

Technologie tworzenia internetowych aplikacji działających po stronie serwera

Piotr Kołaczkowski

e-mail: P.Kolaczkowski@elka.pw.edu.pl

Wydział Elektroniki i Technik Informatycznych Politechniki Warszawskiej,
Instytut Informatyki
ul. Nowowiejska 15/17, 00-665 Warszawa

6 stycznia, 2004

Streszczenie: Od 1993 roku obserwuje się bardzo szybki wzrost liczby systemów opartych o usługę WWW. W pracy przedstawiono wybrane cechy współczesnych technologii tworzenia aplikacji internetowych.

Słowa kluczowe: aplikacje internetowe, serwer WWW, języki programowania, jakość oprogramowania

1. Wprowadzenie

Celem niniejszej pracy jest możliwie obiektywne przedstawienie współczesnych technologii tworzenia serwerowych aplikacji komunikujących się z użytkownikiem za pośrednictwem przeglądarki internetowej w oparciu o protokół HTTP.

W związku z dynamicznym rozwojem sieci Internet i jej usług, w ciągu ostatnich kilkunastu lat obserwuje się bardzo szybki wzrost zapotrzebowania na oprogramowanie tego typu. Obecnie można je spotkać zarówno w zastosowaniach tak błahych jak np. liczniki odwiedzin na stronach, jak i w bardzo zaawansowanych np. interfejsy systemów zarządzania przedsiębiorstwami lub instytucjami.

Praca ta nie jest podręcznikiem, jak tworzyć takie aplikacje, ani jakich używać do tego celu narzędzi. Przedstawia jedynie z jakimi problemami można się zetknąć w tej dziedzinie, jak twórcy oprogramowania starają się je rozwiązać i jak zostały rozwiązane w konkretnych produktach.

W chwili obecnej na rynku jest obecnych wiele różnych technologii tworzenia aplikacji internetowych. Stworzenie kompletnej i aktualnej listy jest trudne (a nie wiadomo, czy w ogóle możliwe), ponieważ dziedzina ta stale się rozwija, a liczba pakietów oprogramowania szybko rośnie. Poniższa lista zawiera najpopularniejsze i najczęściej spotykane systemy tworzenia aplikacji internetowych kolejno omawiane w dalszej części pracy:

- CGI – Common Gateway Interface (NCSA)
- FastCGI – Fast Common Gateway Interface (Open Market, Inc.)
- PHP – Hypertext Preprocessor
- ASP – Active Server Pages (Microsoft, Inc.)
- iHTML (Inline Internet Systems, Inc.)
- ColdFusion (Macromedia)
- J2EE - Java Servlets oraz Java Server Pages (Sun, Inc.)
- ASP.NET - Active Server Pages .NET (Microsoft, Inc.)

Aby zwiększyć przejrzystość opracowania, opis każdej technologii został podzielony na następujące kategorie:

1. Metodologia(e) projektowania aplikacji.
2. Mechanizmy wspomagające dialog pomiędzy użytkownikiem a aplikacją.
3. Mechanizmy dostępu do danych.
4. Mechanizmy przetwarzania danych.
5. Mechanizmy i narzędzia usprawniające wykrywanie i usuwanie błędów.
6. Szybkość działania i zużycie zasobów.
7. Przenośność, uniwersalność i dostępność.
8. Bezpieczeństwo.
9. Inne cechy szczególne.

Ułatwia to porównywanie technologii między sobą oraz wybór najlepszej do rozwiązania konkretnego problemu.

2. CGI – Common Gateway Interface

CGI zostało opracowane w 1993 roku przez NCSA jako technologia pozwalająca przeglądarce internetowej odbierać dane wygenerowane przez program uruchomiony przez serwer WWW. Pierwsza implementacja CGI powstała dla serwera WWW NCSA działającego w systemie operacyjnym UNIX, jednak

bardzo szybko pojawiły się implementacje dla innych serwerów i innych systemów operacyjnych. Obecnie prawie wszystkie serwery WWW potrafią uruchamiać aplikacje CGI.

Zasada działania aplikacji CGI jest bardzo prosta. Kiedy serwer WWW otrzymuje od klienta żądanie pobrania dokumentu, który jest odpowiednio oznaczonym plikiem wykonywalnym (np. poprzez rozszerzenie .cgi), to zamiast przesłać ten plik do klienta, wykonuje go w odpowiednio przygotowanym środowisku. Na środowisko to składają się odpowiednie zmienne systemowe zawierające m.in. dane wysłane przez klienta oraz standardowe strumienie: wejścia, wyjścia i błędów. Plik wykonywalny (zwany też czasem skrypsem CGI) przesyła odpowiedź przeglądarce wysyłając wszystkie dane na standardowe wyjście i kończy swoje działanie. Skrypty CGI można pisać w dowolnym języku programowania. Najczęściej wykorzystywane są Perl, Python i C/C++. Więcej szczegółów o projektowaniu i implementacji aplikacji CGI znajduje się w 10.

2.1. Metodologia projektowania aplikacji

Mechanizm działania CGI ma znaczący wpływ na sposób projektowania złożonych aplikacji. Zwykle aplikacja składa się z wielu samodzielnych, małych plików wykonywalnych, z których każdy realizuje jeden (czasem więcej) proces przetwarzający dane np. formularz wypełniony przez użytkownika albo zapytanie do relacyjnej bazy danych. Naturalne więc wydaje się podejście strukturalne.

CGI nie daje natomiast żadnego wsparcia dla obiektowego projektowania aplikacji. Można stworzyć w pełni obiektowy model całej aplikacji (i zobrazować go np. w języku UML), ale składniki tego modelu (obiekty, klasy) nie będą się w prosty i bezpośredni sposób przekładały na składniki działającej aplikacji.

Dlatego czasem stosuje się podejście mieszane polegające na tym, że poszczególne skrypty projektuje i implementuje się obiektowo (np. w C++), ale całość opisana jest modelem strukturalnym (np. ERD + DFD).

2.2. Dialog między aplikacją a użytkownikiem

Realizacja dialogu między aplikacją CGI a użytkownikiem jest dużo trudniejszym zadaniem niż w przypadku np. zwykłej aplikacji konsolowej. CGI dobrze nadaje się jedynie do obsługi konwersacji bezkontekstowych (odpowiedź jest funkcją bieżącego zapytania – np. tak działają statyczne serwisy WWW). Natomiast, jeśli odpowiedź jest zależna również od poprzednich zapytań wysłanych przez tego samego użytkownika, to powstaje szereg problemów.

Pierwszy problem polega na identyfikacji klienta. Dany skrypt CGI może być “odpytywany” przez wielu klientów na raz – jak rozróżnić poszczególne konwersacje (sesje)? Niestety projektant sam musi poradzić sobie z tym problemem. Nadmienię, że jednym z częstych błędów jest identyfikowanie użytkownika jedynie po jego numerze IP. Niestety może się zdarzyć, że dwóch klientów z tej samej podsieci będzie równocześnie korzystał z naszej aplikacji i będą mieli taki sam nr IP. Metody radzenia sobie z tym problemem są omówione szczegółowo dalej.

Drugi problem polega na zapamiętywaniu stanu aplikacji pomiędzy zapytaniami. Niestety, po wysłaniu odpowiedzi skrypt CGI kończy swoje działanie. Gdy ten sam klient wyśle do tego skryptu nowe zapytanie, to skrypt będzie w tym samym stanie, co przy poprzednim pytaniu. Problem ten projektanci rozwiązują na różne sposoby:

- Zapis stanu sesji na dysku. Jest to sposób dość trudny w implementacji – trzeba pamiętać o zabezpieczeniu danych przed niepowołanym dostępem oraz ich usuwaniu po jakimś czasie. Należy też brać pod uwagę możliwość ataku DOS (denial of service) poprzez całkowite zapełnienie dostępnej przestrzeni na dysku.

- Zapis stanu sesji w bazie danych. Odnoszą się do tego te same uwagi co wyżej. Jeśli aplikacja korzysta z zewnętrznego systemu baz danych, to na ogół ta opcja jest stosowana.
- Zapis danych po stronie klienta. Dane można zapisać jako tzw. „ciasteczko” (cookie). Mechanizm „ciasteczek” polega na przechowywaniu niewielkich porcji danych przez przeglądarkę i odsyłaniu ich z każdym zapytaniem do serwera. Możliwe jest też zapisanie danych w treści wysłanej odpowiedzi. Ponieważ odpowiedź serwera ma zwykle postać dokumentu HTML, to można ten dokument tak przygotować, żeby kolejne zapytanie zawierało w sobie stan sesji. Realizuje się to albo poprzez wykorzystanie parametrów przekazywanych po znaku „?” w odwołaniu do skryptu (dotyczy zapytań typu GET) albo poprzez zastosowanie ukrytych pól formularza (dotyczy zapytań typu POST). Sposoby te są dobre, jeśli zapis stanu sesji jest dość krótki i nie wydłuży znacząco czasu przekazywania danych między aplikacją a przeglądarką.
- Inne, zależne od inwencji twórców oprogramowania.

2.3. Dostęp do danych

CGI nie oferuje żadnych dodatkowych mechanizmów dostępu do danych, prócz oferowanych bezpośrednio przez system operacyjny i dostępnych w danym języku programowania. Zwykle dane można przechowywać w plikach albo w bazie danych (na ogół relacyjnej).

Przechowywanie danych w plikach

Specyfika CGI powoduje, że przechowywanie danych w plikach nastęrcza pewne trudności. W typowych konfiguracjach serwera WWW, wszystkie skrypty CGI są wykonywane z uprawnieniami tego samego użytkownika (np. w systemach Unix może być to “nobody” albo “www”) niezależnie od położenia skryptu. Dlatego wszelkie pliki zapisywane i czytane przez skrypty, muszą mieć ustawione takie uprawnienia, aby ten użytkownik miał do nich dostęp. Często jednak bywa tak, że istnieje więcej użytkowników niż jeden w danym systemie. Wtedy ich aplikacje CGI mogą uzyskać wzajemny dostęp do swoich danych.

Żeby rozwiązać ten problem, dla serwerów WWW opracowano szereg ulepszeń, które pozwalają wykonywać skrypty CGI z uprawnieniami ich właściciela, a nie wspólnego użytkownika. Jednym z takich ulepszeń jest np. suexec dla serwerów Apache. Wtedy każdy skrypt ma dostęp jedynie do swoich danych i danych swojego właściciela.

Z drugiej strony mechanizm ten wprowadza ryzyko zdalnego uzyskania przez niepowołane osoby uprawnień właściciela aplikacji. Uprawnienia te są zwykle dużo szersze niż uprawnienia wspólnego użytkownika, kiedy ten mechanizm nie jest używany. Dlatego bardzo rzadko mechanizm ten jest stosowany na serwerach ogólnodostępnych, gdzie każdy może umieszczać swoje aplikacje CGI.

