Analiza porównawcza technologii tworzenia aplikacji internetowych dla baz danych Oracle

Marek Wojciechowski, Maciej Zakrzewicz
Politechnika Poznańska, Instytut Informatyki
ul. Piotrowo 3a, 60-965 Poznań
e-mail: {marek,mzakrz}@cs.put.poznan.pl

Abstrakt. Projektanci aplikacji internetowych dla baz danych Oracle mają do dyspozycji wiele różnych technologii, począwszy od specyficznych dla platformy Oracle rozwiązań opartych o język PL/SQL, poprzez uniwersalne technologie Java, silnie wspierane przez narzędzia programistyczne i serwery aplikacji Oracle, a skończywszy na językach skryptowych takich jak PHP czy Perl, bardzo popularnych w środowisku Linux. Celem niniejszego artykułu jest przedstawienie zalet i wad wymienionych technologii oraz porównanie ich funkcjonalności i efektywności w kontekście współpracy z bazą danych Oracle.

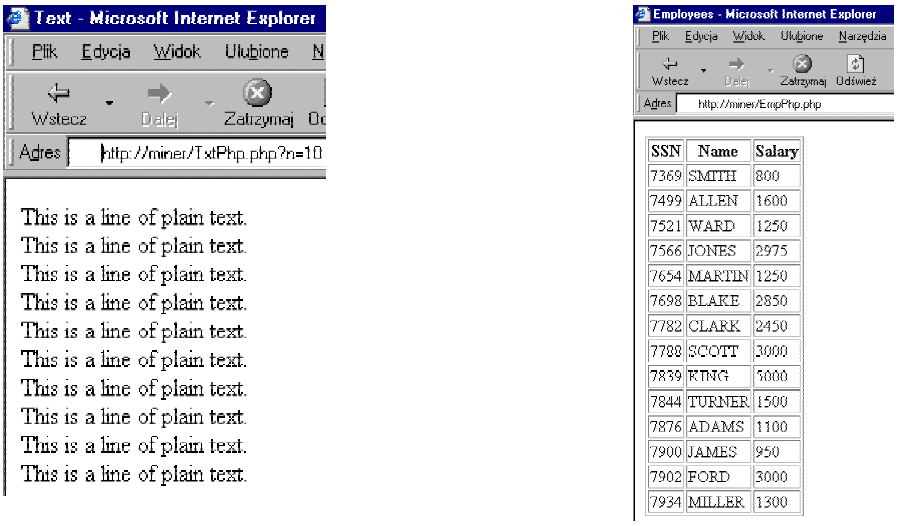
1. Wprowadzenie

Podstawowe znaczenie dla funkcjonowania aplikacji internetowych mają technologie służące do dynamicznego generowania stron WWW (najczęściej w postaci dokumentów HTML). Projektanci aplikacji internetowych dla baz danych Oracle mają w tym zakresie do dyspozycji wiele różnych technologii, z których każda posiada swoje silne i słabe strony. Ponieważ główną platformą zaprojektowaną z myślą o uruchamianiu aplikacji internetowych komunikujących się z bazą danych Oracle jest Oracle9i Application Server (Oracle9iAS), naturalny wybór stanowią technologie mające podstawowe znaczenie dla tej platformy, a do tego silnie wspierane przez narzędzia programistyczne oferowane przez firmę Oracle. Należą do nich technologie bazujące na języku Java (serwlety i JSP) oraz technologie oparte o procedury składowane w PL/SQL. Interesującą alternatywę, przynajmniej w zakresie prostych aplikacji, stanowią języki skryptowe takie jak PHP czy Perl, szczególnie popularne w środowisku Linux, ale dostępne również dla wielu innych systemów operacyjnych (np. Microsoft Windows). Oracle9iAS (aktualna wersja 9.0.2) w standardowych instalacjach nie obsługuje PHP, zawiera natomiast moduł obsługujący język Perl wraz z dużą liczbą bibliotek w tym języku. Uzupełnienie Oracle9iAS o obsługę PHP jest możliwe i wymaga dodania darmowego modułu PHP na takiej samej zasadzie, na której dodaje się moduł PHP do serwera Apache (podstawowym komponentem Oracle9iAS jest Oracle HTTP Server bazujący na technologii serwera Apache).

Celem niniejszego artykułu jest przedstawienie wspomnianych wyżej technologii, porównanie ich możliwości i wydajności oraz krótkie omówienie zagadnień ich współpracy z bazą danych Oracle. Ponieważ każdej z omawianych technologii można by poświęcić obszerny podręcznik, w artykule wymienione zostaną jedynie podstawowe zalety i wady poszczególnych technologii, a styl programowania pokazany będzie na przykładzie implementacji dwóch prostych aplikacji. Pierwsza z aplikacji to program generujący dokument zawierający zadaną za pomocą przekazywanego parametru ‘n’ liczbę linii z tekstem „This is a line of plain text” (Rys. 1a), druga przedstawia w postaci tabelarycznej zawartość tabeli EMP z bazy danych Oracle (Rys. 1b). Te same aplikacje zostaną wykorzystane do przetestowania i porównania wydajności opisywanych technologii.

Jak już wspomniano wcześniej, podstawową platformą dla aplikacji internetowych komunikujących się z bazą danych Oracle jest Oracle9iAS, oferujący zaawansowane mechanizmy buforowania, rozkładania obciążenia oraz skalowalność. Oracle9iAS jest jednak kosztowny, a w przypadku małych systemów o niewielkim przewidywanym obciążeniu oferowana przez niego zaawansowana funkcjonalność nie musi być wymagana. Dlatego przy omawianiu poszczególnych technologii, tam gdzie istnieje taka możliwość, opisane zostaną alternatywne, darmowe platformy

umożliwiające uruchamianie aplikacji tworzonych w danej technologii, choć nie oferujące pełnej funkcjonalności platformy Oracle9iAS.



a) b)

Rys. 1. Wyniki działania omawianych przykładowych aplikacji generujących dokumenty HTML

2. Procedury składowane PL/SQL i PL/SQL Server Pages (PSP)

Aplikacje internetowe w PL/SQL działają na zasadzie wywoływania poprzez serwer WWW procedur składowanych w bazie danych, generujących dokumenty za pomocą specjalnie do tego celu przeznaczonych pakietów. Podstawowe znaczenie mają procedury `htp.print` i `htp.prn` wysyłające dane przekazane im jako argument do przeglądarki (pierwsza z procedur na końcu dodaje znak nowej linii). Ponadto, dostępne są procedury generujące poszczególne znaczniki HTML oraz różne procedury użytkowe np. wyświetlająca zawartość tabeli. Ponieważ cała procedura składowana wykonuje się po dołączeniu się do bazy danych (najczęściej korzystając z parametrów podanych w jednym z plików konfiguracyjnych serwera WWW) w sposób naturalny w procedurze mogą być zagnieżdżane polecenia SQL. Wszystkie akcje wykonujące się w ramach procedury stanowią transakcję, która jest automatycznie zatwierdzana gdy procedura wykona się pomyślnie (jeśli nie są wykorzystywane zaawansowane mechanizmy transakcyjne Oracle9iAS).

