłanie wiadomości pomiędzy komponentami warstwy pośredniej jak również jej otoczeniem. JTS (*ang. Java Transaction Service*) jest wykorzystywana do przeprowadzania rozproszonych transakcji. JNDI (*ang. Java Naming and Directory Interface*) dostarcza usług katalogowych. Dzięki tej usłudze możliwe jest wyszukiwanie obiektów znając jedynie ich nazwę.

JavaMail umożliwia współpracę z systemem wysyłania poczty elektronicznej. Klasy znajdujące się w tym pakiecie umożliwiają odbieranie, wysyłanie oraz modyfikację poczty elektronicznej. Jest on tak skonstruowany, że umożliwia wykorzystanie różnych rodzajów systemów dostępu do folderów poczty: SMTP, IMAP czy POP3, w zależności od dostarczonego „sterownika".

5. Metodologia projektowania aplikacji

Rozwój serwerów aplikacji znacznie rozmył granicę istniejącą pomiędzy warstwami systemowymi (obejmującymi sieć lokalną, system operacyjny, bazę danych, maszynę wirtualną i oprogramowanie serwera aplikacji) a warstwami użytkowymi reprezentującymi logikę różnego typu aplikacji. Aplikacje WWW w większości przypadków bazują na dostępie użytkownika do danych znajdujących się w bazach danych. Przyjmuje się wielowarstwowy model aplikacji, gdzie oprócz warstwy interfejsu z użytkownikiem (warstwa prezentacji) i warstwy współpracy z systemem bazodanowym wyróżnia się jedną lub więcej warstw logiki reprezentujących algorytmy wykorzystywane w aplikacji użytkowej. Taka struktura wielowarstwowa ułatwia nie tylko zarządzanie wykonywaniem aplikacji, ale również znacznie upraszcza jej projektowanie.

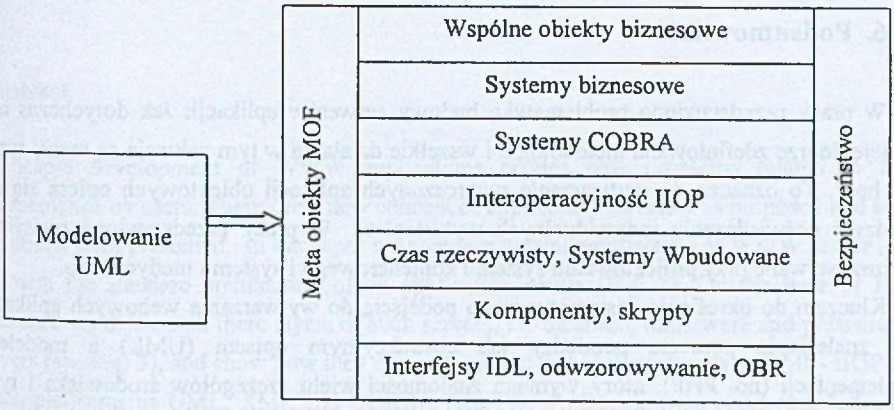
Najprostszym rozwiązaniem jest struktura trójwarstwowa, gdzie pierwsza warstwa to zestaw stron (język HTML, PHP, itp.) wzbogacony o elementy Java Script lub aplety Java. Zadaniem tej warstwy jest prezentacja wyników działania aplikacji użytkowników. Dzięki rozwojowi technologii internetowej zapewniona została duża uniwersalność tego typu interfejsów.

Warstwa druga umożliwia współpracę z bazą danych i wykorzystuje standardowe interfejsy JDBC/ODBC. Interfejsy te mogą być zaimplementowane jako mechanizmy jednej z czterech kategorii: pomosty, rodzime API, typowe i bazodanowe protokoły sieciowe wraz z odpowiednimi sterownikami javowymi.

W przypadku pomostu aplikacja Javy jest uruchamiana po stronie klienta z wykorzystaniem JDBC API. Odwołanie JDBC jest konwertowane na odwołanie ODBC i przekazane do odpowiedniego sterownika bazy danej. Taka konstrukcja jest nieefektywna i

wymaga przeinstalowania kodu na każdym kliencie wykorzystującym taki pomost. W przypadku rodzimego API dostarczana jest biblioteka przekładająca funkcje JDBC na wewnętrzny język zapytań DBMS, różny dla różnych baz danych (np. Dblib dla Sybase, Oclib dla Oracle). Sterowniki te pisane są częściowo w C/C++ i wymagają przeinstalowywania bibliotek dostawcy bazy danych, ale są znacznie szybsze niż bazujące na pomostach. Innym rozwiązaniem jest przekładanie odwołań JDBC na typowy protokół sieciowy niezależny od bazy danych, który następnie jest tłumaczony na wewnętrzne odwołania API do bazy danych. Wówczas sterownik JDBC działa u klienta i odpowiada za przekazywanie przez sieć rozkazów SQL-owych do serwera oraz przyjmowanie danych z serwera. Do serwera bazodanowego oprogramowanie jest podłączone przy użyciu bibliotek klienta lub samego ODBC. Problemem jest dobór odpowiedniej konfiguracji serwera bazodanowego.