Przechowywanie danych w zewnętrznej bazie danych

W przypadku przechowywania danych w bazie danych nie zachodzi problem ochrony dostępu pojawiający się przy użyciu plików. Każda aplikacja może posiadać swoją własną, indywidualnie chronioną bazę danych. Zwykle systemy baz danych używają uwierzytelniania dostępu za pomocą haseł. Hasło może być albo podawane za każdym razem przez użytkownika aplikacji (klienta), albo zaszyte na stałe w jej kodzie. Takie przechowywanie hasła jest dosyć bezpieczne, bo zwykle można tak ustawić atrybuty plików wykonywalnych zawierających hasło, żeby nie można ich było czytać, a jedynie można było wykonywać.

Niestety z drugiej strony wydajność takiego rozwiązania może być mniejsza, a poza tym nie zawsze dysponujemy systemem baz danych.

2.4. Mechanizmy wspomagające przetwarzanie danych

Samo CGI nie dostarcza żadnych udogodnień w zakresie przetwarzania danych. Twórca aplikacji jest zdany na udogodnienia oferowane przez język programowania i ewentualnie dodatkowe biblioteki. Z tego powodu być może jednym z popularniejszych języków pisania aplikacji CGI jest Perl, który jest specjalnie przystosowany do przetwarzania zawiłych łańcuchów znaków w różnych formatach za pomocą wyrażeń regularnych. Ponadto dla wielu języków (głównie C, Perl, Python) powstały pakiety ułatwiające implementację typowych zadań takich jak np. odbieranie danych przekazanych w formularzu zapisanym w HTML, czy odbieranie ciasteczka.

2.5. Wykrywanie błędów

Wykrywanie błędów w skryptach CGI jest trudne. Nie istnieją narzędzia automatyzujące ten proces. Standardowe narzędzia jak np. debuggery nie są zbyt przydatne, gdyż nie zostały przystosowane do pracy z aplikacjami pracującymi jako skrypty CGI. Zwykle wymagają, żeby aplikację uruchamiać za ich pośrednictwem, a skrypty CGI uruchamiane są za pośrednictwem serwera WWW. Nawet te debuggery, które potrafią dołączać się do wcześniej uruchomionego procesu, na niewiele się zdadzą, bo proces aplikacji CGI uruchamiany jest w nieprzewidywalnym momencie i istnieje jedynie przez chwilę.

Testerzy radzą sobie w inny sposób – metoda zwana żargonowo “printf debugging” jest łatwa do zastosowania i nie wymaga żadnych dodatkowych narzędzi. Polega ona na umieszczeniu w kodzie programu dodatkowych instrukcji informujących testera o przepływie sterowania i wartościach zmiennych. Informacje te można wysłać do przeglądarki (strumień wyjściowy), do logów serwera WWW (strumień błędów) lub np. do pliku. Po wykryciu i poprawieniu błędów instrukcje te są dezaktywowane.

2.6. Szybkość działania i zużycie zasobów

Szybkość działania aplikacji CGI ustępuje zwykle aplikacjom tworzonym w innych systemach. Obsługa każdego zapytania wiąże się z uruchomieniem i zakończeniem nowego procesu, a to może stanowić spory narzut czasowy. W poniższej tabeli przedstawione są wyniki pomiaru czasu wykonania bardzo prostego programu w różnych systemach operacyjnych na tym samym komputerze (Pentium II XEON 450 MHz, 128 MB RAM):

Linux – Slackware (kernel 2.4.19)	(7,2 ± 0,2) ms
Unix – FreeBSD 4.6.2	(2,3 ± 0,2) ms
Windows 2000 Professional SP 3	(34 ± 3) ms

Żeby zlikwidować te narzuty, niektóre serwery WWW zostały wyposażone we wbudowane interpretery niektórych popularnych języków, w których pisane są skrypty CGI (np. mod_perl dla serwera Apache). Wtedy obsługa zapytania nie powoduje uruchomienia nowego procesu, a jest wykonywana w kontekście działającego wcześniej procesu serwera WWW. Taki sposób uruchamiania programu narzuca wymaganie, aby plik wykonywalny był dodatkowo “do odczytu” dla serwera WWW, co z kolei może wiązać się z obniżeniem poziomu bezpieczeństwa skryptu.

Niestety to rozwiązanie można stosować jedynie do języków interpretowanych lub “kompilowanych w locie”¹. Zwykle programy pisane w tych językach ustępują efektywności programom normalnie kompilowanym (np. Pascal, C/C++), choć na ogół nie stanowi to poważnej przeszkody, bo skrypty CGI na ogół nie wykonują bardzo złożonego i czasochłonnego przetwarzania danych.

Zużycie pamięci jest natomiast ściśle zależne od tego, jak aplikacja została napisana i do czego służy. Krótkotrwałość istnienia skryptu powoduje, że stworzenie aplikacji, która bezpowrotnie gubi pamięć (tzw. przecieki pamięci), jest trudne.

2.7. Przenośność, uniwersalność i dostępność

Technologia CGI była projektowana z myślą o wykorzystaniu jej w wielu serwerach WWW i przez wielu różnych programistów znających różne języki programowania. W tej chwili ciężko jest znaleźć serwer, który nie posiadałby możliwości wykonywania skryptów CGI, tak samo jak ciężko jest znaleźć język programowania, który nie dawałby możliwości korzystania ze zmiennych systemowych i standardowych strumieni we/wy. Z tego wynika również olbrzymia uniwersalność tej technologii. Dodatkowo możliwość użycia różnych bibliotek specyficznych dla użytego języka programowania powoduje, że technologia ta znalazła bardzo wiele zastosowań (mimo niskiej wydajności).

Standard CGI nie jest zależny również od systemu operacyjnego, dlatego na ogół przeniesienie aplikacji z jednego systemu na drugi zależy głównie od tego, czy aplikacja nie korzysta poza CGI z jakichś niestandardowych bibliotek, rozszerzeń, poleceń systemowych itd. Prawdopodobnie aplikację napisaną w języku Perl i korzystającą jedynie z poleceń tego języka będzie łatwo przenieść na inny system, gdzie jest dostępny interpreter języka Perl. Natomiast aplikację napisaną w języku C i korzystającą z niskopoziomowego API systemu Unix bardzo ciężko będzie dostosować do pracy w systemie MacOS, mimo dostępności kompilatora języka C dla tego systemu. CGI określa jedynie sposób komunikacji pomiędzy aplikacją a klientem. Wszelkie pozostałe aspekty pozostawia projektantom aplikacji i to od nich zależy, czy stworzą przenośne aplikacje, czy nie.

2.8. Bezpieczeństwo

Bezpieczeństwo aplikacji CGI jest zależne od wielu czynników takich jak np. użyty język programowania lub środowisko uruchamiania aplikacji.

Wiadomo, że niektóre języki programowania wysokiego poziomu szczególnie ułatwiają popełnianie błędów krytycznych dla bezpieczeństwa aplikacji (a nawet całego systemu), a inne jedynie w mniejszym stopniu. Przykładowo pisząc skrypt CGI w języku C i używając niskopoziomowych funkcji operujących na łańcuchach znaków takich jak `scanf`, `strcpy`, `strcat`, `gets`² itp. bardzo łatwo można popełnić błąd typu “buffer overflow” lub “off by one”. Błędy tego typu co jakiś czas znajduwane są nie tylko w skryptach CGI pisanych w C, ale nawet w funkcjach jąder popularnych systemów operacyjnych, gdzie w końcu bezpieczeństwo i ochrona danych są dosyć ważne. Konsekwencje ich przeoczenia mogą być bardzo poważne i na ogół skutkują tym, że niepowołana osoba z zewnątrz może przejąć kontrolę nad aplikacją. Różne rodzaje błędów w skryptach CGI oraz metody ich unikania omówiono w 10.

¹ Kompilacja w locie polega na tym, że skrypt jest kompilowany do pamięci przy pierwszym uruchomieniu, a przy każdym następnym jest wykonywany zapamiętany wcześniej kod. Ważna jest przy tym szybkość kompilacji, dlatego jej jakość jest gorsza (optymalizacja!), a i gramatyka języka nie może być zanedbana.

² ciekawostka: `gets()` został już dosyć dawno “zabroniony”. Zalecane jest używanie `fgets()`, ale jego wykorzystanie nie eliminuje całkowicie ryzyka popełnienia błędu.

Podobnie środowisko, w którym aplikacja jest uruchamiana ma kluczowe znaczenie. Wykorzystanie mechanizmów zmieniających poziom uprzywilejowania skryptu (np. suexec) może podnieść poziom bezpieczeństwa (np. poprzez lepszą ochronę plików z danymi), ale z drugiej strony ich nieumiejętne stosowanie wraz z błędami wewnątrz aplikacji może przynieść katastrofalne skutki.

3. FastCGI

FastCGI jest technologią wprowadzoną przez firmę OpenMarket Inc. w 1996 roku, aby umożliwić tworzenie bardzo szybkich i wydajnych aplikacji internetowych. FastCGI zastępuje standardowy interfejs CGI i od strony użytkownika jest do CGI bardzo zbliżone, dzięki czemu konwersja aplikacji CGI do FastCGI jest dosyć prosta. Podobnie jak w klasycznej technologii CGI, zapytania są przekazywane do aplikacji za pośrednictwem strumienia standardowego wejścia i zmiennych systemowych, a wyniki zwracane za pośrednictwem strumienia standardowego wyjścia.

W rzeczywistości dane pomiędzy serwerem WWW, a aplikacją są przekazywane za pośrednictwem specjalnego protokołu, ale mechanizm ten jest dla programisty ukryty. Poza tym FastCGI w nieco inny sposób zarządza uruchamianiem procesów. Zamiast dla każdego zapytania uruchamiać nowy proces systemowy, serwer WWW uruchamia go raz, na samym początku. Proces przetwarza zapytania w pętli i po każdorazowym wysłaniu odpowiedzi do przeglądarki, pozostaje w pamięci. Dla zwiększenia wydajności możliwe jest uruchomienie na początku od razu całej puli procesów, dzięki czemu uzyskuje się lepszą współbieżność – jeśli jedno zapytanie będzie wymagało więcej czasu procesora, to nie wstrzyma wykonywania pozostałych. Szczegółowe informacje o możliwościach konfiguracji FastCGI oraz zasadach tworzenia aplikacji znajdują się na stronie WWW producenta 10.

3.1. Metodologia projektowania aplikacji

Ponieważ narzut na uruchamianie procesu jest pomijalny, zwykle dobrze jest zaprojektować całą aplikację jako pojedynczy skrypt, który przetwarza wszystkie rodzaje zapytań. Skrypt musi zawierać specjalną pętlę obsługi zapytań, której postać jest określona w dokumentacji. Czynności inicjalizacyjne, takie jak np. otwieranie połączeń do systemu baz danych, dobrze jest umieścić poza tą pętlą (dla zwiększenia wydajności).

