
C Přístup k databázím z WWW

C.1. Úvod	2
C.2. PHP (Personal Home Page)	4
C.3. Internet Information Server	9
C.3.1. Active Server Pages (ASP)	9
C.4. Přístup k databázím z jazyka Java	17
C.4.1. Rozhraní JDBC	18
C.4.2. SQLJ	25

C.1. Úvod

- pojem transakční aplikace na WWW (L.Wienczszak (Sybase) - CW 24/97)

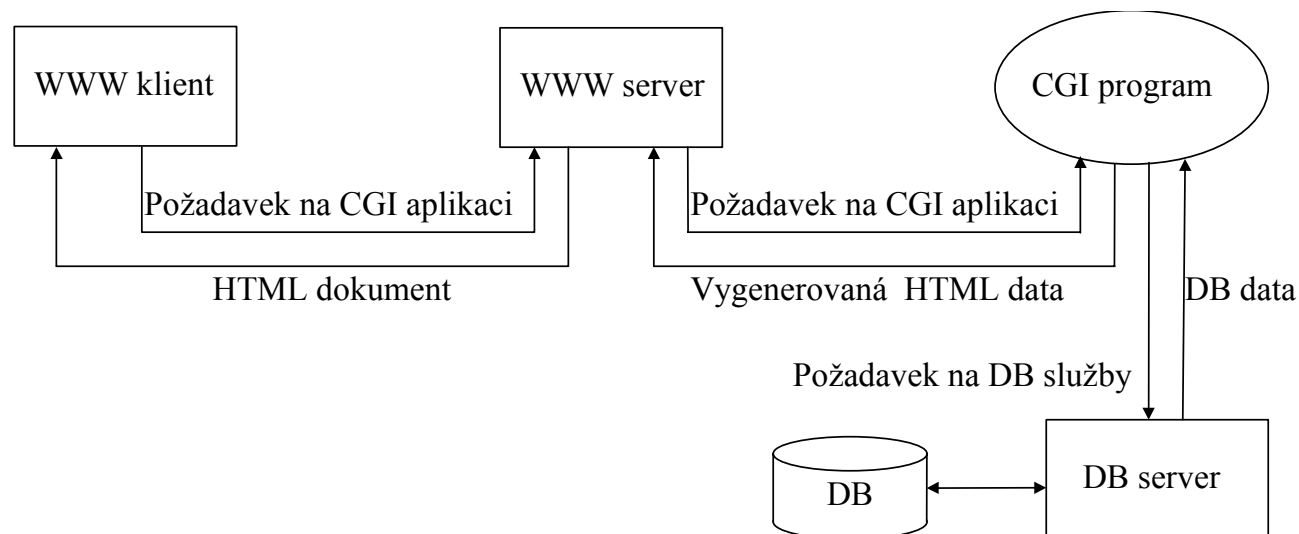
	uživatel čte	uživatel sám zapisuje
statické WWW	„obyčejné“ WWW stránky	jednoduché dotazníky
dynamické WWW	zpravodajské servery	transakční aplikace

→ potřeba skloubit přednosti technologie WWW s databázovou technologií

- architektura klient/server - otázka komunikace WWW a DB serveru.

- Používané přístupy:

a) Použití CGI prostředků



- režie spuštění nových procesů, interpretace skriptů

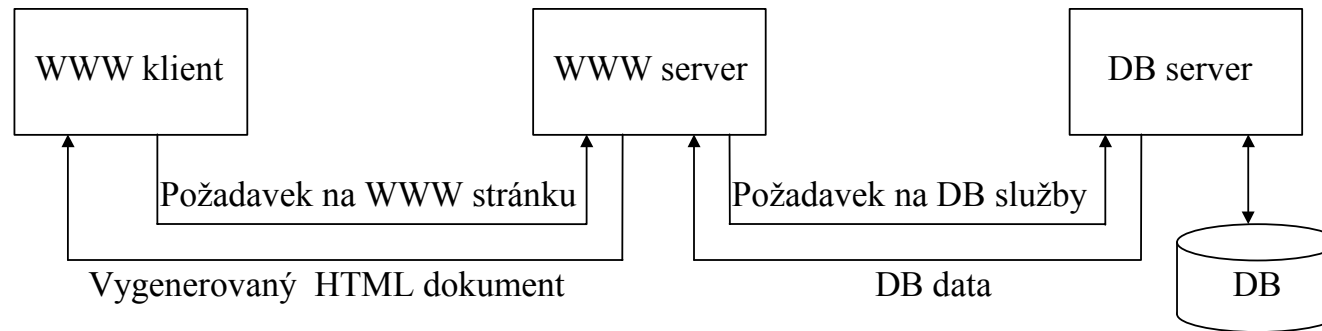
- opakované otevírání spojení s DB serverem

- př. skriptovací jazyky PHP, Perl, programy v C

b) Použití aplikací ve tvaru DLL

- snížení režie spuštění procesů oproti klasickým CGI aplikacím, př. ISAPI

c) Vyhodnocení dokumentu na WWW serveru formou přidáných direktiv



- př. Internet Information Server + IDC, ASP

d) Přístup k databázím z jazyka Java

C.2. PHP (Personal Home Page)

- skriptovací jazyk pro skripty na straně WWW serveru
- podporuje přístup k řadě SŘBD (Adabas D, dBase, Empress, FilePro, Informix, InterBase, mSQL, MySQL, Oracle, PostgreSQL, Solid, Sybase, Velocis, Unix dbm)

Přístup k MySQL (<http://www.mysql.com/>)

Připojení k databázovému serveru a nastavení aktuální aktivní databáze:

mysql_connect - otevře spojení s databázovým serverem MySQL, vrací identifikátor spojení

mysql_pconnect - otevře trvalé spojení s MySQL serverem (neukončuje se při ukončení skriptu ani funkcí *mysql_close()*), vrací identifikátor spojení.

mysql_select_db - nastaví aktuální aktivní databázi pro dané spojení s databázovým serverem

mysql_close - uzavře spojení s databázovým serverem MySQL

Vytvoření a zrušení databáze:

mysql_create_db - vytvoří databázi spravovanou serverem MySQL

mysql_drop_db - zruší databázi spravovanou serverem MySQL

Provádění příkazů SQL a práce s kurzorem (v terminologii PHP3 výsledek (result)):

mysql_query - pošle zadaný příkaz současné aktivní databázi serveru MySQL, pro INSERT, UPDATE a DELETE vrací příznak úspěšnosti, pro SELECT číslo kurzoru

mysql_db_query - vrátí číslo kurzoru pro zadaný dotaz pro danou databázi

mysql_affected_rows - vrátí počet řádků ovlivněných posledním příkazem INSERT, UPDATE nebo DELETE

mysql_fetch_array - vrátí řádek kurzoru jako asociativní pole (výběr podle jména sloupce)