Ponieważ połączenie się z bazą danych jest operacją czasochłonną, moduł serwera WWW odpowiedzialny za aplikacje PL/SQL domyślnie buforuje połączenia z bazą danych na poziomie procesów serwera obsługujących żądania użytkowników. Jeśli żądanie wywołania procedury trafi do procesu, który już posiada wcześniej otwarte połączenie z bazą danych, którego wymaga dana procedura, nie ma konieczności nawiązywania nowego połączenia. Należy w tym miejscu podkreślić, że tego typu mechanizmy „odzysku" połączeń są również wykorzystywane w innych technologiach. Różne są szczegóły implementacyjne związane z buforowaniem połączeń (patrz kolejne rozdziały), niezmienne pozostają jednak zalety tego mechanizmu i niebezpieczeństwa związane z jego stosowaniem. Zaletą jest redukcja czasu potrzebnego na połączenie z bazą danych, szczególnie widoczna gdy wszyscy użytkownicy korzystają z tego samego konta w bazie. Niebezpieczeństwa wiążą się z potencjalnie dużą liczbą otwartych połączeń z bazą danych ze strony serwera WWW (szczególnie gdy różni użytkownicy logują się na różne konta w bazie) - należy brać pod uwagę możliwości serwera bazy danych i odpowiednio dobrać maksymalną liczbę współbieżnych procesów serwera WWW.

Wygodnym sposobem tworzenia procedur składowanych generujących tekstowe strony WWW (najczęściej w języku HTML) jest przygotowywanie ich w postaci dokumentów PL/SQL Server Pages (PSP). Dokument PSP ma postać dokumentu HTML, w którym za pomocą specjalnych

znaczników zagnieżdżone są fragmenty kodu w PL/SQL generujące dynamiczne części dokumentu, najczęściej w oparciu o dane pobrane z bazy danych. Idea zagnieżdżania kodu programu w „statycznych" dokumentach została zaczerpnięta z technologii ASP (opracowanej przez Microsoft) i ułatwia wykorzystanie narzędzi projektanckich HTML do obróbki dokumentów częściowo generowanych dynamicznie. Dokument PSP musi zostać załadowany do bazy danych pod postacią procedury składowanej w PL/SQL narzędziem `loadpsp`.

Technologia PL/SQL jest specyficzna dla platformy Oracle. Aplikacje internetowe zaimplementowane jako procedury składowane PL/SQL poza serwerem aplikacji Oracle9iAS (w każdym z typów instalacji), mogą być uruchamiane również poprzez Oracle HTTP Server dostarczany wraz z serwerem bazy danych Oracle9i. Na uruchamianie procedur PL/SQL składowanych w bazie danych, generujących strony WWW, poprzez Oracle HTTP Server pozwala standardowo z nim dostarczany moduł mod_plsql. W zakresie narzędzi programistycznych i projektanckich Oracle, PL/SQL jako język tworzenia aplikacji internetowych jest wspierany przez Oracle Designer i Oracle Portal (wcześniej WebDB).

Analiza kodu przykładowych aplikacji:

Poniżej przedstawiono kod dwóch przykładowych aplikacji (po lewej jako kod źródłowy procedury, po prawej jako dokument PSP). W procedurach, do generacji poszczególnych znaczników HTML wykorzystane zostały dedykowane procedury biblioteczne (można je zawsze zastąpić przez `htp.print`), w dokumentach PSP cała część statyczna dokumentu występuje jako „czyste" znaczniki HTML, PL/SQL wykorzystany jest wyłącznie do prezentacji informacji odczytanych z bazy danych. Pierwsza z aplikacji pokazuje sposób korzystania w aplikacjach PL/SQL z parametrów wywołania (przekazanych zgodnie ze standardem CGI, metodą GET lub POST). Parametry te muszą być zadeklarowane, po ich deklaracji są automatycznie dostępne w kodzie programu jako parametry procedury składowanej przekazane w trybie IN.

```
create or replace procedure TxtPlsql(n number) is
begin
 htp.htmlOpen;
 htp.headOpen;
 htp.title('Text');
 htp.headClose;
 htp.bodyOpen(cattributes=>'BGCOLOR="#FFFFFF"');
 for i in 1..n
    loop
        htp.print('This is a line of plain text.');
        htp.br;
    end loop;
 htp.bodyClose;
 htp.htmlClose;
end;
/
```

```
<%@ page language="PL/SQL" %>
<%@ plsql parameter="n" type="number" %>
<HTML>
<HEAD><TITLE>Text</TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<% for i in 1..n loop %>
  This is a line of plain text<BR>
<% end loop; %>
</BODY>
</HTML>
```

```
create or replace procedure EmpPlsql is
begin
 htp.htmlOpen;
 htp.headOpen;
 htp.title('Employees');
 htp.headClose;
 htp.bodyOpen(cattributes=>'BGCOLOR="#FFFFFF"');
 htp.tableOpen('border=1');
 htp.tableRowOpen;
 htp.tableHeader('SSN');
 htp.tableHeader('Name');
 htp.tableHeader('Salary');
 htp.tableRowClose;
 for kursor in (SELECT empno, ename, sal FROM emp)
         loop
            htp.tableRowOpen;
            htp.tableData(kursor.empno);
            htp.tableData(kursor.ename);
            htp.tableData(kursor.sal);
            htp.tableRowClose;
         end loop;
 htp.tableClose;
 htp.bodyClose;
 htp.htmlClose;
end;
/
```

```
<%@ page language="PL/SQL" %>
<HTML>
<HEAD><TITLE>Employees</TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<TABLE BORDER=1><TR><TH>SSN</TH>
<TH>Name</TH><TH>Salary</TH></TR>

<% for kursor in
(SELECT empno, ename, sal FROM emp) loop %>
  <TR>
  <TD><%= kursor.empno %></TD>
  <TD><%= kursor.ename %></TD>
  <TD><%= kursor.sal %></TD>
  </TR>
<% end loop; %>

</TABLE>
</BODY>
</HTML>
```