Dzięki temu, że skrypt „żyje” dłużej niż czas jednego zapytania, możliwe jest zastosowanie obiektowej metodologii projektowania aplikacji. Skrypt nie musi być tylko izolowanym procesem, który przetwarza dane wprowadzane na jego wejście, ale może zachowywać się jak obiekt lub zestaw obiektów (w sensie projektowania obiektowego). Pozwala to zaprojektować całą aplikację przy użyciu metod obiektowych.

3.2. Dialog między aplikacją a użytkownikiem

FastCGI oferuje dodatkowe wsparcie dla realizacji dialogu między aplikacją a użytkownikiem. Stosowany może być bardzo prosty mechanizm sesji, w którym użytkownicy identyfikowani są za pomocą wybranych zmiennych ustawianych przez serwer WWW. Może być to np. numer IP klienta lub nazwa użytkownika. Jeśli aplikacja działa jako pula procesów, FastCGI dba o to, aby zapytania od jednego klienta trafiały do dokładnie jednego, zawsze tego samego procesu.

Należy zwrócić uwagę na fakt, że ten mechanizm, choć bardzo wygodny, nie jest wystarczający, aby zapewnić bezpieczeństwo sesji. Problemem jest tu identyfikacja użytkowników. W przypadku, gdy aplikacja nie wykorzystuje mechanizmów uwierzytelniania takich jak HTACCESS, rozróżnianie użytkowników za pomocą

numeru IP może prowadzić do niebezpiecznych anomalii. Przy powszechnym użyciu NAT³, korzystanie z jednego numeru IP przez wielu użytkowników na raz, nie jest rzadkością. Dlatego projektant aplikacji musi wprowadzić dodatkowe mechanizmy wsparcia dla sesji (oparte zwykle o dłuższe lub krótsze numery identyfikacyjne).

3.3. Mechanizmy dostępu do danych

FastCGI nie oferuje żadnego specjalnego wsparcia dla mechanizmów dostępu do danych. Pomaga jednak nieco ten dostęp zoptymalizować. FastCGI eliminuje bowiem konieczność przeprowadzania czasochłonnych czynności inicjalizacyjnych takich jak np. otwarcie pliku lub nawiązanie połączenia z systemem baz danych przy obsłudze każdego zapytania. Zwykle takie czynności wykonuje się jedynie przy uruchamianiu aplikacji, lub ewentualnie przy rozpoczynaniu nowej sesji. Poza tą drobną różnicą, sposoby realizacji dostępu do danych w skryptach FastCGI są takie same jak w CGI.

3.4. Mechanizmy przetwarzania danych

Mechanizmy przetwarzania danych dostępne w FastCGI są takie same jak w CGI, tj. są takie, jakie oferuje język programowania i dostępne biblioteki. Jediną przeszkodę może stanowić fakt, że korzystając z FastCGI nie opłaca się wywoływać zewnętrznych programów do przetwarzania danych (np. w roli filtrów), bo traci się w ten sposób wydajność.

3.5. Mechanizmy i narzędzia usprawniające wykrywanie i usuwanie błędów

Do śledzenia aplikacji FastCGI można użyć narzędzi do śledzenia zwykłych programów. Ponieważ proces aplikacji jest obecny w pamięci przez cały czas aktywności aplikacji, nic nie stoi na przeszkodzie, żeby podłączyć do niego debugger. Ustawiając punkt przerwania na początek ciała pętli obsługi zapytań, można prześledzić od początku do końca, w jaki sposób zachowuje się program przy realizacji zapytania. Oczywiście plik wykonywalny musi zawierać odpowiednie informacje dla debuggera, co uzyskuje się poprzez użycie odpowiedniego kompilatora z odpowiednim zestawem opcji.

Możliwości wykorzystania debuggera są uzależnione od języka programowania, w którym została napisana aplikacja oraz możliwości systemu operacyjnego.

3.6. Szybkość działania i zużycie zasobów

FastCGI jest technologią, która powinna być stosowana tam, gdzie bardzo ważna jest wydajność pracy aplikacji. Wprowadzane narzuty czasowe są tak niewielkie, że czas realizacji zapytania przez aplikację WWW może być zbliżony do czasu realizacji zapytania przez serwer obsługujący statyczne strony WWW.

FastCGI nie zapewnia ochrony przed „przeciekami pamięci”, ale można się przed nimi zabezpieczyć poprzez automatyczne ponowne uruchamianie procesu co pewien czas (lub co pewną liczbę zapytań).

Ciekawą właściwością technologii FastCGI jest jej wsparcie dla aplikacji rozproszonych. Pliki wykonywalne nie muszą znajdować się na tej samej maszynie, co serwer WWW. Serwer WWW może komunikować się z aplikacjami na innych maszynach za pomocą protokołu TCP/IP. Dzięki temu całkowite obciążenie aplikacji można rozdzielić pomiędzy wiele maszyn zmniejszając tym samym ich jednostkowe obciążenie. Sam serwer WWW, choć obsługuje wszystkie zapytania, nie wykonuje żadnego czasochłonnego

³NAT – Network Address Translation – technika umożliwiająca wielu użytkownikom w jednej podsieci dzielić jeden publiczny nr IP dla komunikacji z siecią zewnętrzną.

przetwarzania. Często tego typu zrównoleglenie wykonuje się poprzez postawienie oddzielnych serwerów i dzieleniu ruchu za pomocą odpowiednio skonfigurowanego serwera DNS. Jednak taka metoda wymaga posiadania większej liczby publicznych numerów IP, a zrównoleglenie przy pomocy FastCGI – tylko jednego.

3.7. Przenośność, uniwersalność i dostępność

FastCGI jest dostępne dla wielu różnych serwerów WWW, zarówno komercyjnych jak i darmowych (m.in. Apache, aXesW3, Microsoft IIS, SunOne, WebStar). Pozwala tworzyć aplikacje w wielu różnych językach (m.in. C, C++, Java, Perl, Python, Lisp, Smalltalk), choć liczba wspieranych języków jest ograniczona przez dostępność zestawu odpowiednich bibliotek. Biblioteki te nie są na razie dostępne dla języków Pascal i Basic. Dlatego pod tym względem FastCGI wypada nieco gorzej. Podobnie jak w CGI, przenośność jest silnie zależna od języka programowania oraz sposobu programowania. W ogólnym ujęciu FastCGI nie jest ograniczone przez jakiś konkretny rodzaj architektury ani system operacyjny.

3.8. Bezpieczeństwo

Pod tym względem ta technologia jest bardzo zbliżona do CGI. Ponieważ jednak w FastCGI jeden proces może obsługiwać więcej niż jedną sesję, twórca aplikacji może łatwiej popełnić błąd pozwalający klientowi dostać się do danych nie należących do jego sesji. System operacyjny nie jest w stanie temu zapobiec. W przypadku klasycznej aplikacji CGI, każde wywołanie jest obsługiwane w kontekście odrębnego procesu, więc łatwiej kontrolować dostęp do zasobów związanych z konkretną sesją – ochrona danych w pamięci jest realizowana przez system operacyjny.

Podobnie jak dla CGI, dla FastCGI istnieją mechanizmy pozwalające kontrolować uprawnienia aplikacji. Jednym z nich jest FastCGI suexec.

4. PHP – Hypertext Preprocessor

PHP jest językiem skryptowym ogólnego zastosowania dostosowanym szczególnie do potrzeb aplikacji internetowych. Interpreter PHP jest dostępny w sieci bezpłatnie, a cały projekt jest rozwijany na zasadach Open Source. W sieci Internet można znaleźć bardzo dużo artykułów i poradników na temat tworzenia aplikacji w PHP. Na stronie internetowej 10 są dostępne: interpreter PHP, pełna dokumentacja techniczna oraz bardzo dużo specjalistycznych artykułów dotyczących programowania w PHP. Jest to prawdopodobnie obecnie najpopularniejsza technologia wykorzystywana głównie w prostych zastosowaniach takich jak strony domowe, niewielkie portale i sklepy internetowe.

Zasada działania PHP jest odmienna od CGI. Instrukcje PHP mogą być zagnieżdżane w kodzie HTML. Kiedy z sieci nadejdzie żądanie pobrania odpowiednio oznaczonego dokumentu (zwykle jest to plik z rozszerzeniem .php), serwer WWW przepuszcza dokument przez interpreter PHP działający jako filtr. Interpreter pozostawia kod HTML prawie bez zmian, natomiast wstawki kodu PHP umieszczone pomiędzy odpowiednimi znacznikami są interpretowane i wykonywane. PHP potrafi podstawiać w kodzie HTML wartości zmiennych, które są oznaczane znakiem „\$”. Kod HTML, w którym takie zmienne nie występują, nie jest modyfikowany. Dlatego interpreter PHP może być wykorzystany do udostępniania zarówno aplikacji PHP jak i statycznych stron WWW. Interpreter PHP może być albo niezależnym programem, albo może być wbudowany w serwer WWW.

4.1. Metodologia projektowania aplikacji

PHP jest językiem, który dobrze sprawdza się w realizacji prostych aplikacji zaprojektowanych strukturalnie. Brak wyraźnego podziału między interfejsem użytkownika (najwyższa warstwa – zwykle statyczny kod HTML) a logiką aplikacji (wstawki kodu) powoduje, że PHP niezbyt dobrze nadaje się do tworzenia rozbudowanych aplikacji. Tę właściwość ma dużo języków skryptowych. Przy większej złożoności zachodzi potrzeba stosowania dodatkowych bibliotek służących do uporządkowania projektu np. poprzez wprowadzenie szablonów. Przy takim podejściu dane (np. pliki HTML) są oddzielone od kodu umieszczonego w osobnych plikach. Ułatwia to znacznie późniejszą pielęgnację aplikacji oraz modyfikowanie jej interfejsu użytkownika.

PHP posiada wsparcie dla programowania obiektowego. Wykorzystanie obiektowości w PHP wiąże się jednak obecnie z dużymi narzutami czasowymi, co powoduje, że zysk z zastępowania kodu strukturalnego kodem obiektywnym jest raczej niewielki. W PHP nie jest możliwe tworzenie obiektów o długim czasie życia (przechowywanych w obiektowej bazie danych, pliku, „ciasteczku” itd...). Nie można przez to stworzyć interfejsu użytkownika sterowanego zdarzeniami. Ponadto standardowa biblioteka PHP dostarcza jedynie zestaw pojedynczych funkcji, nie zawiera natomiast definicji ani jednej klasy.