Př)

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("uivt","select * from osoby");
while($row = mysql_fetch_array($result)) {
    echo $row["os_cislo"];
    echo $row["jmeno"];
}
mysql_free_result($result);
?>
```

mysql_fetch_row - vrátí řádek kurzoru jako pole s prvky zpřístupňovanými pořadovým číslem

Př)

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("uivt","select * from osoby");
while($row = mysql_fetch_row($result)) {
    echo $row[0];    // předp., že os_cislo je prvním vybraným sloupcem
    echo $row[1];    // předp., že jmeno je druhým vybraným sloupcem
}
mysql_free_result($result);
?>
```

mysql_fetch_object - vrátí řádek kurzoru jako objekt - přístup přes jména (obdoba *fetch_array*)

Př)

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("uivt","select * from osoby");
while($row = mysql_fetch_object($result)) {
    echo $row->os_cislo;
    echo $row->jmeno;
}
mysql_free_result($result);
?>
```

mysql_result - vrátí hodnotu daného sloupce daného řádku daného kurzoru, neměla by být používána s jinými funkcemi zpřístupňujícími řádky kurzoru

mysql_data_seek - posune ukazatel kurzoru na řádek s daným pořadovým číslem

mysql_free_result - uvolní paměťový prostor kurzoru

mysql_num_rows - vrátí počet řádků kurzoru

mysql_num_fields - vrací počet sloupců kurzoru

mysql_fetch_lengths - vrátí délky polí posledně vybraného řádku kurzoru funkcí *mysql_fetch_row*

mysql_insert_id - vrátí identifikátor generovaný posledním příkazem *INSERT* pro sloupec typu *AUTO_INCREMENTED*

Ošetření chyb:

mysql_errno - vrátí číslo chyby poslední operace MySQL

mysql_error - vrátí text chybového hlášení poslední operace MySQL

Přístup k metadatům:

mysql_list_dbs - vrátí kurzor obsahující seznam dostupných databází, k procházení slouží funkce *mysql_tablename*

mysql_list_tables - vrátí kurzor obsahující seznam tabulek dané databáze, k procházení slouží funkce *mysql_tablename*

mysql_tablename - vrátí jméno tabulky s daným pořadovým číslem prostřednictvím kurzoru vráceného funkcí *mysql_list_tables*

mysql_fetch_field - vrátí objekt, který nese informaci o daném sloupci daného kurzoru

mysql_field_seek - nastaví ukazatel kurzoru na daný sloupec

mysql_field_name - vrátí jméno daného sloupce kurzoru

mysql_field_table - vrátí jméno tabulky, které patří zadaný sloupec kurzoru

mysql_field_type - vrátí jméno tabulky, které patří zadaný sloupec výsledku

mysql_field_flags - vrátí příznaky ("not_null", "primary_key", ...), spojené se zadaným sloupcem výsledku

mysql_field_len - vrátí délku specifikovaného sloupce daného kurzoru

mysql_list_fields - zpřístupní metadata pro danou tabulku, vrací číslo kurzoru, které lze použít ve funkcích pro práci s metadaty (*mysql_field_flags()*, *mysql_field_len()*, *mysql_field_name()* a *mysql_field_type()*)

Př)

```
<?php
mysql_connect($host,$user,$password);
mysql_select_db("uivt");
$result = mysql_query("SELECT * FROM osoby");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$i = 0;
$table = mysql_field_table($result, $i);
echo "Tabulka '". $table. "' má ". $fields. " Sloupců a ". $rows. " řádků <BR>";
echo "Tabulka má následující sloupce: <BR>";
while ($i < $fields) {
    $type = mysql_field_type ($result, $i);
    $name = mysql_field_name ($result, $i);
    $len  = mysql_field_len ($result, $i);
    $flags = mysql_field_flags ($result, $i);
    echo $name. " ". $type. " ". $len. " ". $flags. "<BR>";
    $i++;
}
mysql_close();
?>
```

C.3. Internet Information Server

- podpora pro přístup k databázi:

CGI aplikace, Internet Database Connector, Active Server Pages

C.3.1. Active Server Pages (ASP)

- skripty vykonávány na straně serveru, provedení před odesláním stránky klientovi

- textové informace, HTML značky, příkazy skriptu (VBScript, JavaScript), oddělovače příkazů skriptu dvojice `<% %>`, resp. značka `<SCRIPT RUNAT=SERVER ... > </SCRIPT>`

- klientské a serverovské skripty lze kombinovat (modifikace klientského skriptu)

- zabudované objekty :

- *Request* - získání informací od uživatele.
- *Response* - zaslání informací uživateli.
- **Server** - práce s ActiveX komponentami (pro DB Database Access - **ADO**).
- *Session* - uchování informací o uživatelské sezení.
- *Application* - uchování informací o uživatelské dané aplikaci.

Př) **ADODB** pro Microsoft OLE DB přes ODBC

- třídy pro přístup k datům v databázi:

- *Connection* - reprezentace spojení se zdrojem dat.
- *Recordset* - reprezentace tabulky vrácené po vykonání příkazu (obdoba kurzoru).

Př)

...