Zalety technologii PL/SQL:

- oparcie o znany większości programistom aplikacji dla baz danych Oracle język PL/SQL
- łatwość zagnieżdżania poleceń SQL
- duża efektywność operacji bazodanowych

Wady technologii PL/SQL:

- technologia specyficzna dla platformy Oracle - ograniczona przenaszalność
- język PL/SQL, zorientowany na przetwarzanie bazodanowe, nie jest najwygodniejszym rozwiązaniem przy skomplikowanej logice aplikacji

3. Serwlety Java i JavaServer Pages (JSP)

Serwlety i JSP są to aplikacje w języku Java, uruchamiane po stronie serwera WWW, dynamicznie generujące strony WWW. Serwlety i JSP stanowią część technologii Java 2 Platform Enterprise Edition (J2EE), platformy dla złożonych aplikacji działających w oparciu o architekturę trójwarstwową, odpowiadając za prezentację danych i interakcję z użytkownikiem. Serwlety i JSP posiadają wszystkie zalety języka Java: przenaszalność, bezpieczeństwo, bogactwo bibliotek. Dostęp do baz danych z serwletów i JSP, podobnie jak z innych aplikacji Java, realizowany jest w oparciu o standardy JDBC i SQLJ (w chwili obecnej bardziej rozpowszechniony jest JDBC). Oba te standardy oferują przenaszalność aplikacji i podobną funkcjonalność, umożliwiając m.in. wykonywanie zapytań, instrukcji DML i DDL, wywoływanie procedur składowanych i wielokrotne wykorzystywanie prekompilowanych poleceń SQL. Przetwarzanie transakcyjne w standardach JDBC i SQLJ jest możliwe w 2 trybach: z automatycznym zatwierdzaniem poszczególnych poleceń lub z jawnym zatwierdzaniem sekwencji operacji stanowiących transakcję, po jej zakończeniu (zaawansowane mechanizmy związane z przetwarzaniem transakcyjnym definiuje standard JTA, wspierany przez Oracle9iAS). Zaletą JDBC jest elastyczność, z kolei SQLJ oferuje lepszą kontrolę błędów na etapie translacji. Niezależnie od wykorzystywanego standardu dla operacji bazodanowych, zarówno w serwletach jak i w JSP możliwe jest wykorzystanie mechanizmu buforowania połączeń z bazą danych.

W środowisku Oracle9iAS (wersja 9.0.2) serwlety i JSP, jako część J2EE, uruchamiane są przez komponent Oracle Containers for J2EE (OC4J), któremu Oracle HTTP Server przekazuje żądania za pośrednictwem modułu mod_oc4j. Komponent OC4J dostępny jest w każdym z typów instalacji Oracle9iAS. Alternatywnym środowiskiem dla serwletów i JSP jest Oracle HTTP Server instalujący się wraz z serwerem bazy danych Oracle9i, wyposażony w moduł jserv. Jednakże jserv bazuje na starszych specyfikacjach serwletów i JSP niż OC4J, przez co może nie obsługiwać niektórych mechanizmów. Darmową platformą zgodną z najnowszymi specyfikacjami jest kombinacja Apache/Tomcat (Tomcat stanowi referencyjną implementację specyfikacji serwletów i JSP). Należy jednak podkreślić, że możliwości OC4J znacznie przewyższają jserv i Tomcat, gdyż OC4J oprócz serwletów i JSP wspiera również inne standardy obejmowane przez J2EE. W zakresie narzędzi programistycznych i projektanckich oferowanych przez Oracle, technologie Java (w tym serwlety i JSP) wspierane są przez JDeveloper.

3.1. Serwlety Java

Serwlet jest programem tworzonym jako klasa dziedzicząca z klasy `javax.servlet.http.HttpServlet`. Przy pierwszym uruchomieniu serwletu jest wywoływana jego metoda `init`, następnie serwlet pozostaje aktywny w pamięci serwera, obsługując żądania użytkowników metodą `service` (metoda ta może być wywołana wiele razy równolegle - dla każdego wywołania tworzony jest oddzielny wątek). W zależności od typu żądania kierowanego do serwletu, metoda `service` wywołuje metody `doGet`, `doPost`, itd. Gdy serwlet jest usuwany

z pamięci, np. przy zatrzymaniu serwera WWW, wywoływana jest metoda `destroy`. Wspomniane metody składają się na tzw. cykl życia serwletu i determinują sposób implementacji serwletów. Metoda `init` jest odpowiednim miejscem do przygotowania zasobów, które będą wykorzystywane przy obsłudze wielu żądań. Często stosowaną praktyką jest otwieranie współużytkowanych połączeń z bazą danych w metodzie `init`. Połączenia otwarte w metodzie `init`, są dostępne w metodach `doGet`, `doPost`, itd., a powinny być zamykane dopiero w metodzie `destroy`. Jest to oszczędne i wydajne rozwiązanie, które powinno być stosowane gdy przemieszanie operacji wykonywanych w wyniku obsługi różnych żądań w ramach jednego połączenia z bazą danych jest dopuszczalne. W przypadku gdy wskazane jest odseparowanie operacji bazodanowych wykonywanych w ramach obsługi różnych żądań, w celu redukcji kosztu nawiązywania połączenia z bazą danych można wykorzystać pule połączeń (dostępne również z JSP - patrz rozdział 3.2).

Analiza kodu przykładowych aplikacji:

Poniżej przedstawiono przykładowy serwlet generujący zadaną liczbę linii tekstu. Zwraca uwagę konieczność generacji na tej samej zasadzie (`out.println`) statycznych jak i dynamicznych fragmentów dokumentu. Odczyt parametrów wywołania serwletu jest realizowany poprzez obiekt klasy `HttpServletRequest` przekazywany przez środowisko serwera jako parametr metod `doGet`, `doPost`, itd.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class TxtServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException
  {
    int n = Integer.parseInt(request.getParameter("n"));
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<HTML>" + "<HEAD>" + " <TITLE>" + "Text" + "</TITLE>" + "</HEAD>");
    out.println("<BODY BGCOLOR=\"#FFFFFF\">");

    for (int i=0; i<n; i++)
      out.println("This is a line of plain text.<BR>\n");

    out.println("</BODY>" + "</HTML>");
  }
}
```

Poniżej przedstawiono dwa warianty serwletu przedstawiającego zawartość tabeli EMP. W pierwszym wariancie otwierane jest nowe połączenie z bazą danych przy każdym wywołaniu serwletu (w metodzie `doGet`). Druga wersja otwiera połączenie w metodzie `init`, dzięki czemu jest ono później wykorzystywane przy obsłudze wszystkich zgłaszanych żądań. W obu wersjach operacje bazodanowe są realizowane w standardzie JDBC.