4.2. Dialog między aplikacją a użytkownikiem

PHP posiada wbudowany mechanizm sesji realizowany za pośrednictwem „ciasteczek” oraz przepisywania adresów wywołań. Żeby skorzystać z tego mechanizmu, należy w skrypcie wywołać odpowiednią funkcję, która przydzieli identyfikator sesji oraz umożliwi przechowywanie danych sesji w zewnętrznej pamięci (zwykle na dysku). Identyfikator sesji jest odsyłany do przeglądarki w postaci „ciasteczka” lub, jeśli przeglądarka nie obsługuje „ciasteczek”, zostaje dopisany jako parametr do wszystkich hiperłączy względnych odwołujących się do skryptów PHP. W ten sposób kolejne zapytanie HTTP niesie w sobie informację o sesji.

Niestety, choć mechanizm ten jest bardzo prosty i wygodny w użyciu, w swojej domyślnej postaci poważnie obniża bezpieczeństwo aplikacji. Dzieje się tak z powodu braku ochrony identyfikatorów sesji przed niepowołanym dostępem. Mianowicie PHP przechowuje te numery w systemowym katalogu tymczasowym (na wielu systemach oznaczanym jako *tmp* albo *temp*). Katalog ten jest zwykle dostępny dla wszystkich lokalnych użytkowników systemu, w tym również dla demona(ów) serwera WWW. Rozwiązaniem problemu jest zwykle albo użycie relacyjnej bazy danych jako miejsca składowania danych sesji, albo skorzystanie z gotowej, lepszej biblioteki, albo samodzielne napisanie obsługi sesji. Niestety tego typu pomyłka projektowa sugeruje, że twórcy PHP niezbyt poważnie traktują zagadnienia bezpieczeństwa systemów komputerowych.

4.3. Mechanizmy dostępu do danych

PHP posiada dużo udogodnień służących do obsługi relacyjnych baz danych. Liczba obsługiwanych systemów baz danych jest bardzo duża i stale rośnie. Dla niekomercyjnych systemów baz danych MySQL i PostgreSQL PHP posiada wbudowane wsparcie i nie zachodzi konieczność instalowania osobnych modułów.

Niestety wielość niekompatybilnych ze sobą interfejsów bazodanowych przyczynia się do tego, że nie zawsze jest możliwa szybka migracja z jednego systemu baz danych do innego. PHP obsługuje również uniwersalny interfejs ODBC, jednak jest on rzadko wybierany przez programistów, ponieważ nie może zapewnić takiej funkcjonalności jak interfejsy specjalizowane. Dużą zaletą PHP jest możliwość tworzenia

połączeń stałych z systemem baz danych. Oznacza to, że przy każdym uruchamianiu skryptu nie zachodzi konieczność ponownego, czasochłonnego nawiązywania połączenia.

Warto wspomnieć, że w PHP bardzo prosto realizuje się operacje pobierania danych z innych źródeł niż bazy danych, np. z sieci, za pośrednictwem protokołów FTP, HTTP, POP3 czy IMAP. Można też przechowywać dane na komputerze klienta, za pośrednictwem ciasteczek.

4.4. Mechanizmy przetwarzania danych

Wbudowane mechanizmy przetwarzania danych są kolejną mocną stroną PHP. Sam język programowania jest niemal tak elastyczny jak C (posiada podobną składnię, choć charakteryzuje się słabszą typizacją), a poza tym programista ma do dyspozycji bogatą bibliotekę funkcji. Twórcy PHP starali się dostosować zestaw funkcji tak, żeby odpowiadał niemal każdemu, stąd często te same zadania można wykonać na wiele sposobów. Przykładowo do obsługi wyrażeń regularnych zaimplementowano zestawy funkcji znanych z języka Perl oraz określonych standardem POSIX. Zestaw funkcji został opracowany tak, żeby typowe zadania takie jak np. wysłanie poczty, czy wygenerowanie obrazu dało się bardzo łatwo realizować.

Niestety nic nie jest idealne – zdarza się, że wraz z uaktualnieniem PHP do najnowszej wersji, niektóre funkcje nie są do końca „kompatybilne wstecz”. Poza tym mimo możliwości programowania obiektowego (można definiować klasy, jest dziedziczenie itp.), w PHP nie ma bibliotek obiektowych. Potwierdza to tezę, że PHP jest nadal technologią strukturalną. Brak ten rekompensuje na szczęście możliwość takiego skonfigurowania interpretera, żeby korzystał z biblioteki klas języka Java. Wymaga to oddzielnej instalacji pakietu Java i powoduje spadek wydajności.

4.5. Mechanizmy usprawniające wykrywanie i usuwanie błędów

Napisano wiele debuggerów dla PHP. Istnieją zarówno projekty komercyjne jak i rozprowadzane na bazie licencji GPL. PHP 4 nie posiada natomiast wbudowanych narzędzi do wykrywania błędów.

Język PHP nie posiada również wysokopoziomowych udogodnień takich jak obsługa wyjątków. Programista jest zdany pod tym względem na klasyczną metodę zwracania z funkcji ustalonej wartości w przypadku niepowodzenia (typowo „-1” lub „0”). Z drugiej strony, jak w większości języków skryptowych, w PHP trudniej popełniać błędy – nie ma wskaźników ani niskopoziomowych funkcji dynamicznego przydziału pamięci i innych funkcji, które w języku C sprawiają zwykle kłopoty. W PHP przyjęto strategię, że skrypt musi się wykonać do końca, nawet jeśli w trakcie wykonania wystąpią błędy. Zarówno błędy wykonania, jak i błędy składniowe są sygnalizowane w trakcie wykonywania skryptu poprzez wyświetlenie komunikatu w oknie przeglądarki (faza kompilacji nie istnieje).

4.6. Szybkość działania i zużycie zasobów

Aplikacje tworzone w PHP działają o wiele szybciej, jeśli interpreter PHP jest zintegrowany z serwerem WWW. W przeciwnym wypadku, każdorazowe uruchamianie interpretera wprowadza narzut większy niż w standardowych aplikacjach CGI. Natomiast wykonanie kodu PHP przebiega niemal tak szybko, jak gdyby aplikacja była skompilowana do kodu maszynowego. Dzieje się tak dlatego, że wszystkie funkcje biblioteczne są częścią interpretera, który został napisany w C++ i skompilowany do kodu maszynowego. Proces interpretacji też jest dobrze zoptymalizowany. Zadbano o to, żeby każda instrukcja była interpretowana dokładnie raz, nawet jeśli wykonywana jest więcej razy. Kolejne wykonania tego samego fragmentu programu trwają krócej niż za wykonanie za pierwszym razem.

Zużycie pamięci przez aplikację PHP jest niewielkie. Interpreter zwykle zużywa kilka MB, ale jest ładowany do pamięci tylko raz.

4.7. Przenośność, uniwersalność i dostępność

Aplikacje PHP działają wszędzie, gdzie jest dostępny interpreter tego języka. A interpreter ten jest dostępny praktycznie dla większości serwerów WWW.

W PHP trudno jest pisać nieprzenośne aplikacje. Wbudowane funkcje biblioteczne oferują tak dużą funkcjonalność, że bardzo rzadko zachodzi potrzeba wywoływania funkcji specyficznych dla konkretnego systemu operacyjnego. Jedyne problemy mogą pojawić się z chwilą uaktualnienia wersji interpretera PHP. Pewne funkcje są wycofywane jako przestarzałe i po pewnym czasie starsze programy mogą nie działać.

4.8. Bezpieczeństwo

Bezpieczeństwo każdej aplikacji jest zależne od sposobu jej napisania. Niestety PHP nie ułatwia pisania bezpiecznych aplikacji. Jest to spowodowane dążeniem do uniwersalności i wygody użytkownika biblioteki funkcji. Często jedna funkcja może być użyta w bardzo wielu zastosowaniach. Przykładowo bardzo złą renome ma funkcja „include” służąca do wykonania kodu PHP z innego pliku. Często funkcję tę wykorzystuje się do wstawiania dokumentów HTML, które mają być jakoś zagnieżdżone wewnątrz dokumentu generowanego przez skrypt PHP. Niestety funkcja ta potrafi dużo więcej. Można za jej pomocą wczytać kod PHP z innego źródła niż lokalny dysk twardy np. z obcego serwera! Ataki wykorzystujące tę technikę nazywane są fachowo „cross site scripting” i ich ofiarą padają często nawet duże serwisy tworzone przez profesjonalistów.

Pogorszeniu poziomu bezpieczeństwa sprzyjają też trudności z prawidłowym reagowaniem na błędy – domyślna strategia „ignoruj błąd i idź dalej” nie jest wskazana w aplikacjach, gdzie bezpieczeństwo i stabilność są cechami pożądanymi.

Istnieje również całkiem poważny problem zabezpieczenia haseł dostępu do baz danych. Są one wstawiane do kodu, a ten z kolei musi być czytelny dla interpretera. Niepowołanej osobie wystarczy zdobyć niewielkie uprawnienia (np. w systemie UNIX: nobody), żeby odczytać hasła.

5. ASP – Active Server Pages

ASP to technologia stworzona przez Microsoft realizująca podobne zadania jak PHP i sprawdzająca się w podobnych zastosowaniach. Zasada działania aplikacji ASP jest bardzo podobna do PHP. Interpreter ASP będący częścią serwera WWW (w systemach Windows interpreter jest ładowany jako biblioteka DLL) wczytuje skrypt ASP i przesyła wynik jego działania przeglądarce internetowej. Postać skryptu jest niemalże identyczna jak postać skryptu PHP. Jest to plik HTML z wstawkami kodu napisanego na ogół w języku JScript lub VBScript. Więcej o zasadach tworzenia aplikacji ASP można przeczytać w 10.

5.1. Metodologia projektowania aplikacji

ASP, tak jak PHP, dobrze wspiera programowanie strukturalne, choć VBScript i JScript nie oferują takiej elastyczności jak np. język C, z którego jednak w ASP korzystać nie można. W ASP występuje również problem słabego oddzielenia interfejsu aplikacji od jej logiki. Powoduje to utrudnienia nie tylko przy projektowaniu, ale przede wszystkim przy późniejszej pielęgnacji aplikacji. Są dostępne produkty innych firm, które pozwalają ten problem rozwiązać (podobnie jak w PHP).

Aplikacje ASP mogą być projektowane obiektowo. ASP pozwala definiować klasy oraz korzystać z wbudowanych obiektów, klas i komponentów, które dostarczają odpowiednie usługi. Możliwe jest też stworzenie dodatkowych komponentów w technologii COM/DCOM i wykorzystanie w aplikacji ASP. Dzięki wbudowanej obsłudze sesji można tworzyć obiekty o długim czasie życia. Z drugiej strony zysk z zastosowania klas pisanych w ASP nie jest duży, ponieważ brakuje wielu podstawowych udogodnień takich jak np. dziedziczenie klas.