```
<%Conn = Server.CreateObject("ADODB.Connection")
Conn.Open("ADOZamestnanci")
```

```

sql="SELECT Jmeno, Plat FROM Zamestnanci WHERE plat > 10000"
RS = Conn.Execute(sql)
counter = 0
while (!RS.EOF) {
    hodnota[counter].jmeno = RS("Jmeno")
    hodnota[counter].plat = RS(1)
    RS2.MoveNext()
    counter = counter + 1
} %>
...

```

- **Command** - definice příkazu pro zdroj dat.
 - **Field** - reprezentuje sloupec dat objektu třídy Recordset.
 - **Parameter** - reprezentuje parametr objektu třídy Command.
- **Error** - informace o chybě vzniklé při vykonání příkazu.

Př)

```

<%@ LANGUAGE = "JavaScript" %>
<HTML>
<HEAD>
<TITLE> Správa hotelů </TITLE>
<%
Conn = Server.CreateObject("ADODB.Connection")
Conn.Open("dsn=Dracon;uid=web")
SQL1 = "SELECT HCislo, RTRIM(Hotel) as Hotel, RTRIM(Stranka) as Stranka,
CenaDo, CenaPo, "

```



WWW server

```
SQL1 += "CenaXx FROM Hotely WHERE HCislo <> 0 ORDER BY HCislo"
```

```
SQL2 = "SELECT Aktualni_Cislo, Krok FROM Ciselnik_Hotelu"
```

```
RS1 = Conn.Execute(SQL1)
```

```
RS2 = Conn.Execute(SQL2)
```

```
%>
```

```
<SCRIPT LANGUAGE="JavaScript">
```

```
function THotel(HCislo, Hotel, Stranka, CenaDo, CenaPo, CenaXX){ ... }
```

```
../* Tato funkce se volá při načtení stránky */
```

```
function Load(){
```

```
var f
```

```
/* Načte informace z DB do pole objektů Hotel */
```

```
  <%counter = 0
```

```
while(!RS1.EOF){%>
```

```
  Hotely[<%=counter%>] = new THotel (<%=RS1("HCislo")%>,
    '<%=RS1("Hotel")%>', '<%=RS1("Stranka")%>',
    <%=RS1("CenaDo")%>, <%=RS1("CenaPo")%>,
    <%=RS1("CenaXX")%>)
```

```
    document.formik.hotely.options[<%=counter%>] = new Option
      (Hotely[<%=counter%>].Hotel,<%=counter%>,false,false)
```

```
  <% counter = counter + 1
```

```
  RS1.MoveNext()
```

```
%>
```

```
PocetHotelu = <%=counter%>
```

WWW klient

modifikace klientského skriptu

```

...

} /* function Load */

/* Tato funkce volá skript pro vykonání SQL příkazů */
function Odeslat(){ ... }
  /* Funkce vkládá nové hotely. */
function NovyHotel(f){ ... }
function UlozZmeny(){ ... }
function ZmenaHotelu(f){ ... }
function ZrusHotel(){ ... }
/* ***** */
</SCRIPT>
</HEAD>
<BODY onLoad="Load()">

<CENTER><H1>Správa hotelů</H1></CENTER>

<FORM NAME="formik">
<TABLE WIDTH="100%" >
  <TR> <TH WIDTH="30%"><CENTER>Hotely</CENTER></TH>
    <TH><CENTER>Údaje</CENTER></TH>
  </TR>

```

```

<TR><TD><SELECT NAME = "hotely" SIZE = "8" onFocus=""
      onChange="ZmenaHotelu(this.form)" WIDTH = 150>
      <OPTION VALUE=0> xxxxxxxxxxxxxxxx
      <OPTION VALUE=0> xxxxxxxxxxxxxxxx
      <OPTION VALUE=0> xxxxxxxxxxxxxxxx
      <OPTION VALUE=0> xxxxxxxxxxxxxxxx
    </SELECT>
  </TD>
  <TD><TABLE WIDTH="100%" >
    <TR> <TD WIDTH="40%">Název hotelu </TD>
      <TD><INPUT TYPE="TEXT" NAME="name" SIZE="15" MAXLENGTH="14"
        onChange="OKModifikace=true"></TD>
    </TR>
    <TR> <TD >Adresa stránky</TD>
      <TD><INPUT TYPE="TEXT" NAME="addr" SIZE="30" MAXLENGTH="31"
        onChange="OKModifikace=true"></TD>
    </TR>
    <TR><TD>Cena do termínu</TD>
      <TD><INPUT TYPE="TEXT" NAME="cenado" SIZE="5" MAXLENGTH="4"
        onChange="OKModifikace=true"></TD>
    </TR>
    <TR><TD>Cena po termínu</TD>
      <TD><INPUT TYPE="TEXT" NAME="cenapo" SIZE="5" MAXLENGTH="4"
        onChange="OKModifikace=true"></TD>
  </TD>
</TR>