```
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class EmpServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException
  {
    Connection conn = null;

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<HTML>"
        + "<HEAD>"
        + " <TITLE>" + "Employees" + "</TITLE>"
        + "</HEAD>"
        + "<BODY BGCOLOR=\"#FFFFFF\">");

    try
```

```
    {
      Class.forName("oracle.jdbc.OracleDriver");
      conn = DriverManager.getConnection("jdbc:oracle:oci:@miner", "scott", "tiger");
    }
    catch (Exception e) {}

    if (conn != null)
    {
      try
      {
        Statement stmt = conn.createStatement();
        ResultSet rset = stmt.executeQuery("SELECT empno, ename, sal FROM emp");
        out.println("<TABLE BORDER=1><TR><TH>SSN</TH><TH>Name</TH><TH>Salary</TH></TR>");
        while (rset.next())
          out.println("<TR><TD>" + rset.getInt(1)+ "</TD><TD>" + rset.getString(2) + "</TD><TD>"
          + rset.getDouble(3) + "</TD></TR>" );
        out.println("</TABLE>");
        rset.close();
        stmt.close();
      }
      catch (SQLException e)
      {
        out.println(e);
      }
    }

    out.println("</BODY>"
        + "</HTML>");

    if (conn != null)
    {
      try
      {
        conn.close();
      }
      catch (SQLException e) {}
    }
  }
}
```

```
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class EmpServletP extends HttpServlet {

  Connection conn;

  public void init(ServletConfig config)
      throws ServletException
  {
    super.init(config);
    try
    {
      Class.forName("oracle.jdbc.OracleDriver");
      conn = DriverManager.getConnection("jdbc:oracle:oci:@miner", "scott", "tiger");
    }
    catch (Exception e) {}
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<HTML>"
        + "<HEAD>"
        + " <TITLE>" + "Employees" + "</TITLE>"
        + "</HEAD>"
        + "<BODY BGCOLOR=\"#FFFFFF\">");

    if (conn != null)
    {
      try
      {
        Statement stmt = conn.createStatement();
        ResultSet rset = stmt.executeQuery("SELECT empno, ename, sal FROM emp");
        out.println("<TABLE BORDER=1><TR><TH>SSN</TH><TH>Name</TH><TH>Salary</TH></TR>");
        while (rset.next())
          out.println("<TR><TD>" + rset.getInt(1)+ "</TD><TD>" + rset.getString(2) + "</TD><TD>"
          + rset.getDouble(3) + "</TD></TR>" );
        out.println("</TABLE>");
        rset.close();
        stmt.close();
```

```
      }
      catch (SQLException e)
      {
        out.println(e);
      }
    }

    out.println("</BODY>"
        + "</HTML>");
  }

  public void destroy()
  {
    if (conn != null)
    {
      try
      {
        conn.close();
      }
      catch (SQLException e) {}
    }
  }
}
```

Zalety serwletów:

- oparte o uniwersalny, stosunkowo szybki, bezpieczny, oferujący przenaszalność i bogactwo bibliotek język Java
- część technologii J2EE - odpowiedni wybór dla złożonych aplikacji biznesowych

Wady serwletów:

- przemieszanie kodu odpowiedzialnego za statyczną i dynamicznie generowaną zawartość
- stosunkowo duży rozmiar kodu niezbędny do realizacji prostych funkcji

3.2. JavaServer Pages (JSP)

Technologia JSP jest odpowiedzią na podstawowy problem związany z wykorzystywaniem serwletów Java: brak izolacji pomiędzy kodem wykonywalnym a definicją szaty graficznej generowanego dokumentu. JSP nie jest całkowicie nowym rozwiązaniem – jest to rodzaj nakładki na popularną technologię serwletów Java. Tworzenie aplikacji JSP polega na przygotowywaniu tekstowych dokumentów HTML, wewnątrz których zanurzany jest kod Java (za pomocą znaczników w stylu ASP), służący do konstruowania części dynamicznych. Przy pierwszym odwołaniu do dokumentu JSP, serwer WWW dokonuje translacji dokumentu do postaci serwletu, kompiluje, a następnie uruchamia wynikowy serwlet. Przy kolejnych odwołaniach do już skompilowanego dokumentu JSP, uruchamiany jest wynikowy serwlet.

Mimo że technologia JSP prowadzi do uruchamialnych aplikacji w postaci serwletów Java, styl programowania jest w przypadku JSP nieco inny niż w przypadku serwletów. Po pierwsze, programista nie ma dostępu do metod `init` i `destroy` wynikowego serwletu. W uproszczeniu można powiedzieć, że programista tworzy metodę `service` i ma możliwość ustawiania zmiennych na poziomie serwletu. Dlatego też współużytkowanie zasobów używanych w poszczególnych wywołaniach dokumentu JSP odbywa się na innych zasadach niż w przypadku serwletów. W dokumentach JSP można w wygodny sposób (za pomocą odpowiednich znaczników) specyfikować zasięg używanych obiektów, przez co obiekty mogą być współdzielone na poziomie dokumentu, żądania, sesji lub wszystkich dokumentów składających się na złożoną aplikację. Często wykorzystywanym mechanizmem redukcji kosztu połączenia z bazą danych w JSP jest wykorzystanie tzw. puli połączeń wspólnej dla wszystkich dokumentów składających się na aplikację.

Istotną zaletą JSP jest możliwość niemal całkowitego odseparowania kodu programu od statycznych części dokumentu. Umieszczenie kodu w obrębie specjalnych znaczników w dokumencie HTML nie pozwala jeszcze na niezależną pracę projektantów statycznych fragmentów stron i programistów, dlatego JSP umożliwia znacznie więcej w zakresie separacji kodu. Po pierwsze, możliwe jest umieszczenie kodu Java w klasach zewnętrznych w stosunku do

dokumentu, a następnie wykorzystywanie obiektów tych klas poprzez specjalne znaczniki, jako tzw. komponenty JavaBeans. Po drugie, możliwe jest oprogramowanie własnego zestawu znaczników w postaci tzw. biblioteki znaczników (ang. tag library). Wykorzystanie biblioteki znaczników pozwala na tworzenie dokumentów JSP nie zawierających żadnych fragmentów kodu w języku Java. Umieszczanie kodu Java w klasach zewnętrznych w stosunku do dokumentu JSP poprawia jego czytelność i jest szczególnie zalecane gdy rozmiar wymaganego kodu jest duży. (W przedstawionych poniżej przykładach dokumentów JSP, ze względu na ich prostotę, nie wykorzystano strategii przenoszenia kodu Java do klas zewnętrznych.)

Analiza kodu przykładowych aplikacji:

Poniżej przedstawiono przykładowy dokument JSP generujący zadaną liczbę linii tekstu. Odczyt parametrów wywołania dokumentu jest realizowany poprzez obiekt `request` (klasy `HttpServletRequest`) automatycznie udostępniany przez środowisko serwera.