5.2. Dialog między aplikacją a użytkownikiem

ASP posiada wbudowany mechanizm obsługi sesji. Mechanizm ten wykorzystuje „ciasteczko” do przechowywania identyfikatora sesji, a stan sesji jest przechowywany w wewnętrznych strukturach danych serwera. Programista ma do dyspozycji specjalny obiekt o nazwie „Session”, który zachowuje się jak tablica asocjacyjna przechowująca wartości różnych typów. Zmienne w ASP nie mają określonych typów. Obiekt „Session” może oprócz prostych zmiennych przechowywać inne obiekty.

5.3. Mechanizmy dostępu do danych

Aplikacje ASP mogą łączyć się z systemami baz danych poprzez interfejs ADO (ActiveX Data Objects). ASP posiada wbudowane sterowniki do systemów: MS SQL i MS Access. Może też współpracować z dowolnym systemem baz danych poprzez interfejs ODBC lub za pośrednictwem sterowników dostarczonych przez producenta systemu baz danych. Cechą charakterystyczną interfejsu ADO jest możliwość przetwarzania danych po stronie klienta. Można np. pobrać zawartość części tabeli, a następnie ją zmodyfikować i odesłać do serwera wykorzystując specjalny mechanizm RDS (Remote Data Service).

5.4. Mechanizmy przetwarzania danych

Biblioteka funkcji w ASP nie jest aż tak rozbudowana jak w PHP, ale mimo to oferuje funkcjonalność wystarczającą w wielu typowych zastosowaniach. Przy tworzeniu dużych aplikacji uciążliwy staje się brak typizacji zmiennych. Kod jest przez to dużo mniej czytelny i łatwiej popełniać błędy. Nie można też tworzyć złożonych struktur danych takich jak kolejki, drzewa itp. VBScript i JScript nie udostępniają wskaźników.

5.5. Mechanizmy usprawniające wykrywanie i usuwanie błędów

ASP nie posiada wbudowanych mechanizmów usprawniających wykrywanie błędów. VBScript posiada dyrektywę „On Error”, która działa podobnie jak mechanizm obsługi wyjątków w C++ lub Java, choć wyjątek reprezentuje wartość prosta, a nie obiekt. Nie jest możliwe zaprojektowanie hierarchii błędów. Śledzenie wykonania aplikacji ASP przeprowadza się za pomocą klasycznej metody „printf debugging” znanej z języka C, z tą różnicą, że zamiast funkcji `printf()` wykorzystuje się znacznie mniej elastyczne `Response.Write()`. W sieci Internet można znaleźć bardzo dużo gotowych fragmentów kodu pozwalających wyświetlać na ekranie pewne typowe informacje takie jak np. zawartość „ciasteczek” lub wartość zmiennych sesji.

5.6. Szybkość działania i zużycie zasobów

ASP jest zbudowane w oparciu o technologię COM. Powoduje to, że wywołanie każdej funkcji wbudowanej w ASP wiąże się z dużym narzutem czasowym i pamięciowym. Dodatkowo skrypty ASP są interpretowane (podobnie jak PHP), a interpreter firmy Microsoft interpretuje kod ASP wolniej, niż interpreter

PHP uruchomiony w tym samym systemie operacyjnym. Interpreter ASP dosyć rozrzutnie traktuje też przydzielanie pamięci operacyjnej. Wielokrotne włączenie tego samego pliku jako część składowa aplikacji ASP (za pomocą dyrektywy include) powoduje wielokrotne umieszczenie tego pliku w pamięci. Wady tej nie posiada jedynie interpreter ASP w wersji 5.0 pracujący pod kontrolą systemu Windows 2000 Server.

5.7. Przenośność, uniwersalność i dostępność

ASP jest dostępne jedynie dla serwerów WWW firmy Microsoft. Istnieją implementacje ASP działające z serwerem Apache pod kontrolą systemu operacyjnego Linux lub Unix, jednak nie są one całkowicie zgodne z produktami firmy Microsoft. Mimo że technologia ASP jest rozpowszechniana bezpłatnie, jej wykorzystanie jest obciążone dodatkowymi kosztami ukrytymi, związanymi z koniecznością zakupu systemu operacyjnego MS Windows. Przenośność aplikacji ASP pomiędzy różnymi odmianami systemu MS Windows jest ograniczona przez przenośność komponentów COM, z których aplikacja korzysta. Komponenty te są bowiem pisane zwykle w języku C++ i korzystają ze specyficznych usług systemu operacyjnego. Standardowe komponenty wbudowane w ASP są przenośne.

5.8. Bezpieczeństwo

W aplikacjach ASP można popełnić podobne błędy, co w aplikacjach PHP. Szczególnie należy zachować ostrożność przy konstruowaniu zapytań SQL na podstawie danych uzyskiwanych poprzez formularze⁴ oraz przy wykorzystywaniu dyrektywy include. Oprócz tego istnieje ryzyko wystąpienia błędów w samym interpreterze lub komponentach. Błędy te mogą poważnie naruszyć bezpieczeństwo systemu komputerowego, na którym są uruchamiane aplikacje ASP, ponieważ interpreter i komponenty pracują bezpośrednio pod kontrolą systemu operacyjnego.

6. iHTML

iHTML to komercyjny produkt firmy Inline, Inc. Stanowi on rozszerzenie języka HTML o zestaw znaczników umożliwiających wstawianie do dokumentu treści generowanych dynamicznie. Zestaw ten jest bardzo szeroki i pozwala na automatyzację takich czynności jak manipulacja danymi w relacyjnych bazach danych, generowanie grafiki, czy wykonywanie prostych obliczeń. iHTML posiada też znaczniki odpowiadające podstawowym instrukcjom kontroli przepływu sterowania takim jak instrukcje warunkowe i pętle.

Zasada udostępniania dokumentów iHTML przez WWW jest bardzo podobna do PHP. Strona jest wczytywana przez interpreter zainstalowany jako rozszerzenie serwera WWW. Dokumentacja 10 interpretera jest dosyć obszerna i zawiera również przykłady gotowych prostych aplikacji. Można ją uzyskać ze strony producenta 10 po uprzednim zarejestrowaniu się.

6.1. Metodologia projektowania aplikacji

iHTML słabo wspiera metodologię projektowania zarówno strukturalnego jak i obiektowego. iHTML nie posiada takich możliwości oddzielenia kodu od danych, jak PHP. Nie jest możliwe tworzenie funkcji, procedur ani złożonych struktur danych w tym języku. Aplikację najprościej projektuje się poprzez dodawanie do statycznego serwisu WWW treści generowanych dynamicznie (np. z bazy danych).

⁴ Przede wszystkim nie należy kopiować z sieci Internet gotowych przykładów o charakterze edukacyjnym. Wiele z nich zawiera typowe błędy pozwalające w prosty sposób nieupoważnionym osobom uzyskać dostęp do danych.

6.2. Dialog między aplikacją a użytkownikiem

iHTML nie posiada znaczników automatyzujących realizację sesji. Projektant aplikacji może jednak wykorzystać do tego celu ciasteczka lub przepisywanie adresów.

6.3. Mechanizmy dostępu do danych

iHTML został zaprojektowany szczególnie z myślą o wykorzystaniu relacyjnych baz danych. Potrafi obsługiwać dowolny system, który udostępnia interfejs ODBC. Znaczniki do konstruowania zapytań i prezentacji danych są bardzo łatwe w użyciu i można się ich nauczyć w kilka minut. iHTML nie posiada interfejsów dedykowanych konkretnym systemom baz danych. Dzięki temu możliwe jest przystosowanie wcześniej napisanej aplikacji do pracy z innym systemem, ale z drugiej strony nie jest możliwe korzystanie z systemu bazy danych, jeśli odpowiedni sterownik ODBC nie został zainstalowany.

iHTML potrafi również przechowywać dane w plikach oraz pozyskiwać je za pośrednictwem sieci TCP/IP.

6.4. Mechanizmy przetwarzania danych

W porównaniu z PHP możliwości przetwarzania danych wyglądają dość skromnie. iHTML pozwala automatyzować wiele typowych, prostych czynności takich jak wysłanie poczty, złożenie obrazu z kilku plików, przetworzenie uzyskanych danych z formularza lub zapytania SQL, czy wykonanie prostych obliczeń matematycznych. Można też korzystać z wyrażeń regularnych.

Brak możliwości programowania strukturalnego powoduje jednak, że iHTML sprawdza się tylko w typowych sytuacjach. Nietypowe zadania, dla których algorytmu nie da się zakodować przy pomocy sekwencji kilku prostych instrukcji, implementuje się bardzo trudno. Brakuje podstawowych struktur danych takich jak wektor lub tablica asocjacyjna (w innych językach skryptowych takich jak PHP, Perl czy Python jest to standard). Na szczęście ograniczenie to można pominąć poprzez wprowadzanie własnych znaczników, których znaczenie definiuje się w zewnętrznej bibliotece dołączanej dynamicznie.

6.5. Mechanizmy usprawniające wykrywanie i usuwanie błędów

iHTML posiada kilka mechanizmów pozwalających usprawnić wykrywanie i usuwanie błędów. Za pomocą specjalnego znacznika iERROR można zbudować blok, który działa jak sekcja wyłapywania wyjątków. Jeśli wystąpi błąd, kod zawarty wewnątrz tej sekcji jest wykonywany i program może na tę sytuację zareagować. Drugim mechanizmem jest logowanie komunikatów o błędach. Można tak skonfigurować iHTML, żeby wszystkie informacje o błędach (składniowych i wykonania) były kierowane do określonego pliku tekstowego.

6.6. Szybkość działania i zużycie zasobów

iHTML ma bardzo niewielkie wymagania sprzętowe, mniejsze niż PHP. Interpreter jest zintegrowany z serwerem WWW (działa jako biblioteka dynamicznie dołączana), więc narzuty czasowe nakładane na realizację pojedynczego zapytania HTTP są pomijalnie małe. Należy jednak pamiętać, że kod iHTML jest bezpośrednio interpretowany, więc musi się wykonywać wolniej niż gdyby był skompilowany do postaci natywnej lub pseudokodu. Dlatego przeprowadzanie złożonego przetwarzania w skrypcie iHTML nie jest wskazane.

6.7. Przenośność, uniwersalność i dostępność

iHTML działa na wszystkich serwerach WWW zgodnych ze specyfikacjami NSAPI, ISAPI i WSAPI. iHTML nie jest teoretycznie zależne od platformy sprzętowej ani systemu operacyjnego. Niestety firma Inline nie rozprowadza kodu źródłowego, więc w praktyce instalacja jest jedynie możliwa na tych systemach, dla których dostępne są binaria iHTML tj. na MS Windows, Linux, Solaris, BSDi oraz FreeBSD.

Jeśli aplikacja nie definiuje własnych znaczników, to jej przenośność pomiędzy systemami jest gwarantowana. Na wszystkich platformach wszystkie znaczniki mają zuniifikowane działanie.