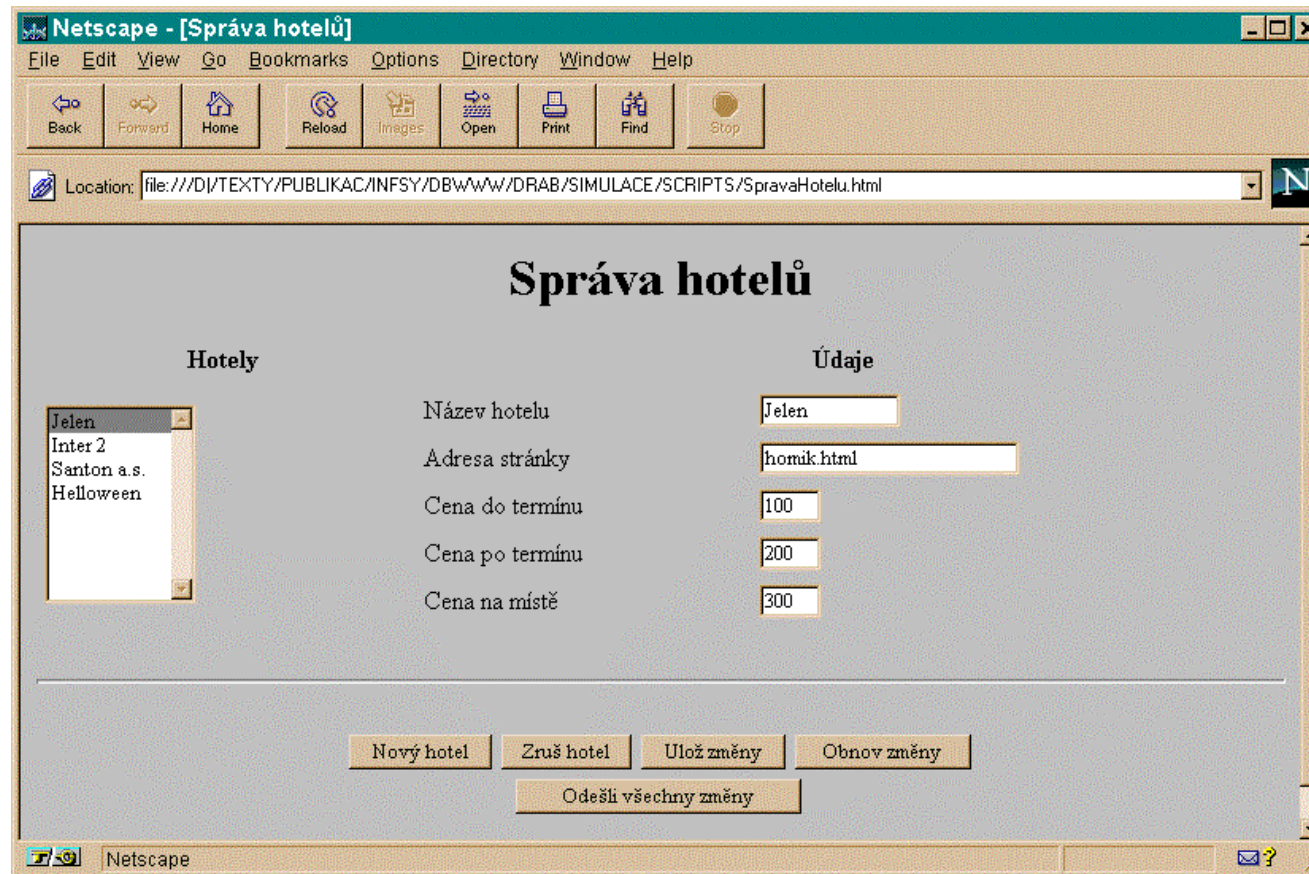
```

```

</TR>
<TR><TD>Cena na místě</TD>
    <TD><INPUT TYPE="TEXT" NAME="cenaxx" SIZE="5" MAXLENGTH="4"
        onChange="OKModifikace=true"></TD>
</TR>
</TABLE>
</TD>
</TR>
</TABLE>
<BR>
<HR>
<BR>
<CENTER>
<TABLE WIDTH="50%"
    <TR><TD><INPUT TYPE="BUTTON" VALUE="Nový hotel"
        onClick="NovyHotel(this.form)"></TD>
        <TD><INPUT TYPE="BUTTON" VALUE="Zruš hotel"
            onClick="ZrusHotel()"></TD>
        <TD><INPUT TYPE="BUTTON" VALUE="Ulož změny"
            onClick="UlozZmeny()"></TD>
        <TD><INPUT TYPE="BUTTON" VALUE="Obnov změny"
            onClick="ZmenaHotelu(this.form)"></TD>
    </TR>
    <TR><TD COLSPAN="4" WIDTH="100%" ALIGN="CENTER"><INPUT
        TYPE="BUTTON" VALUE="Odešli všechny změny" onClick="Posli()"></TD>

```

```
</TR>
</TABLE>
</CENTER>
</FORM>
<FORM NAME="stat">
<INPUT TYPE="HIDDEN" NAME="Flag1">
<INPUT TYPE="HIDDEN" NAME="Flag2">
<INPUT TYPE="HIDDEN" NAME="Turn">
<INPUT TYPE="HIDDEN" NAME="OK">
</FORM>
</BODY>
```

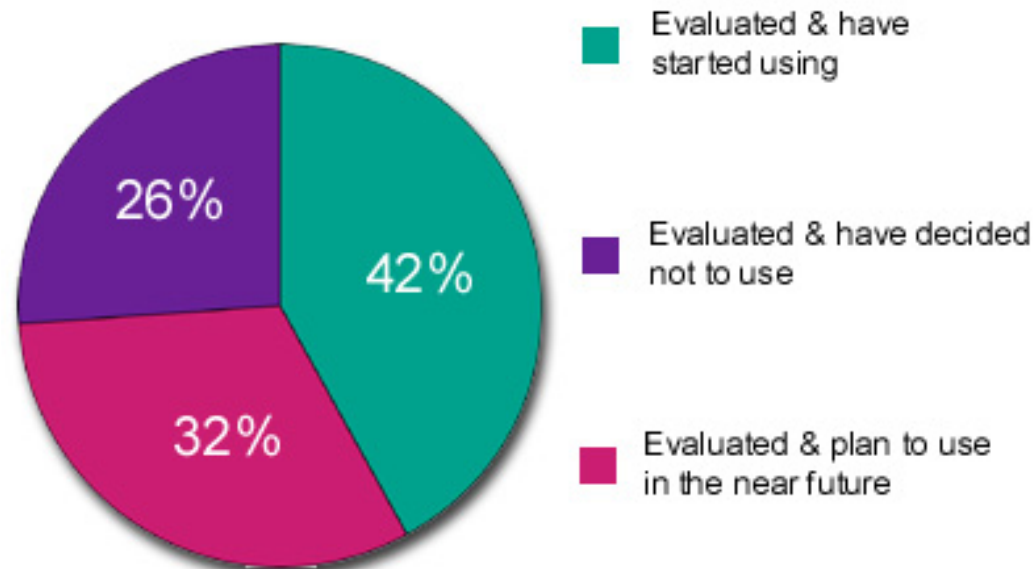


</HTML>

C.4. Přístup k databázím z jazyka Java

What is your company's relationship with Java?

74% of those evaluated WILL use Java



BYTE Java Adoption Study, December 1997

Možnosti přístupu k databázi z prostředí jazyka Java:

- JDBC
- vložený SQL (SQLJ)

C.4.1. Rozhraní JDBC

- Java API pro styk s relačními databázemi, rozhraní „nízké“ úrovně
- typické kroky při přístupu k databázi:

- *Vytvoření spojení:*

```
Connection con = DriverManager.getConnection  
    ("jdbc:odbc:db", "login", "password");
```

- *Vytvoření příkazu:*

```
Statement stmt = con.createStatement();
```

- *Vykonání příkazů a vytvoření objektu pro zpracování výsledků :*

```
ResultSet rs = stmt.executeQuery("SELECT a, b, c  
    FROM table1");
```

- *Zpracování výsledků:*