```
<%@ page language="java" %>
<HTML>
<HEAD>
<TITLE>Text</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<%
    int n = Integer.parseInt(request.getParameter("n"));
    for (int i=0; i<n; i++) {
%>
      This is a line of plain text.<BR>
<%
    }
%>
</BODY>
</HTML>
```

Poniżej przedstawiono dwa warianty dokumentu JSP przedstawiającego zawartość tabeli EMP. W pierwszym wariancie otwierane jest nowe połączenie z bazą danych przy każdym odwołaniu do dokumentu. Druga wersja wykorzystuje pulę połączeń jako komponent JavaBean o zasięgu całej aplikacji (wspólny dla wszystkich dokumentów JSP uruchamianych na serwerze w ramach danego kontekstu aplikacji). Dzięki wykorzystaniu tego mechanizmu nowe połączenie jest fizycznie otwierane tylko gdy w puli nie ma wolnego połączenia o takich samych parametrach. W obu wersjach dokumentu operacje bazodanowe są realizowane w standardzie JDBC.

```
<%@ page language="java" import="java.sql.*" %>
<HTML>
<HEAD>
<TITLE>Employees</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<%
   try
    {
      Class.forName("oracle.jdbc.OracleDriver");
      Connection conn = DriverManager.getConnection("jdbc:oracle:oci:@miner", "scott", "tiger");
      Statement stmt = conn.createStatement();
      ResultSet rset = stmt.executeQuery("SELECT empno, ename, sal FROM emp");
%>
      <TABLE BORDER=1><TR><TH>SSN</TH><TH>Name</TH><TH>Salary</TH></TR>

      <% while (rset.next()) { %>
        <TR>
        <TD><%= rset.getInt(1) %></TD>
        <TD><%= rset.getString(2) %></TD>
        <TD><%= rset.getDouble(3) %></TD>
        </TR>
      <% } %>

      </TABLE>
<%
      rset.close();
      stmt.close();
      conn.close();
    }
    catch (Exception e)
    {
      out.println(e);
    }
```

```
%>
</BODY>
</HTML>
```

```
<%@ page language="java" import="java.sql.*, javax.sql.*, oracle.jdbc.pool.* " %>

<jsp:useBean id="ods" class="oracle.jdbc.pool.OracleConnectionCacheImpl"
scope="application" />

<HTML>
<HEAD>
<TITLE>Employees</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<%
   try
    {
      ods.setURL("jdbc:oracle:oci:@miner");
      ods.setUser("scott");
      ods.setPassword("tiger");
      ods.setMaxLimit(150);
      Connection conn = ods.getConnection ();
      Statement stmt = conn.createStatement();
      ResultSet rset = stmt.executeQuery("SELECT empno, ename, sal FROM emp");
%>
      <TABLE BORDER=1><TR><TH>SSN</TH><TH>Name</TH><TH>Salary</TH></TR>

      <% while (rset.next()) { %>
        <TR>
        <TD><%= rset.getInt(1) %></TD>
        <TD><%= rset.getString(2) %></TD>
        <TD><%= rset.getDouble(3) %></TD>
        </TR>
      <% } %>