6.8. Bezpieczeństwo

Zwykle im prostsza jest dana technologia, tym mniej niesie zagrożeń. Można przypuszczać, że tak też jest w tym przypadku. Rzeczywiście historia zna mniej przypadków wykrycia błędów w interpreterze iHTML niż PHP (może to wynikać też z mniejszej popularności iHTML). Nie oznacza to jednak, że można całkowicie zaufać, że prostota iHTML ochroni przed każdym niebezpieczeństwem. Szczególnie należy zachować ostrożność przy operowaniu wszelkimi danymi uzyskiwanymi z zewnątrz, głównie z formularzy. Napisanie kodu sprawdzającego poprawność tych danych jest w iHTML dużo trudniejszym zadaniem niż skonstruowanie na ich podstawie zapytania SQL, zwłaszcza przy braku możliwości definiowania funkcji. W związku z tym łatwo ulec pokusie niesprawdzania tych danych.

7. ColdFusion

ColdFusion to komercyjny pakiet oprogramowania stworzony przez firmę Macromedia, podobny do iHTML, ale bardziej rozbudowany i o dużo większych możliwościach. ColdFusion jest udostępniane w trzech edycjach: „Developer”, „Standard” i „Enterprise”. Edycja „Developer” jako jedyna jest oferowana bezpłatnie, jednak umożliwia uruchamianie aplikacji ColdFusion jedynie w ograniczonym, testowym środowisku (np. możliwe jest połączenie tylko jednego klienta do aplikacji). Dokumentacja techniczna do wszystkich edycji jest dostępna pod adresem 10.

Zasada działania ColdFusion jest taka sama jak iHTML. Firma Macromedia stworzyła specjalny język CFML (ColdFusion Markup Language), który jest rozszerzeniem języka HTML o dodatkowy zestaw znaczników realizujących różne funkcje. Dokumenty CFML są przetwarzane przez interpreter ColdFusion, który interpretuje wszystkie znaczniki o nazwach zaczynających się od liter „cf”.

7.1. Metodologia projektowania aplikacji

CFML wspiera zarówno programowanie strukturalne jak i obiektowe. Dostarcza znaczniki realizujące podstawowe instrukcje sterujące przepływem (warunki, pętle o określonej i nieokreślonej liczbie powtórzeń, obsługa wyjątków, itp.). Dostarcza też znaczniki pozwalające budować własne funkcje, grupować kod w moduły i komponenty. Programowanie obiektowe w CFML nie sprowadza się jedynie do grupowania funkcji i danych w klasy. W CFML są dostępne jednokrotne dziedziczenie klas, kontrola dostępu do składowych oraz szczegółowe informacje czasu wykonania o typie obiektu. Z drugiej strony brak wskaźników utrudnia tworzenie i używanie złożonych struktur danych, jednak łatwe użycie relacyjnych baz danych pozwala ten problem ominąć. Mimo że CFML nie przypomina „z wyglądu” klasycznego obiektowego języka programowania takiego jak Java czy C++, to do projektowania aplikacji można zastosować metodologię obiektową.

7.2. Dialog między aplikacją a użytkownikiem

ColdFusion posiada wbudowany mechanizm obsługi sesji, pozwala też używać mechanizmu sesji J2EE (omówiony w rozdziale poświęconym JSP i Java Servlets). Standardowy mechanizm sesji udostępnia programiście dwa obiekty „Session” oraz „Client”. Pierwszy z nich przechowuje dane dotyczące użytkownika, drugi dane związane z bieżącą sesją. Dane dotyczące użytkownika mogą być zapisywane w „ciasteczku”, bazie danych albo w plikach na twardym dysku serwera i mogą składać się jedynie ze zmiennych typów prostych. Dane dotyczące sesji są przechowywane w pamięci serwera i mogą być zmiennymi dowolnego typu, również obiektami. Użytkownicy są identyfikowani za pomocą losowo generowanego numeru. ColdFusion używa tych samych identyfikatorów do identyfikowania użytkowników i identyfikowania sesji. Identyfikatory te są generowane losowo przy każdym zapytaniu, które nie zawiera identyfikatora użytkownika.

7.3. Mechanizmy dostępu do danych

ColdFusion potrafi współpracować z dowolnym systemem baz danych dostępnym przez interfejsy ODBC i JDBC. Ponadto posiada wbudowane sterowniki do systemów: Oracle, Sybase Adaptive Server, IBM DB2 UDB, Informix Dynamic Server, Microsoft SQL, Microsoft Access, MySQL, SQLAnywhere i PostgreSQL. Niektóre z tych sterowników są niestety dostępne tylko w wersji „Enterprise”. ColdFusion udostępnia jednolity interfejs służący do korzystania z różnych systemów baz danych.

CFML posiada znaczniki pozwalające korzystać z podstawowych protokołów komunikacyjnych (HTTP, FTP, SMTP, POP3), usług katalogowych (LDAP) oraz z dostępu do plików.

7.4. Mechanizmy przetwarzania danych

ColdFusion posiada wiele użytecznych funkcji ułatwiających przetwarzanie tekstu (np. biblioteka funkcji pozwalających korzystać z wyrażeniach regularnych), sprawdzanie poprawności danych wprowadzanych przez użytkownika oraz prezentowanie wyników w graficznej postaci. Bardzo ważną cechą jest też uwzględnienie różnych standardów kodowania znaków. ColdFusion posiada również zestaw komponentów do przetwarzania dokumentów zapisanych w formacie XML. ColdFusion dobrze współpracuje z technologią Macromedia Flash, dzięki czemu prezentacja danych w przeglądarce nie jest ograniczona jedynie do możliwości języka HTML.

7.5. Mechanizmy usprawniające wykrywanie i usuwanie błędów

Pakiet ColdFusion zawiera zestaw narzędzi pozwalających monitorować stan aplikacji bez ingerencji w jej działanie. Debugger wyświetla informacje dotyczące liczby zapytań, szybkości odpowiedzi aplikacji, aktywności bazy danych, zawartości zmiennych związanych z sesją i wiele innych. Za pomocą specjalnego znacznika można też w aplikacji ustawiać tzw. punkty kontrolne, o których osiągnięciu debugger będzie informował. Istnieje też możliwość weryfikacji, czy kod CFML nie zawiera błędów składniowych ani przestarzałych funkcji, które mogłyby spowodować problemy z przenośnością.

Język CFML posiada mechanizm wyjątków działający podobnie do mechanizmu wyjątków znanego z języka Java czy C++. Umożliwia to elegancko oddzielenie kodu obsługi błędów od logiki aplikacji.

7.6. Szybkość działania i zużycie zasobów

Aby wykorzystać w pełni możliwości ColdFusion, należy zaopatrzyć się w sprzęt o dostatecznie dużej ilości pamięci operacyjnej. Producent zaleca co najmniej 512 MB. Serwer ColdFusion jest bardzo rozbudowany. Zwykle aplikacje wykorzystują niewielką część funkcjonalności pakietu, ale nie da się z ColdFusion usunąć niewykorzystywanych komponentów. Producent jedynie oferuje wybór między bardziej rozbudowaną wersją „Enterprise” i nieco ograniczoną wersją „Standard”.

Natomiast szybkość działania aplikacji ColdFusion jest porównywalna z szybkością wykonywania skryptów PHP. Dzieje się tak dlatego, że język CFML ma bardzo prostą składnię, a wszystkie funkcje są częścią interpretera i wykonują się pod kontrolą systemu operacyjnego, bez pośrednictwa dodatkowych warstw. Stwierdzenie to nie dotyczy oczywiście wykorzystania komponentów wykonanych w innych technologiach takich jak J2EE, CORBA, COM, z którymi ColdFusion może współpracować. ColdFusion umożliwia rozproszenie ruchu sieciowego na więcej niż jedną maszynę. Potrafi też wykorzystać moc obliczeniową maszyn wieloprocesorowych.

7.7. Przenośność, uniwersalność i dostępność

Aplikacje ColdFusion mogą pracować na systemach operacyjnych Windows, Linux, Solaris i HP-UX. W skryptach CFML nie zachodzi na ogół konieczność wywoływania funkcji specyficznych dla systemu operacyjnego (choć istnieje taka możliwość), więc aplikacje dają się łatwo przenosić pomiędzy różnymi systemami. Jedynym wymaganiem jest zainstalowanie pakietu ColdFusion na serwerze WWW. ColdFusion współpracuje z następującymi serwerami WWW: Microsoft IIS, Netscape Enterprise Server, iPlanet SunONE WebServer, Apache. ColdFusion posiada też własny, wbudowany serwer WWW.

7.8. Bezpieczeństwo

ColdFusion pozwala na ograniczenie aplikacji możliwości użycia wybranych znaczników. Dzięki temu można zabronić aplikacji korzystać np. z plików. Takie zabezpieczenie może być utrudnieniem przy przeprowadzaniu ataku typu „cross-site scripting”. ColdFusion posiada też funkcje pozwalające na sprawdzanie poprawności danych dostarczanych za pośrednictwem formularzy. Szczególnie zaleca się ich użycie przy konstruowaniu zapytań SQL.

ColdFusion ułatwia też tworzenie modułów autoryzacji użytkowników. Standardowo obsługuje sposób autoryzacji określony w protokole HTTP, ale oferuje również udogodnienia dla autoryzacji obsługiwanej przez aplikację. Jednym z ważniejszych jest możliwość przyporządkowania określonym użytkownikom odpowiednich, ograniczonych uprawnień.

8. Java Servlets i Java Server Pages

Java Servlets i JSP to technologia umożliwiająca tworzenie aplikacji internetowych w języku Java (w edycji „Enterprise” – J2EE) stworzona przez firmę Sun Microsystems Inc. Język Java powstał dużo wcześniej i sprawdził się w wielu różnych zastosowaniach (aplikacje okienkowe, „aplety” na stronach internetowych, aplikacje dla urządzeń przenośnych itp.). Z pewnością jest to jeden z najbardziej uniwersalnych i najszybciej rozwijających się obiektowych języków programowania dostępnych obecnie na rynku.

U podstaw aplikacji internetowych tworzonych w technologii⁵ Java leżą „serwlety”. „Serwlet” to klasa przystosowana do realizacji żądań HTTP napisana w języku Java. Obiekt tej klasy jest tworzony przy pierwszym żądaniu. Każde następne żądanie jest obsługiwane przez wcześniej utworzony obiekt, poprzez wywołanie jego metody *doGet* lub *doPost*. Metody te przesyłają odpowiedź za pośrednictwem obiektu *Response*. Taki sposób komunikowania się aplikacji z użytkownikiem może być nieco niewygodny, jeśli zachodzi konieczność wysłania dużej ilości statycznych danych, zwykle określających wygląd interfejsu aplikacji. Wtedy kod „serwletu” może składać się w większości z wywołań *response.write()* i stać się nieczytelny. Dla lepszego oddzielenia wyglądu interfejsu od logiki aplikacji, wprowadzono JSP. Aplikacje w JSP tworzy się bardzo podobnie jak w ASP lub PHP. Aplikacja składa się z dokumentów HTML ze wstawkami kodu pisanego w języku Java. Dokumenty te są przekształcane automatycznie do postaci „serwletów”, kompilowane i uruchamiane. Pełna dokumentacja techniczna znajduje się na stronach internetowych firmy Sun (10). Istnieje też wiele przystępnych podręczników w języku polskim (1010) opisujących metody tworzenia internetowych aplikacji w języku Java.