```
WHILE(rs.next()){  
    int x = rs.getInt("a");  
    String s = rs.getString(2);  
    Float f = rs.getFloat("c");  
}
```

- kolekce Java tříd a rozhraní:

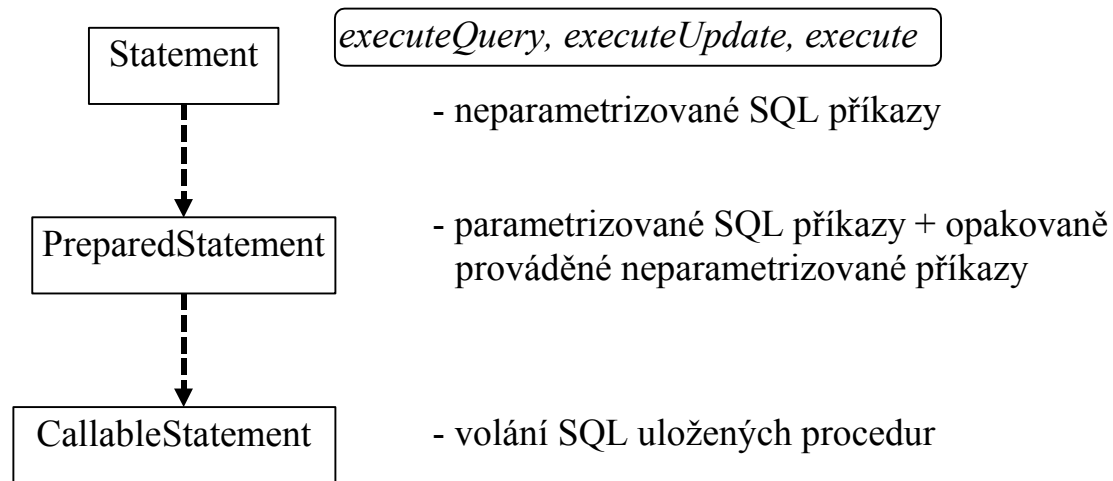
- *java.sql.DriverManager* - práce s ovladači, ustavení spojení s databází
 - metoda *getConnection* (URL, user, password)
URL: jdbc:<subprotokol>:<jméno>
- *java.sql.Connection* - reprezentace spojení s databází

Př)

```
Connection connection;
String url = "jdbc:odbc:Web SQL";
/* Pokus o pripojeni a nastaveni AUTOCOMMIT */
try { /* zavedeni driveru */
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}
catch (java.lang.ClassNotFoundException ex) { ... }
try { /* ustaveni spojeni */
    connection = DriverManager.getConnection(url,
        "sa", "brno");
}
catch (java.sql.SQLException ex){ ... }
try { /* nastaveni rizeného COMMIT */
    connection.setAutoCommit(false);
}
catch (java.sql.SQLException ex){ ... }
```

- *java.sql.Statement, PreparedStatement, CallableStatement* - reprezentuje kontejner pro vykonání (SQL) příkazů

- tři třídy pro posílání příkazů:



- **java.sql.ResultSet** - reprezentuje výsledek dotazu (obdoba kurzoru z SQL).
 - metody next() a getXXX(), získání metadat, testování NULL, ...

Př)

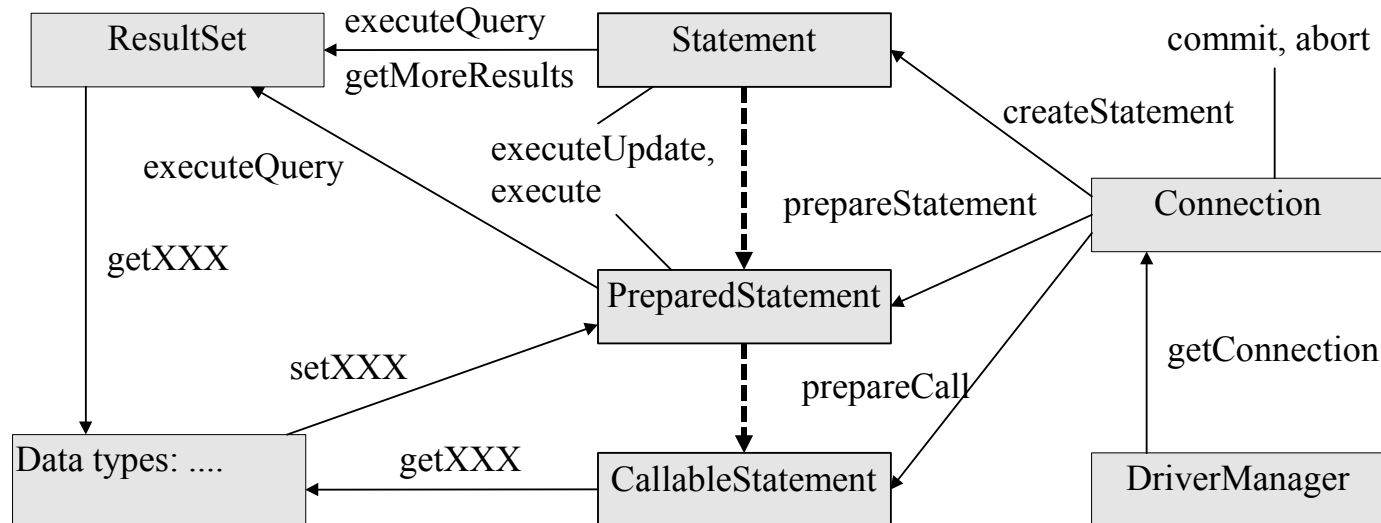
```
Connection con = ....
PreparedStatement pstmt = con.prepareStatement(
    "UPDATE table2 SET m=? WHERE x=?");
pstmt.setLong(1, 123456);
pstmt.setLong(2, 0);
pstmt.executeUpdate();
CallableStatement cstmt = con.prepareCall(
    "{call getNewData(?, ?)}" );
cstmt.registerOutParameter(1, java.sql.Types.TINYINT);
cstmt.registerOutParameter(2, java.sql.Types.DECIMAL, 2);
```