      </TABLE>
<%
      rset.close();
      stmt.close();
      conn.close();
    }
    catch (Exception e)
    {
      out.println(e);
    }
%>
</BODY>
</HTML>
```

Zalety JSP:

- oparte o uniwersalny, stosunkowo szybki, oferujący przenaszalność i bogactwo bibliotek język Java
- część technologii J2EE - odpowiedni wybór dla złożonych aplikacji biznesowych
- duże możliwości w zakresie separacji kodu Java od statycznych treści dokumentów

Wady JSP:

- w przypadku bardzo prostych aplikacji, w których nie jest wykorzystywana moc języka Java, wadą jest duży rozmiar kodu w porównaniu z językami skryptowymi

4. Perl

Język Perl (skrót od Practical Extraction and Report Language) jest bezpłatnym językiem skryptowym dostępnym na prawie wszystkich architekturach i systemach operacyjnych. Składnia języka Perl zawiera elementy takich języków jak C, awk, sed, grep i Bourne shell, co czyni go doskonałym narzędziem do obróbki tekstu. Zaletą Perla jest bogactwo dostępnych modułów bibliotecznych, w tym modułów ułatwiających pisanie w Perlu skryptów CGI. Język Perl był jednym z pierwszych języków, który odegrał dużą rolę jako narzędzie tworzenia aplikacji WWW. Szczególnie nadaje się do aplikacji wykonujących wiele złożonych operacji na łańcuchach znaków i przetwarzających dane z plików tekstowych. Składnia języka Perl jest bardzo elastyczna i pozwala zaawansowanym programistom na stosowanie wielu „sztuczek" znacznie skracających rozmiar kodu. Duża elastyczność składni często jest jednak postrzegana jako istotna wada języka, gdyż

może prowadzić do tworzenia nieczytelnych programów (odbywają się nawet konkursy na najbardziej nieczytelny program w języku Perl!).

Operacje bazodanowe są w Perlu realizowane poprzez moduł DBI. DBI stanowi uniwersalny interfejs do baz danych definiujący pewne konwencje, zmienne i metody. DBI pośredniczy między aplikacją a modułami DBD dedykowanymi dla poszczególnych systemów baz danych, wykonującymi w rzeczywistości operacje na bazie danych. Za współpracę aplikacji w języku Perl z bazą danych Oracle odpowiada moduł DBD::Oracle. Interfejs DBI (w oparciu o moduł DBD::Oracle) umożliwia wykonywanie zapytań, instrukcji DML i DDL, wywoływanie procedur składowanych, przetwarzanie transakcyjne i wielokrotne wykorzystywanie prekompilowanych poleceń SQL. Możliwe jest w Perlu korzystanie z mechanizmu buforowania połączeń z bazą danych w środowisku serwera Apache/mod_perl (na poziomie procesów serwera WWW). Umożliwia to moduł serwera Apache o nazwie Apache::DBI, który po jego uaktywnieniu, buforuje wszystkie połączenia z bazami danych wykorzystywane przez aplikacje w języku Perl.

W środowisku Oracle9iAS skrypty Perl uruchamiane są przez moduł mod_perl, w który wyposażony jest Oracle HTTP Server (wraz z bogatą kolekcją modułów bibliotecznych od wersji 9.0.2, w tym DBI i DBD::Oracle). Moduł mod_perl jest powszechnie dostępny i może być wykorzystywany w połączeniu z oryginalnym serwerem Apache (dla niego był opracowywany) lub serwerem Oracle HTTP Server dostarczanym z serwerem bazy danych Oracle9i (wchodzi w skład instalacji, ale z mniejszą liczbą modułów bibliotecznych np. brak DBI i DBD::Oracle w Oracle9i Release 1 & 2). Wykorzystując do uruchamiania programów w Perlu inne środowisko niż Oracle9iAS 9.0.2, należy liczyć się z koniecznością ściągnięcia z Internetu potrzebnych modułów bibliotecznych, aby możliwy był dostęp do baz danych Oracle. Programy w Perlu mogą być również uruchamiane na serwerach WWW jako skrypty CGI, ale rozwiązanie to cechuje się słabszą wydajnością niż w przypadku mod_perl i nieoszczędnym gospodarowaniem zasobami.

Analiza kodu przykładowych aplikacji:

Poniżej przedstawiono kod dwóch przykładowych aplikacji w języku Perl. Skrypt w lewej kolumnie wykorzystuje moduł biblioteczny CGI, aby funkcją `param` odczytać wartość przekazanego parametru ‘n’, określającego liczbę linii tekstu do wyświetlenia. Skrypt w prawej kolumnie wyświetla zawartość tabeli EMP korzystając z interfejsu DBI (to czy połączenie z bazą danych wykorzystywane w skrypcie będzie buforowane czy nie, zależy od konfiguracji serwera).

```
use CGI qw/:standard/;

print "Content-type: text/html\n\n";
print
"<HTML><HEAD><TITLE>Text</TITLE></HEAD><BODY>";

$n = param('n');

for ($i=0; $i<$n; $i++) {
  print "This is a line of plain text.<BR>\n";
}
print "</BODY></HTML>";
```

```
use DBI;

print "Content-type: text/html\n\n";
print
"<HTML><HEAD><TITLE>Employees</TITLE></HEAD><BODY>";

my $dbh =
 DBI->connect('DBI:Oracle:miner', 'scott', 'tiger');

my $sth;
my @data;

$sth =
 $dbh->prepare('SELECT empno, ename, sal FROM emp');

$sth->execute;