8.1. Metodologia projektowania aplikacji

Aplikacje internetowe w technologii Java projektuje się zwykle stosując metodologię obiektową, ponieważ język Java jest językiem obiektowym. Zwykle kod odpowiedzialny za logikę aplikacji realizuje się w postaci zestawu oddzielnych klas języka Java (na ogół są to komponenty Java Beans), natomiast stronę wizualną interfejsu tworzy się korzystając z JSP. Cechą charakterystyczną dobrze zaprojektowanych aplikacji w języku Java jest ich olbrzymia skalowalność. Java idealnie nadaje się do tworzenia dużych aplikacji przez duże zespoły. Szczegóły projektowania dużych aplikacji w języku Java opisano w 10.

8.2. Dialog między aplikacją a użytkownikiem

Java posiada predefiniowany obiekt *session*, który może przechowywać dane związane z bieżącą sesją. Obiekt *session* nie wymaga inicjowania i jest dostępny zawsze, o ile programista celowo nie wyłączy obsługi sesji. *Session* może przechowywać dane dowolnego typu. Dostęp do nich uzyskuje się poprzez tekstowe identyfikatory – *Session* działa jak tablica asocjacyjna. Przechowywanie obiektów, które mogą być serializowane (tj. mogą zostać zapisane w postaci ciągu bajtów, który następnie może zostać z powrotem przetworzony na obiekt) umożliwia przenoszenie sesji pomiędzy różnymi maszynami, co ułatwia budowanie niezawodnych systemów rozproszonych. Aby obiekty danej klasy można było serializować, klasa musi implementować interfejs *Serializable*, który definiuje metody do zapisu/odczytu obiektu do/ze strumienia.

8.3. Mechanizmy dostępu do danych

Java oferuje jednolity i prosty w użyciu interfejs JDBC do łączenia się z różnymi systemami baz danych. Liczba dostępnych sterowników JDBC do różnych systemów baz danych jest bardzo duża i bardzo szybko rośnie. W chwili pisania tej pracy, Sun Microsystems Inc. dysponował 197 sterownikami. Trudno znaleźć system baz danych, dla którego nie istniałby sterownik.

Java jest uniwersalnym językiem programowania, więc posiada komponenty służące do dostępu do plików, strumieni, gniazd, gniazd SSL, protokołów internetowych (m.in. SMTP, POP3, FTP, HTTP) itp. Biblioteka klas jest bardzo duża. Jej omówienie znacznie wykracza poza zakres tej publikacji.

⁵ Java to nie tylko język programowania, ale również rozbudowany zestaw bibliotek i narzędzi wspomagających projektowanie, implementację i testowanie aplikacji.

8.4. Mechanizmy przetwarzania danych

Java oferuje standardowe mechanizmy przetwarzania danych spotykane w językach wysokiego poziomu. Rozbudowana biblioteka oferuje zarówno klasy implementujące podstawowe struktury danych takich jak drzewa, tablice asocjacyjne, itp. jak i komponenty realizujące złożone przetwarzanie danych w formacie XML z użyciem szablonów XSLT (10). W przeciwieństwie do bibliotek PHP, ASP i ColdFusion, biblioteka klas Java jest biblioteką uniwersalną, nieukierunkowaną na użycie jedynie w aplikacjach internetowych.

8.5. Mechanizmy usprawniające wykrywanie i usuwanie błędów

W JSP i „Serwletach” można korzystać z wielu zaawansowanych metod śledzenia wykonania aplikacji internetowej. Środowisko J2EE umożliwia zarówno śledzenie lokalne jak i zdalne, za pośrednictwem protokołu TCP/IP. Standardowy debugger *jdb* udostępnia bardzo wiele funkcji m.in.: ustawianie punktów przerwań, śledzenie wartości zmiennych, praca krokowa, śledzenie aplikacji wielowątkowych itp. Dostępnych jest też wiele graficznych narzędzi zwiększających wygodę użytkownika *jdb*. Możliwe jest używanie standardowych sposobów śledzenia poprzez wypisywanie różnych komunikatów na konsolę za pomocą polecenia *System.out.print()*. Jak przystało na obiektowy język programowania, w Java nie brakuje też obsługi wyjątków.

8.6. Szybkość działania i zużycie zasobów

Aplikacje Java są kompilowane do pseudokodu, który następnie jest wykonywany przez tzw. maszynę wirtualną Java (JVM). Dzięki temu aplikacje łatwo dają się przenosić między różnymi platformami, ale mają znacznie mniejszą wydajność niż gdyby były skompilowane do kodu maszynowego. Szybkość wykonywania aplikacji jest zależna przede wszystkim od jakości maszyny wirtualnej. Można spotkać różne implementacje JVM. Niektóre jedynie interpretują pseudokod, inne stosują bardziej zaawansowane techniki, polegające zwykle na kompilacji najczęściej wykonywanych fragmentów pseudokodu do kodu maszynowego.

Podobnie jak w przypadku ColdFusion, aby uruchamiać aplikacje internetowe napisane w Java należy wyposażyć maszynę (nie wirtualną, a rzeczywistą) w dużą ilość pamięci operacyjnej. Przeciętnej wielkości aplikacje (np. sklepy internetowe) mogą potrzebować nawet kilkuset MB pamięci. Zwykle aplikacje Java mają tendencje do zapełniania całej wolnej pamięci operacyjnej. W języku Java mechanizm destrukcji obiektów i zwalniania pamięci jest realizowany automatycznie przez tzw. *garbage collector*, który na ogół zaczyna intensywnie działać z chwilą, gdy pamięci zaczyna brakować.

8.7. Przenośność, uniwersalność i dostępność

Pakiet J2EE jest dostępny bezpłatnie i można go wykorzystywać do tworzenia aplikacji bez konieczności wykupywania licencji. Aplikacje pisane w Java (nie tylko te internetowe, ale w ogóle wszystkie aplikacje) dają się uruchamiać na różnych platformach sprzętowych i systemowych bez konieczności rekompilacji kodu źródłowego. Jedynym warunkiem jest dostępność na danej platformie odpowiedniej maszyny wirtualnej. JVM jest dostępne na bardzo wiele systemów operacyjnych m.in. na: MS Windows, Sun Solaris, HP-UX, BSD, Linux, MacOS.

Dla J2EE powstało kilka specjalizowanych serwerów aplikacji (np. JBoss), ale J2EE dobrze się też integruje z serwerami Apache, Microsoft IIS, Netscape Enterprise Server czy SunONE WebServer.

8.8. Bezpieczeństwo

Dzięki istnieniu maszyny wirtualnej, projektant aplikacji internetowych ma możliwość selektywnego określenia uprawnień aplikacji do poszczególnych zasobów systemowych. Opis przywilejów jest osobną częścią aplikacji. Ograniczenie uprawnień pozwala zmniejszyć skutki ewentualnej awarii aplikacji. Java jest językiem specjalnie zaprojektowanym tak, aby utrudnić popełnianie błędów, które często występowały w aplikacjach C++. Przede wszystkim wskaźniki zastąpiono „inteligentnymi” referencjami, zrezygnowano z niskopoziomowych łańcuchów zakończonych znakiem „0”, uproszczono zarządzanie pamięcią, zabezpieczono mechanizm rzutowania typów. Podwyższyło to nie tylko stabilność i bezpieczeństwo aplikacji, ale również uprościło proces implementacji (a wiadomo – im coś jest prostsze tym zwykle mniej podatne na usterki).

9. ASP.NET – Active Server Pages .NET

Technologia ASP.NET ma niewiele wspólnego z technologią ASP. Obydwie mają podobne zastosowania, skrót „ASP” w nazwie oraz zostały stworzone przez tę samą firmę, na tym jednak podobieństwa się kończą. ASP.NET jest częścią systemu .NET, który stanowi uniwersalną platformę programistyczną wspierającą różne języki programowania, różne platformy sprzętowe i różne systemy operacyjne. Z założenia .NET ma spełniać podobne założenia co Java. Zasada działania wszystkich aplikacji .NET jest podobna do działania aplikacji Java. Kod źródłowy pisany w jednym z wielu języków programowania wspieranych przez .NET (zwykle C#, C++, VB, Java) jest kompilowany do pseudokodu, który następnie zostaje wykonany na maszynie wirtualnej CLR (Common Language Runtime).

Aplikacja ASP.NET to specjalna odmiana aplikacji .NET przystosowana do pracy jako aplikacja internetowa. Kod źródłowy aplikacji jest podzielony na dwie części: kompilowaną i interpretowaną. W części kompilowanej umieszcza się funkcje odpowiedzialne za logikę aplikacji (dostęp do danych, przetwarzanie) oraz obsługę zdarzeń pochodzących od interfejsu użytkownika, a w części interpretowanej – kod definiujący wygląd interfejsu. Tak jak w przypadku „serwletów” Java, aplikacja ASP.NET zostaje uruchomiona przy pierwszym żądaniu HTTP, a później tylko oczekuje na zdarzenia generowane przez interfejs użytkownika przesyłane przez przeglądarkę. Zagadnienia tworzenia aplikacji w ASP.NET zostały omówione w 10.

9.1. Metodologia projektowania aplikacji

ASP.NET jest jedyną technologią z omawianych w tej pracy, która pozwala na projektowanie graficznego interfejsu użytkownika w metodologii obiektowo-zdarzeniowej, tj. tak jak projektuje się aplikacje okienkowe za pomocą współczesnych narzędzi typu RAD takich jak Delphi, Kylix, QTDesigner czy JBuilder. Zadaniem programisty nie jest implementacja obsługi zapytania HTTP, a obsługa zdarzeń. Informacja o zdarzeniu jest przesyłana poprzez protokół HTTP, ale dekodowanie parametrów, ustalenie źródła zdarzenia oraz metody obiektu, która powinien na zdarzenie zareagować, jest dla programisty przezroczyste. Zdarzeniami są na ogół kliknięcia myszką w hiperłącza lub przyciski na stronie WWW.

Kolejną ważną cechą ASP.NET jest zwolnienie twórców interfejsu graficznego z konieczności kodowania interfejsu w języku HTML i pisania skryptów działających po stronie klienta. ASP.NET oferuje zestaw kontrolki WebForms, z których buduje się interfejs. Kontrolki te same generują odpowiedni dla danej przeglądarki kod HTML. Dzięki temu aplikacja ma szansę prawidłowo działać w większej liczbie różnych przeglądarek. Oczywiście użycie gotowych komponentów znacznie skraca czas produkcji aplikacji.