```

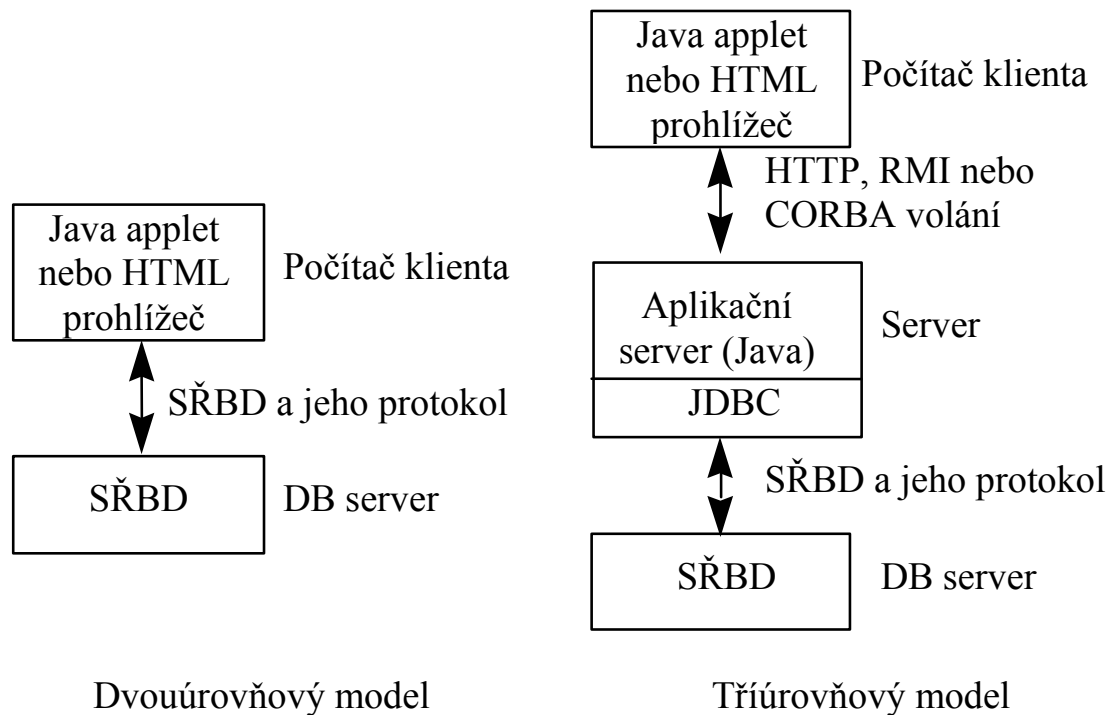
cstmt.executeUpdate();
byte x = cstmt.getBytes(1);
Bignum y = cstmt.getBignum(2,2);

```

▪ **Komunikace rozhraní a tříd**

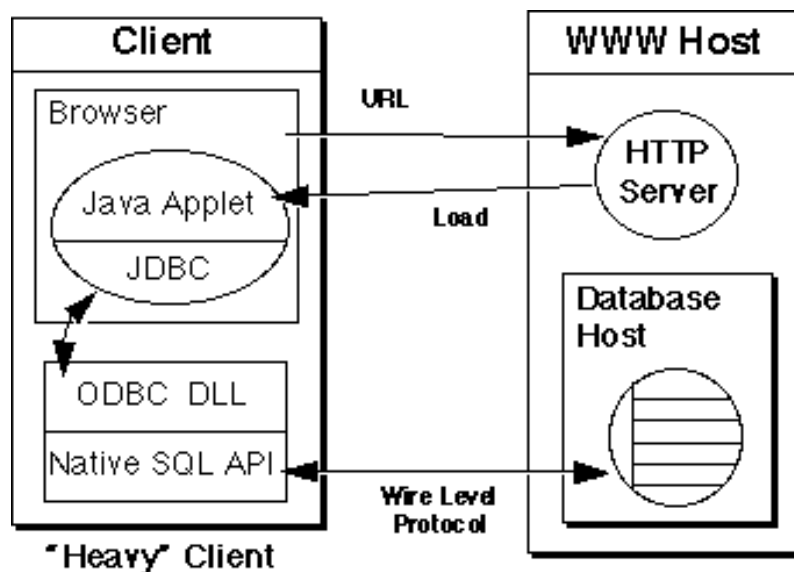


- podpora dvou- i tříúrovňového modelu přístupu k databázi.



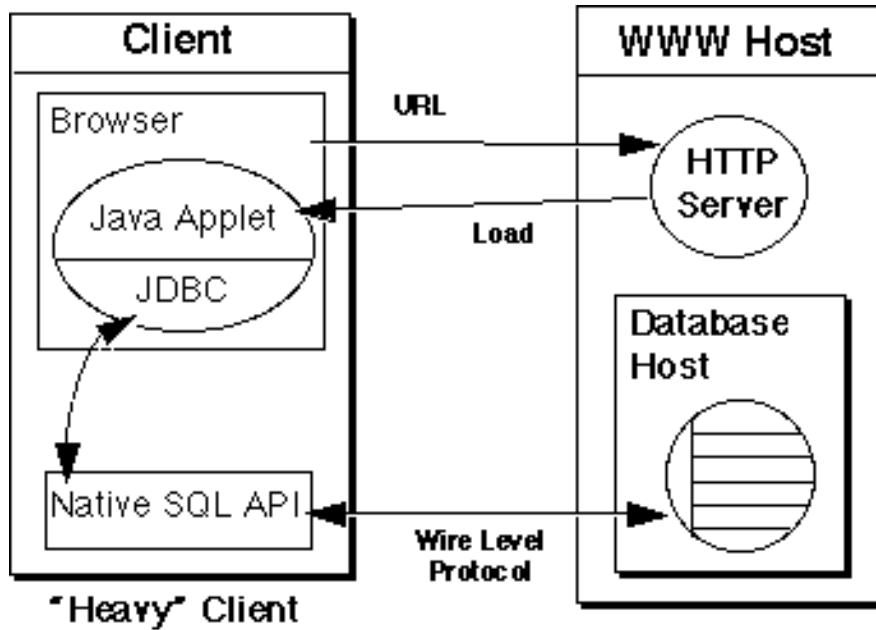
- kategorie JDBC ovladačů:

1. *JDBC-ODBC most + ODBC ovladač*

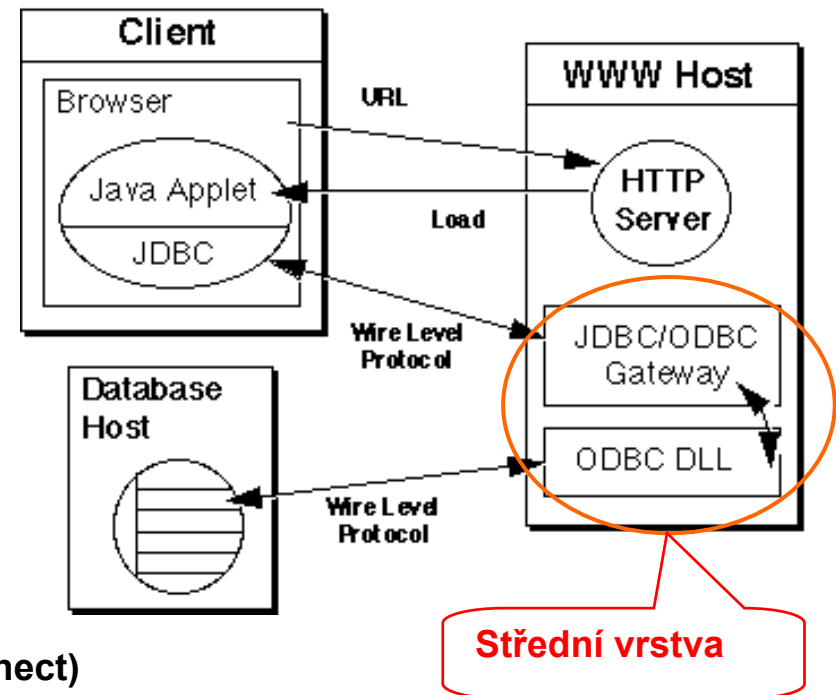


- problémy:
 - vyšší režie
 - náročnější údržba - nesplňuje požadavek „*Just-in-time delivery*“

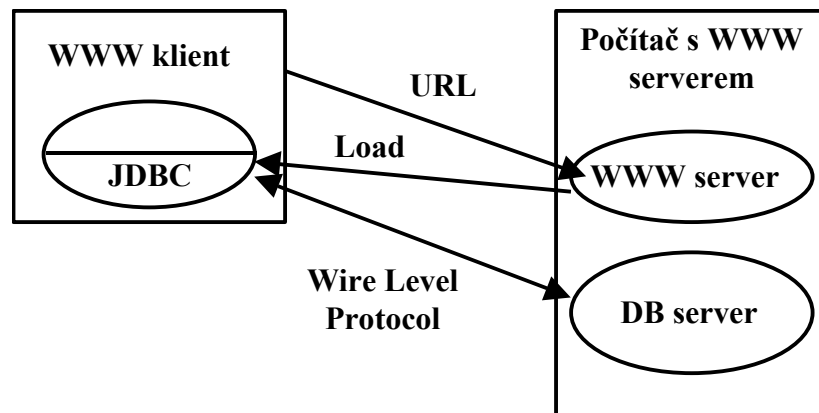
2. „Native - API partly-Java“ ovladač
 ovladač
 (kombinace Javy a C (C++))



3. „JDBC-Net all-Java“



3. „Native-protokol all-Java“ ovladač (např. Sybase jConnect)



C.4.2. SQLJ

Charakteristika SQLJ

Standardní způsob vkládání SQL:

- ANSI dokument X3.135.10-1998,
- ISO - ISO/IEC 9075, část 10 (OLB).

Motivace:

- kompaktní aplikační rozhraní vysoké úrovně,
- syntaktickou a sémantickou kontrolu příkazů SQL před prováděním programu,
- nezávislost syntaxe a sémantiky příkazů SQL na místě provádění (klient, databázový server, střední vrstva),
- možnost kombinace rozhraní SQLJ a JDBC sdílením identifikátorů spojení (connection handle),
- binární přenositelnost

SQL: SQL92

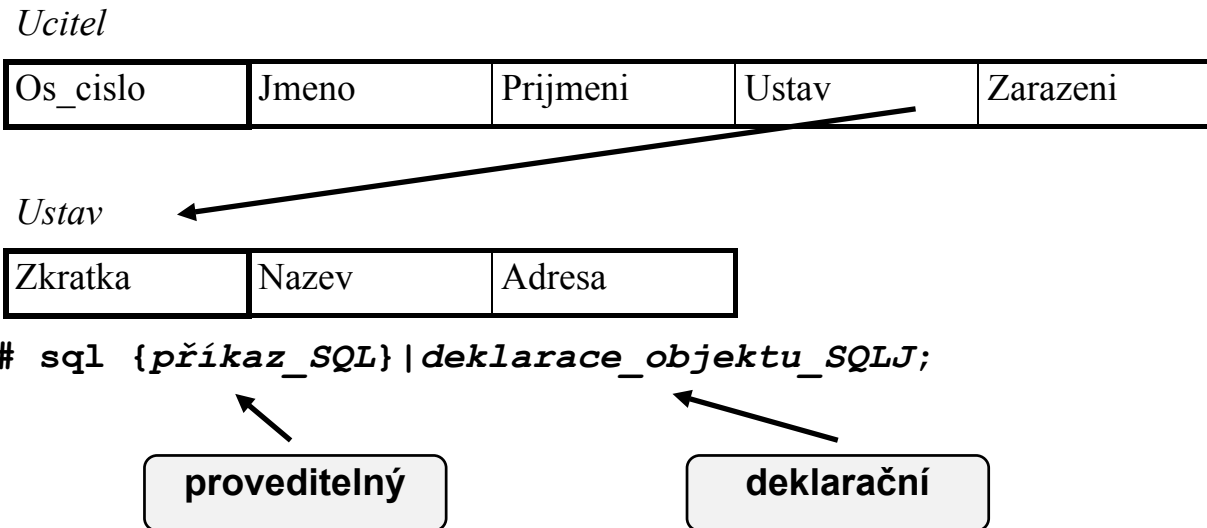
Zvláštnosti prostředí jazyka Java oproti hostitelským jazykům pro SQL:

- objektová orientace,
- automatická správa paměti,
- existence odlišitelné prázdné hodnoty (null) pro složené datové typy,
- binární přenositelnost,
- podpora pro spolupráci komponent různých výrobců



Rysy SQLJ

Př)



- SELECT nebo výraz,
- INSERT, UPDATE, DELETE,
- FETCH, SELECT...INTO,
- COMMIT, ROLLBACK,
- CREATE, DROP, ALTER,
- CALL, VALUES
- SET.

- spojení,
- iterátor.

- *hostitelské proměnné a výrazy*