print "<TABLE BORDER=1><TR>”;
print
"<TH>SSN</TH><TH>Name</TH><TH>Salary</TH></TR>";

while (@data = $sth->fetchrow_array()) {
  print "<TR><TD>$data[0]</TD><TD>$data[1]</TD>";
  print "<TD>$data[2]</TD></TR>\n";
}
print "</TABLE>\n";

$sth->finish;
$dbh->disconnect;

print "</BODY></HTML>";
```

Zalety języka Perl:

- silne wsparcie dla przetwarzania łańcuchów znaków
- łatwość operacji na plikach tekstowych
- bogactwo powszechnie dostępnych modułów bibliotecznych

Wady języka Perl:

- skomplikowana i bardzo elastyczna składnia - łatwo o tworzenie nieczytelnego kodu
- powtórne wykorzystywanie połączeń z bazami danych konfigurowane na poziomie całego serwera, a nie oddzielnie dla poszczególnych połączeń

5. PHP

PHP (PHP: Hypertext Preprocessor) jest językiem skryptowym ogólnego przeznaczenia, rozwijanym na zasadach Open Source, szczególnie popularnym w środowisku Linux, ale dostępnym również na innych platformach np. Microsoft Windows. PHP został opracowany z myślą o aplikacjach WWW, dlatego też do ich tworzenia szczególnie dobrze się nadaje. Najczęściej PHP jest zagnieżdżany w dokumentach HTML, jednakże standardowo używane są do tego celu inne znaczniki niż w przypadku ASP, JSP czy PSP (można włączyć styl znaczników ASP za pomocą jednej z opcji konfiguracyjnych PHP). PHP jest łatwy w opanowaniu, przynajmniej w zakresie podstawowej funkcjonalności. Jego składnia przypomina języki C i Java, pewne mechanizmy, szczególnie dotyczące operacji na łańcuchach znaków, zostały zapożyczone z Perla.

Za jedną z najmocniejszych stron PHP uważane jest wsparcie dla około 20 systemów zarządzania bazami danych poprzez zestawy funkcji bibliotecznych dedykowanych dla poszczególnych systemów (funkcje dla Oracle działają w oparciu o interfejs OCI). Ponadto PHP obsługuje standard ODBC oraz wprowadza uniwersalny interfejs dbx dla operacji bazodanowych, ułatwiający przenaszalność aplikacji, niestety w tej chwili (PHP 4.2.2) nieobejmujący systemu Oracle. Funkcje biblioteczne dla Oracle umożliwiają wykonywanie zapytań, instrukcji DML i DDL, wywoływanie procedur składowanych, przetwarzanie transakcyjne i wielokrotne wykorzystywanie prekompilowanych poleceń SQL. PHP oferuje bardzo wygodny sposób korzystania z mechanizmu buforowania połączeń z bazą danych. Połączenia są buforowane na poziomie procesów serwera WWW (tak jak np. w Perlu), jednakże to programista decyduje które połączenia mają być buforowane, wybierając odpowiednią funkcję biblioteczną. PHP dostarcza kilka wariantów funkcji nawiązującej połączenie z bazą danych: jedna z nich otwiera zawsze nowe połączenie, które ma być zamknięte po zakończeniu pracy skryptu; inna otwiera tzw. stałe połączenie (ang. persistent connection), które jest buforowane na poziomie procesu serwera WWW.

Aby możliwe było uruchamianie programów w PHP w środowisku Oracle9iAS należy wyposażyć Oracle HTTP Server w moduł mod_php dla serwera Apache (Oracle HTTP Server bazuje na architekturze serwera Apache). Moduł ten jest darmowy, dostępny w Internecie w postaci źródłowej oraz binarnej dla Windows. Rozszerzenie Oracle HTTP Server o wsparcie dla PHP odbywa się tak samo jak w przypadku serwera Apache - należy dobrać wersję modułu odpowiednią dla wersji serwera Apache, na której bazuje posiadany Oracle HTTP Server. Oczywistą i darmową alternatywą dla rozszerzania Oracle9iAS, jest wykorzystanie oryginalnego Apache'a z modułem mod_php. Programy w PHP mogą być również uruchamiane na serwerach WWW jako skrypty CGI, ale rozwiązanie to cechuje się słabszą wydajnością niż w przypadku mod_php i nieoszczędnym gospodarowaniem zasobami (stosowanie stałych połączeń nie przynosi żadnych korzyści - połączenia zawsze są zamykane po zakończeniu pracy skryptu).

Analiza kodu przykładowych aplikacji:

Poniżej przedstawiono kod przykładowych aplikacji w PHP. Skrypt po lewej stronie generuje dokument HTML z zadaną liczbą linii tekstu. Skrypt ten działa w oparciu o założenie, że włączona jest opcja konfiguracyjną PHP, która wymusza rejestrowanie wszystkich parametrów przekazanych

do skryptu jako zmienne globalne (w tym wypadku mechanizm ten obejmuje zmienną `$n`). Alternatywnym sposobem odczytu przekazanych parametrów jest odwołanie się do stosownej tablicy asocjacyjnej (alternatywny sposób odczytu parametru 'n' w naszym przykładzie to `$HTTP_GET_VARS['n']`).

Prawa kolumna przedstawia skrypt wyświetlający zawartość tabeli EMP. W tej postaci nie jest wykorzystywane buforowanie połączeń z bazą danych. Aby włączyć ten mechanizm, poprzez żądanie nawiązania stałego połączenia, należałoby zastąpić wywołanie funkcji `OCINLogon` przez `OCIPLogon`.