9.2. Dialog między aplikacją a użytkownikiem

W ASP.NET mechanizm obsługi sesji jest niewidoczny dla twórcy aplikacji. Każda sesja jest widoczna jako oddzielna, niezależna instancja aplikacji z własnymi danymi, które nie są kasowane po zakończeniu obsługi zdarzenia. W rzeczywistości mechanizm obsługi sesji jest realizowany poprzez umieszczanie w kodzie wygenerowanej strony WWW zakodowanej informacji o stanie aplikacji. Informacja ta jest przesyłana przez przeglądarkę z każdym żądaniem HTTP.

9.3. Mechanizmy dostępu do danych

Aplikacje ASP.NET mogą korzystać z relacyjnych systemów baz danych za pośrednictwem interfejsu ADO.NET. Interfejs ten jest nowszą wersją interfejsu ADO i pod względem funkcjonalności istotnie się od niego nie wyróżnia. Został dostosowany do obiektowych realiów programowania (np. wykorzystuje wyjątki do sygnalizacji błędów), umożliwia też korzystanie z puli połączeń w celu zwiększenia wydajności.

9.4. Mechanizmy przetwarzania danych

Platforma .NET została zaprojektowana z myślą o tworzeniu w niej aplikacji o różnych zastosowaniach, dlatego dostępne mechanizmy przetwarzania danych są bardzo rozbudowane. Biblioteka standardowa dostarcza wiele gotowych do wykorzystania komponentów, co znacznie skraca czas produkcji oprogramowania (podobnie jak w J2EE).

9.5. Mechanizmy usprawniające wykrywanie i usuwanie błędów

Z pakietem Visual Studio .NET firmy Microsoft jest dostarczany graficzny debugger, który pozwala śledzić dowolną aplikację ASP.NET, zarówno uruchomioną lokalnie, jak i na zdalnym serwerze. Możliwości debugera są zbliżone do możliwości debuggerów dostępnych dla J2EE.

Ponieważ w .NET można programować w wielu językach programowania, są dostępne różne konstrukcje językowe pozwalające tworzyć kod obsługi błędów. W językach C++ C# i Java jest obsługa wyjątków, w VB podobna trochę, ale niehierarchiczna konstrukcja „*ON ERROR*”.

9.6. Szybkość działania i zużycie zasobów

Wyniki testów dostępnych na stronie 10 wskazują, że aplikacje ASP.NET są ponad dwukrotnie szybsze od aplikacji stworzonych w J2EE. W testach nie podano jednak, które serwery aplikacyjne J2EE były testowane. .NET wykorzystuje technikę zwaną JIT (Just In Time), polegającą na zapamiętywaniu kodu maszynowego, na który zostały przełożone instrukcje pseudokodu w chwili ich pierwszego wykonania. W ten sposób każde następne ich wykonanie trwa krócej, ponieważ nie trzeba wykonywać czasochłonnej interpretacji. Technika JIT nie pozwala osiągnąć takiej wydajności, jak aplikacje kompilowane od razu do kodu maszynowego.

Zapotrzebowanie na pamięć operacyjną przez aplikacje ASP.NET oraz środowisko CLR jest na ogół większe niż J2EE. Maszyna wirtualna, nie dość, że musi mieć załadowany do pamięci kod wszystkich komponentów wykorzystanych przez aplikacje, to na dodatek w trakcie wykonywania pseudokodu musi zapamiętywać odpowiadający mu kod maszynowy.

9.7. Przenośność, uniwersalność i dostępność

.NET jest platformą zamkniętą. Microsoft nie udostępnia kodów źródłowych bibliotek ani narzędzi. Platforma .NET jest przeznaczona do pracy w różnych odmianach systemu operacyjnego Microsoft Windows. Istnieją wprawdzie śmiało projekty przeniesienia jej na inne systemy (np. projekt MONO – [1]), jednak oprogramowanie zapewne długo jeszcze nie będzie w pełni zgodne z oficjalną dystrybucją .NET. Dlatego aplikacje ASP.NET mogą być przenośne jedynie pomiędzy różnymi wersjami systemu Windows.

Zupełną nowością, która odróżnia platformę .NET od np. J2EE jest możliwość tworzenia aplikacji w różnych językach programowania. Nawet jedna aplikacja może składać się z wielu fragmentów napisanych każdy w innym języku. Dzięki temu można lepiej dopasować język programowania do konkretnego problemu oraz łatwiej znaleźć pracowników o odpowiednich kwalifikacjach. Niestety dowolność ta dotyczy jedynie składni języka programowania a nie dostępnych bibliotek. Zestaw bibliotek musiał zostać ujednoczony.

9.8. Bezpieczeństwo

Nad bezpieczeństwem i stabilną pracą aplikacji .NET czuwa całe środowisko uruchamiania tych aplikacji tj. CLR wraz z zestawem odpowiednich komponentów. W .NET są dostępne komponenty, za pomocą których w bardzo prosty i wygodny sposób można zbudować moduł autoryzacji użytkowników. Można też grupować użytkowników systemu przydzielając im odpowiednie role i uprawnienia. Rzeczywisty stopień bezpieczeństwa aplikacji zbudowanej z takich komponentów jest zależny jedynie od ich jakości. Ponieważ .NET jest platformą zamkniętą, nie ma teoretycznie możliwości sprawdzenia, jaka jest ta jakość. Można jedynie polegać na opiniach innych użytkowników oraz publikowanych w sieci Internet informacjach o znalezionych błędach.

10. Podsumowanie

Zagadnienie tworzenia aplikacji internetowych jest bardzo rozległe, o czym świadczy duża liczba dostępnych technologii i duży stopień ich zróżnicowania. Z tego powodu wybór technologii i narzędzi służących do stworzenia aplikacji/systemu jest bardzo istotnym elementem fazy strategicznej i krytycznym dla powodzenia przedsięwzięcia. Przy wyborze należy mieć na uwadze wiele istotnych czynników takich jak: koszty zakupu odpowiednich licencji, koszty szkoleń oraz zatrudnienia odpowiednich specjalistów, dostępność gotowych komponentów oraz dokumentacji, dostępność kodu źródłowego, wsparcie techniczne, skalowalność, wydajność, bezpieczeństwo danej technologii itp. Szczegółowe omówienie wszystkich tych zagadnień znacznie wykracza poza zakres tej pracy. Większość danych dostępne jest w literaturze oraz dokumentacji technicznej konkretnych produktów.

Jednak na podstawie tego pobieżnego przeglądu można wstępnie określić, które technologie będą lepiej sprawdzały się w pewnych zastosowaniach, a które gorzej:

- **CGI** – Nadaje się do tworzenia prostych aplikacji, w zastosowaniach, w których nie jest wymagany wysoki poziom bezpieczeństwa ani wydajność oraz wszędzie tam, gdzie nie jest dostępna żadna inna technologia.
- **FastCGI** – Doskonale sprawdza się wszędzie tam, gdzie liczy się bardzo krótki czas odpowiedzi aplikacji i niewielkie zużycie zasobów. Koszty rozwoju dużych aplikacji mogą być wysokie ze względu na brak gotowych, standardowych komponentów realizujących typowe zadania.

- **PHP** – Najpopularniejsza obecnie technologia służąca do tworzenia małych aplikacji internetowych, ze względu na swoją prostotę, bogaty zestaw funkcji, dosyć wysoką wydajność i dobrą przenośność. Dużą zaletą jest dostępność kodu źródłowego całego pakietu. Trudności pojawiają się przy zespołowym tworzeniu i pielęgnacji rozbudowanych aplikacji.
- **ASP** – Technologia dobrze działająca jedynie na systemach operacyjnych firmy Microsoft. Generalnie podobna do PHP, ale oferuje mniejszy zestaw funkcji oraz niższą wydajność.
- **iHTML** – Zestaw prostych narzędzi do szybkiego przetwarzania statycznych stron WWW w niewielkie aplikacje internetowe.
- **ColdFusion** – Technologia o podobnej funkcjonalności i wydajności co PHP, ale oferuje więcej możliwości zabezpieczania systemu przed intruzami. Wadą są większe wymagania sprzętowe.
- **J2EE (Java Servlets i JSP)** – Technologia przystosowana do tworzenia rozbudowanych systemów (często rozproszonych) przez duże zespoły. Oferuje wysoki poziom bezpieczeństwa, dużą liczbę gotowych komponentów oraz niespotykaną w innych technologiach przenośność kodu. Niezbyt dobrze sprawdza się w małych aplikacjach ze względu na niską wydajność oraz złożony proces konfiguracji.
- **ASP.NET** – Technologia konkurencyjna do J2EE pod względem zakresu zastosowań. Pozwala na bardzo szybkie tworzenie złożonych interfejsów użytkownika dzięki programowaniu zdarzeniowemu. Jako jedyna oferuje możliwość tworzenia kodu w wielu językach programowania. Z drugiej strony aplikacje są łatwo przenośne jedynie pomiędzy systemami operacyjnymi firmy Microsoft.

Widać, że żadna z tu wymienionych technologii nie jest idealna. Większość z nich jest stale rozwijana i być może za niedługi czas część przedstawionych tu informacji stanie się z tego powodu nieaktualna.

Bibliografia

- [1] Alur, D., Crupi, J., Malks, D., *J2EE. Wzorce projektowe*, wyd. Helion, 2003
- [2] Colburn, R. *CGI - Teach Yourself CGI Programming in a Week*, edycja polska, wyd. Helion, 1998
- [3] Foristal, J., Traxler, J., *Hack Proofing Your Web Applications*, edycja polska, wyd. Helion, 2003
- [4] Goodwill, J., *Java Server Pages. Podręcznik z przykładami*, wyd. Helion, 2001
- [5] Hougland, D., Tavistock, A., *JSP – tworzenie stron WWW*, Wyd. RM, 2002
- [6] Inline Internet Systems, Inc., *User's Guide to iHTML Extensions Version 2.20*, 2001
- [7] Kim, E., E., *Programowanie CGI – przewodnik*, Oficyna Wydawnicza LT&P Sp. z o.o., 1996
- [8] Kochmer, C., Frandsen, E., *JSP i XML*, wyd. Helion, 2003
- [9] Mitchell, S., *Active Server Pages 3.0 dla każdego*, wyd. Helion, 2001
- [10] Worley, S., *ASP.NET. Vademecum Profesjonalisty*, wyd. Helion, 2003
- [11] <http://java.sun.com/j2ee/>
- [12] <http://msdn.microsoft.com/netframework/technologyinfo/overview/default.aspx>
- [13] <http://www.fastcgi.com/>
- [14] <http://www.ihtml.com/>
- [15] <http://www.macromedia.com/support/documentation/en/coldfusion/index.html>
- [16] <http://www.php.net/>