```
#sql {SELECT ADRESA INTO :adresaUstavu FROM USTAV  
      WHERE ZKRATKA = :zkratkaUstavu};
```

```
#sql {SELECT PRIJMENI INTO :prijmeni FROM UCITEL  
      WHERE OS_CISLO = :(docenti[i++])};
```

vyhodnocení před JVM (zleva)
→ možnost vedlejšího efektu

- *iterátor* - Java objekt s funkcí kurzoru

- mechanismy pro mapování sloupců dotazu na sloupce iterátoru:

- ◆ poziční (poziční iterátor),

```
#sql iterator ZamestnanciUstavu(int, String);
```

- ◆ podle jména (pojmenovaný iterátor).

```
#sql iterator ZamestnanciUstavu(int osCislo, String prijmeni);
```

```
#sql iterator ZamestnanciUstavu(int, String);
ZamestnanciUstavu iterP;      // deklarace objektu iterP
// inicializace objektu iterP
#sql iterP = {SELECT OS_CISLO, PRIJMENI
              FROM UCITEL
              WHERE ZKRATKA = :zkratkaUstavu};

int c; String p;
#sql {FETCH :iterP INTO :c, :p};
while (!iterP.endFetch()){
    System.out.println(p + " (os.číslo: " + c + ")");
    #sql {FETCH :iterP INTO :c, :p};
}
```

```

#sql iterator ZamestnanciUstavu(int osCislo, String prijmeni);
ZamestnanciUstavu iterN;      // deklarace objektu iterN
// inicializace objektu iterN
#sql iterN = {SELECT OS_CISLO AS OSCISLO, PRIJMENI
              FROM UCITEL
              WHERE ZKRATKA = :zkratkaUstavu};

int c; String p;
while (iterN.next()){
    c = iterN.osCislo();
    p = iterN.prijmeni();
    System.out.println(p + " (os.číslo: " + c + ")");
}

```

- **uložené procedury a funkce**

```

CREATE PROCEDURE ZMEN_NAZEV(IN ZKR_USTAVU CHAR(3), NOVY_NAZEV VARCHAR(150)); ...
CREATE FUNCTION POCTY(ZKR_USTAVU IN CHAR(3), PROF OUT NUMBER(2),
                     DOC OUT NUMBER(2)) RETURNING NUMBER(2); ...

```

```

#sql {CALL ZMEN_NAZEV(:zkratkaUstavu, :novyNazev)};
#sql celkem = {VALUES(POCTY(:IN zkratkaUstavu, :OUT prof, :OUT doc))};

```

- **objekt kontextu spojení**

```

#sql context DBkontext;
DBkontext kontext; // deklarace objektu kontextu spojení
kontext = new DBkontext(url, true); // inic. objektu, včetně otevření spojení

```

- **exemplární schéma (exemplar schema)**

- skutečné/“typické“ schéma pro účely kontroly

- přiřazení třídám kontextu při spuštění překladače SQLJ

- **další rysy**

- několik otevřených spojení, kombinace s JDBC, ...

```
#sql [kontext] {...};
```

Porovnání SQLJ s JDBC

JDBC - obecnější

SQLJ - vyšší úroveň statické vazby se schématem DB

Důsledky:

- SQLJ program je kratší,
- možnost spojení s databází v době překladu pro typovou kontrolu,
- jednodušší práce s hostitelskými proměnnými, možnost použití výrazů,
- typování výsledků dotazů a návratových hodnot, JDBC předává hodnoty do/z SQL bez typové kontroly v době překladu,
- volání uložených procedur a funkcí je v SQLJ také jednodušší než v JDBC.

Př)

```
Connection con = ....
```

```
PreparedStatement pstmt = con.prepareStatement("UPDATE table2 SET m=?  
                                         WHERE x=?");
```

```
pstmt.setLong(1, 123456);
```

```
pstmt.setNULL(2);
```

```
pstmt.executeUpdate();
```

```
CallableStatement cstmt = con.prepareCall("{call getNewData(?, ?)}");
```

```
cstmt.registerOutParameter(1, java.sql.Types.TINYINT);
```

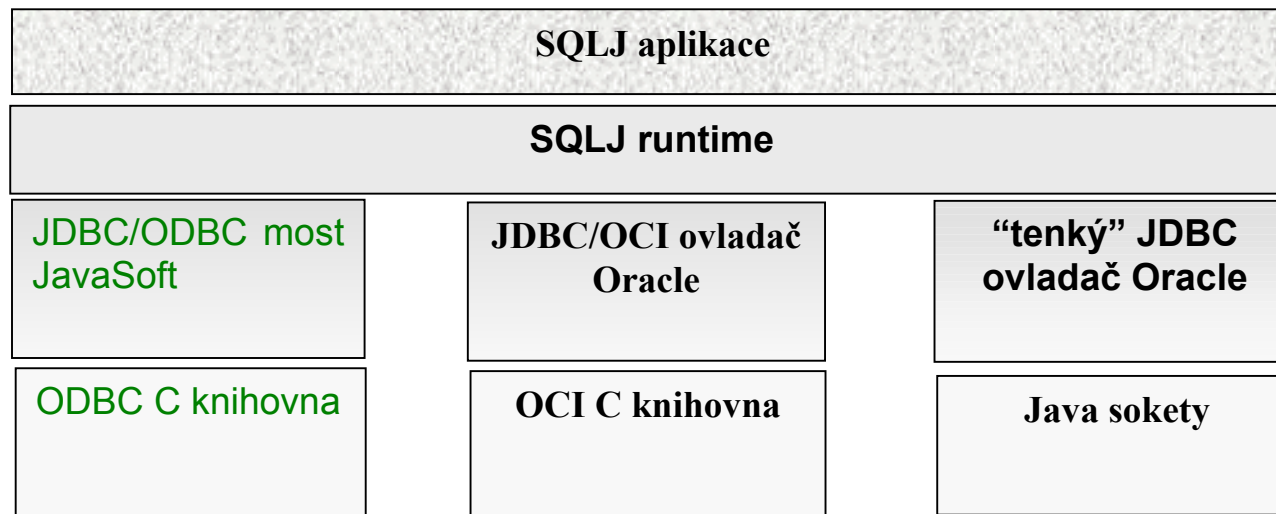
```

cstmt.RegisterOutParameter (2 , java.sql.Types.DECIMAL , 2) ;
cstmt.executeUpdate () ;
byte x = cstmt.getBytes (1) ;
Bignum y = cstmt.getBignum (2,2) ;

```

Podpora jazyka Java ze strany Oracle

- **strategická podpora Javy ze strany Oracle**
 - JDeveloper + Oracle Application Server + Oracle8i (vč. JServeru- podpora JVM, Corba, JDBC, Enterprise Java Beans, SQLJ),
- **referenční implementace překladače SQLJ**
Oracle, IBM, Sybase, Informix, Compaq/Tandem, JavaSoft,



- *varianty rozložení*