```
<HTML>
<HEAD>
<TITLE>Text</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">

<?php
for ($i=0; $i<$n; $i++) {
  echo "This is a line of plain text.<BR>\n";
}
?>

</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>Employees</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">

<TABLE BORDER=1><TR><TH>SSN</TH>
<TH>Name</TH><TH>Salary</TH></TR>
<?php
$link = OCINLogon ("scott", "tiger", "miner");

$select = 'SELECT empno, ename, sal FROM emp';

$stmt = OCIParse($link, $select);
OCIExecute($stmt);
while (OCIFetch($stmt))
{
  echo "<TR><TD>" . OCIResult($stmt,1)
  . "</TD><TD>"
  . OCIResult($stmt,2) . "</TD><TD>"
  . OCIResult($stmt,3) . "</TD></TR>\n";
}

OCIFreeStatement($stmt);
OCILogoff($link);
?>

</TABLE>

</BODY>
</HTML>
```

Zalety PHP:

- oszczędne gospodarowanie zasobami serwera i duża szybkość działania
- technologia darmowa (Open Source) doskonale współpracująca z Linux/Apache - pozwala na redukcję kosztu serwera aplikacji
- język łatwy do opanowania w zakresie tworzenia prostych aplikacji
- silne wsparcie dla przetwarzania łańcuchów znaków i plików tekstowych

Wady PHP:

- na platformie Windows ciągle nie tak stabilny jak na platformie Unix/Linux
- mimo wsparcia dla ogromnej liczby systemów zarządzania bazami danych, uniwersalny interfejs obsługuje tylko niektóre z nich, co może utrudniać przenaszalność aplikacji

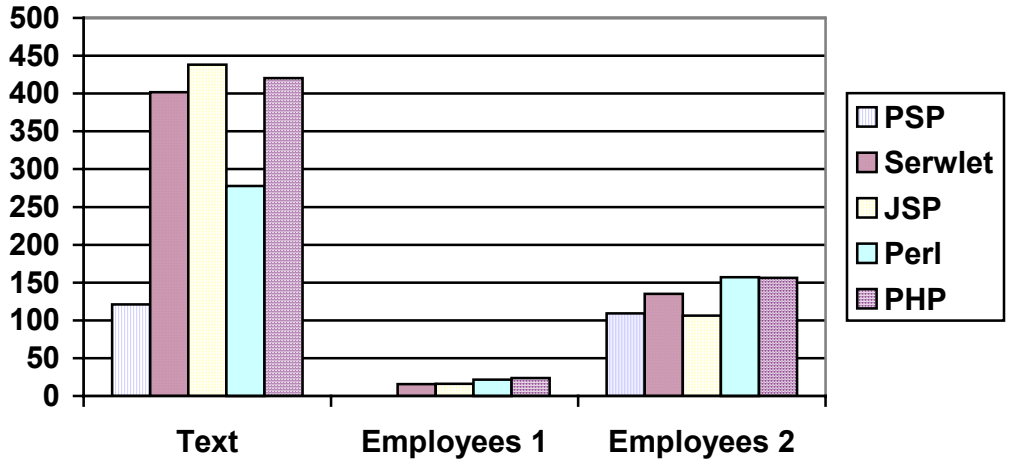
6. Porównanie wydajności omawianych technologii

Jednym z podstawowych kryteriów oceny technologii internetowych jest szybkość działania tworzonych w nich aplikacji. W celu weryfikacji wydajności oferowanej przez omawiane technologie, przeprowadzona została seria eksperymentów na komputerze klasy PC wyposażonym w 2 procesory Intel Pentium III 933MHz i 1GB pamięci RAM, pracującym pod kontrolą systemu operacyjnego Linux (wersja SuSE 7.2). Jako serwer aplikacji wykorzystany został Oracle9iAS Release 2 (9.0.2) rozszerzony o moduł mod_php 4.2.2. W czasie trwania eksperymentów komponent WebCache był skonfigurowany tak, aby nie buforował testowanych aplikacji. Pozostałe

moduły były wykorzystane w domyślnych konfiguracjach, uzyskanych w wyniku standardowej instalacji.

Testowana była efektywność działania opisanych we wcześniejszych rozdziałach aplikacji przykładowych. Aplikacja generująca powtarzający się tekst wywoływana była z parametrem n=170, co w wyniku dawało plik HTML o rozmiarze ok. 6 kB. Rozmiar wynikowego dokumentu w przypadku aplikacji bazodanowej wynosił ok. 1.5 kB. Aplikacje bazodanowe (poza aplikacjami PL/SQL) były testowane w wersjach/konfiguracjach korzystających i niekorzystających z mechanizmów „odzysku" połączeń z bazą danych.

Do symulacji obciążenia i pomiaru efektywności przetwarzania (mierzonej jako liczba obsłużonych żądań na sekundę) wykorzystane zostało narzędzie OpenLoad (dostępne pod adresem http://openload.sourceforge.net/). Wybrany został model obciążenia, w którym w każdej chwili 120 użytkowników równolegle wymaga obsługi ze strony serwera (konfiguracja Oracle9iAS zakładała maksymalnie 150 równolegle obsługiwanych żądań). Dla każdej implementacji zmierzono średnią liczbę obsługiwanych żądań na sekundę w czasie potrzebnym na obsłużenie 5000 żądań. Wyniki pomiarów pokazuje Rys. 2. (Text – aplikacja generująca linie stałego tekstu, Employees 1 – zawartość tabeli EMP bez „odzysku" połączeń, Employees 2 – zawartość tabeli EMP z „odzyskiem" połączeń).



Rys. 2. Liczba obsługiwanych żądań na sekundę dla aplikacji testowych zaimplementowanych w różnych technologiach

Wyniki eksperymentów pokazują, że w przypadku aplikacji generujących HTML bez odwoływania się do bazy danych, największą wydajność wykazują technologie Java (serwlety i PSP) oraz PHP, a zdecydowanie najwolniejszy jest PL/SQL. Z kolei w przypadku aplikacji bazodanowych PL/SQL jest najlepszym rozwiązaniem gdy implementacje w językach Java, Perl i PHP nie wykorzystują mechanizmu „odzysku" połączeń z bazą danych (który domyślnie jest wykorzystywany przez moduł mod_plsql), ponieważ w przypadku aplikacji realizujących proste zapytania i pobierających niewiele rekordów, nawiązanie połączenia z bazą jest najbardziej czasochłonną operacją. Wykorzystanie mechanizmów „odzysku" połączeń w serwletach, JSP, PHP i Perlu pozwala jednak na osiągnięcie wydajności równej lub przewyższającej wydajność PL/SQL, nawet w kontekście aplikacji, których działanie sprowadza się do operacji bazodanowych. Serwlety i JSP cechują się bardzo zbliżoną wydajnością, co nie jest zaskoczeniem, gdyż technologia JSP sprowadza się do serwletów. Istotna różnica wydajności aplikacji zaimplementowanych w tych technologiach wystąpiła jednak w przypadku aplikacji bazodanowej z „odzyskiem" połączeń. Różnica była spowodowana wykorzystaniem innej techniki „odzysku" połączeń z bazą danych.

7. Podsumowanie

W zakresie wsparcia przez oprogramowanie Oracle przewagę wykazują technologie PL/SQL i Java, wspierane zarówno przez serwery Oracle jak i narzędzia programistyczne. Perl jest w pełni wspierany przez Oracle9iAS, natomiast wykorzystywanie PHP na tej platformie wymaga dodania modułu mod_php dla Apache do Oracle HTTP Server.

W przypadku dużych i złożonych systemów dobrym rozwiązaniem są serwlety Java i JSP, osadzone w technologii J2EE. Dla małych systemów, szczególnie przy ograniczonym budżecie, z pewnością wartą rozważenia alternatywą jest darmowy, a bardzo szybki, PHP. PL/SQL jest dobrym rozwiązaniem tylko dla aplikacji bardzo intensywnie korzystających z bazy danych, jednakże pozostałe technologie są w stanie w tym zakresie mu dorównać pod warunkiem wykorzystania mechanizmów „odzysku" połączeń. Perl w obszarze aplikacji internetowych jest wypierany przez PHP, oferujący podobną funkcjonalność przy bardziej czytelnej składni i większej efektywności przetwarzania. Zaletą Perla jest natomiast bogactwo modułów bibliotecznych.

Gdy generowane dokumenty mają charakter w przeważającej części statyczny, z niewielkimi dynamicznie generowanymi wstawkami, słusznym wyborem spośród technologii Java z pewnością będzie JSP, spośród technologii PL/SQL - PSP, a z języków skryptowych - PHP. Technologie te pozwalają bowiem na tworzenie aplikacji w formie dokumentów HTML zawierających fragmenty kodu programu ograniczone specjalnymi znacznikami. Rozwiązanie takie ułatwia korzystanie z graficznych edytorów HTML przy opracowywaniu w części dynamicznych stron WWW.

Podsumowując, trudno jest wskazać zdecydowanego zwycięzcę spośród przedstawionych technologii. Ostateczny wybór zawsze zależeć będzie od rozmiaru i charakteru tworzonej aplikacji, umiejętności posiadanych przez programistów, przewidywanych możliwości przenoszenia aplikacji na inne platformy, a także od posiadanych środków, które można przeznaczyć na serwer aplikacji. Niekiedy istotna może być możliwość korzystania ze wsparcia technicznego firmy Oracle. Wreszcie w konkretnych, specyficznych zastosowaniach, czynnikiem decydującym może być dostępność na danej platformie gotowego modułu bibliotecznego realizujące potrzebne funkcje.

Bibliografia

1. http://jakarta.apache.org/tomcat/ (Tomcat)
2. http://java.sun.com/beans (JavaBeans)
3. http://java.sun.com/j2ee/ (J2EE)
4. http://java.sun.com/products/jdbc/ (JDBC)
5. http://java.sun.com/products/jsp/ (JSP)
6. http://java.sun.com/products/servlet/ (Java servlets)
7. http://msdn.microsoft.com/asp/ (ASP)
8. http://openload.sourceforge.net/ (OpenLoad)
9. http://technet.oracle.com/docs/ (Oracle Documentation)
10. http://www.apache.org/ (Apache)
11. http://www.oracle.com/ip/deploy/ias/ (Oracle iAS)
12. http://www.perl.com/ (Perl)
13. http://www.php.net/ (PHP)
14. http://www.sqlj.org/ (SQLJ)