Czwarte rozwiązanie korzysta z bazodanowego protokołu sieciowego. Wówczas sterowniki przekładają JDBC bezpośrednio na protokoły rodzime bez użycia API i ODBC. Takie rozwiązanie zapewnia maksymalną wydajność i najczęściej jest stosowane w implementacjach serwerów aplikacji.



Rys. 5. Możliwości budowy aplikacji w architekturze CORBA
Fig. 5. Application development possibilities on the base of CORBA architecture

Trzecia warstwa (środkowa), stanowiąca logikę aplikacji, zawiera serwlety Javy, komponenty EJB (Enterprise Java Beans), skrypty Perl, XML, CGI, czy PHP. Zależy ona od konkretnych zadań użytkownika, które mają być zaimplementowane, lub innych aplikacji, które mają być jednocześnie wykorzystane. Przy opracowaniu konstrukcji, wzorców czy komponentów danej aplikacji wykorzystuje się UML (Unified Modelling Language). Uzupełniając UML [5] wykorzystuje się MOF (Meta-Object Facility), który dostarcza

standardowego katalogu dla metadanych wewnątrz architektury CORBA. Wszystkie aspekty interoperacyjności (GIOP) wraz z protokołami transportowymi (TCP/IP) tworzą standardowy protokół współpracy IIOP (Internet Inter - ORB Protocol). Na rys. 5 przedstawiono podstawowe warstwy definiowane przez OMG. Istotne jest określenie źródła danych, z których korzystać będzie aplikacja. Może to być typowa baza danych, czy rozproszone strony XML. Następnym krokiem jest utworzenie warstw pośredniczących Entity Beans - komponenty związane z danymi, Session Beans - komponenty związane z powodzeniem sesji przez klienta. Tak więc projektowanie aplikacji WWW związane jest z trzema zasadniczymi działaniami:

- opracowanie serwletów i komponentów EJB dla żądanej aplikacji,
- przygotowanie odpowiedniej konfiguracji i środowiska wykonania aplikacji,
- uruchomienie aplikacji w zadanym środowisku.

Dalsze działania wymagają zarówno odpowiedniego rozlokowania aplikacji na węzłach sieci, jak i skonfigurowania systemu, co powoduje konieczność bezpośredniego współdziałania administratora projektanta i użytkownika tej aplikacji.

6. Podsumowanie

W pracy przedstawiono problematykę budowy serwerów aplikacji. Jak dotychczas nie istnieje dobrze zdefiniowana metodologia i wszelkie działania w tym zakresie są raczej typu "ad hoc". To oznacza, że wytwarzanie rozproszonych aplikacji obiektowych opiera się na wiedzy i doświadczeniu indywidualnych wytwórców. W pracy przedstawiono podejście wykorzystywane przy projektowaniu systemu kontenerowego i systemu medycznego.

Kluczem do określenia systematycznego podejścia do wytwarzania webowych aplikacji jest znalezienie pomostu pomiędzy ich abstrakcyjnym opisem (UML) a modelem implementacji (np. PHP), który wymaga znajomości wielu szczegółów środowiska i tym samym odpowiada niskiemu poziomowi programowania.

LITERATURA

1. Arnold K., Gosling J.: The Java Programming Language. Reading, Mars: Addison-Wesley, 1996.
2. Baker S.: CORBA Distributed Objects Using Orbix. Essex UK: Addison - Wesley Longman, 1997.

3. Coulouris G., Dallimore J., Kindberg T.: Distributed Systems: Concept and Design. Harlow UK: Addison - Wesley, 1994.
4. Gellersen H.W., Gaedke M.: Object - Oriented Web Application Development. IEEE Internet Computing, Jan/Feb., 1999.
5. Harman P., Watson M.: Understanding UML: The Developers Guide. San Francisco: Morgan Kaufman, 1998.
6. Krawczyk H., Dziubich T., Mazurkiewicz A., Szpryngier P., Ślesicki M.: Problemy projektowania i implementacji systemu Nordcon. V Konferencja "Komputerowe Systemy Wielodostępne", 1999.
7. Krawczyk H., Knopa R., Lipczyńska K. Lipczyński M.: web-based Endoscopy Recommender System - ERS. Int. Conference on Parallel Computing in Electrical Engineering. Quebec, Canada, 2000.

Recenzent: Prof. dr hab. inż. Andrzej Grzywak

Wpłynęło do Redakcji 28 marca 2001 r.

Abstract

Rapid development of WWW applications creates new problems related to their maintenance by users. Therefore a new concept of application servers was proposed and some solutions were presented. In the paper we compare typical architecture of WWW server (Fig. 1) with the standard architecture of on application server (Fig. 2). On the base of J2EE standard we discuss all there layers of such servers, i.e. database, midlewere and presentation layers (see Fig. 3), and show how they can interact and use such protocols as RMI - IIOP and such platforms as UML, XML and CORBA (see Fig. 4). Besides, the solid (but "ad hoc") design methodology is proposed. This methodology allows to specify metaobjects of user specifications in UML and then use the available, web-based, mechanisms to create the suitable layers of the user application (see Fig. 5). Several web-based software development phases: such as implementation, maintenance and evaluation were distinguished and their difference from conventional application development ones, due to them being oriented on existing component integration and available system layers configuration was stressed.