Seminář Java

X

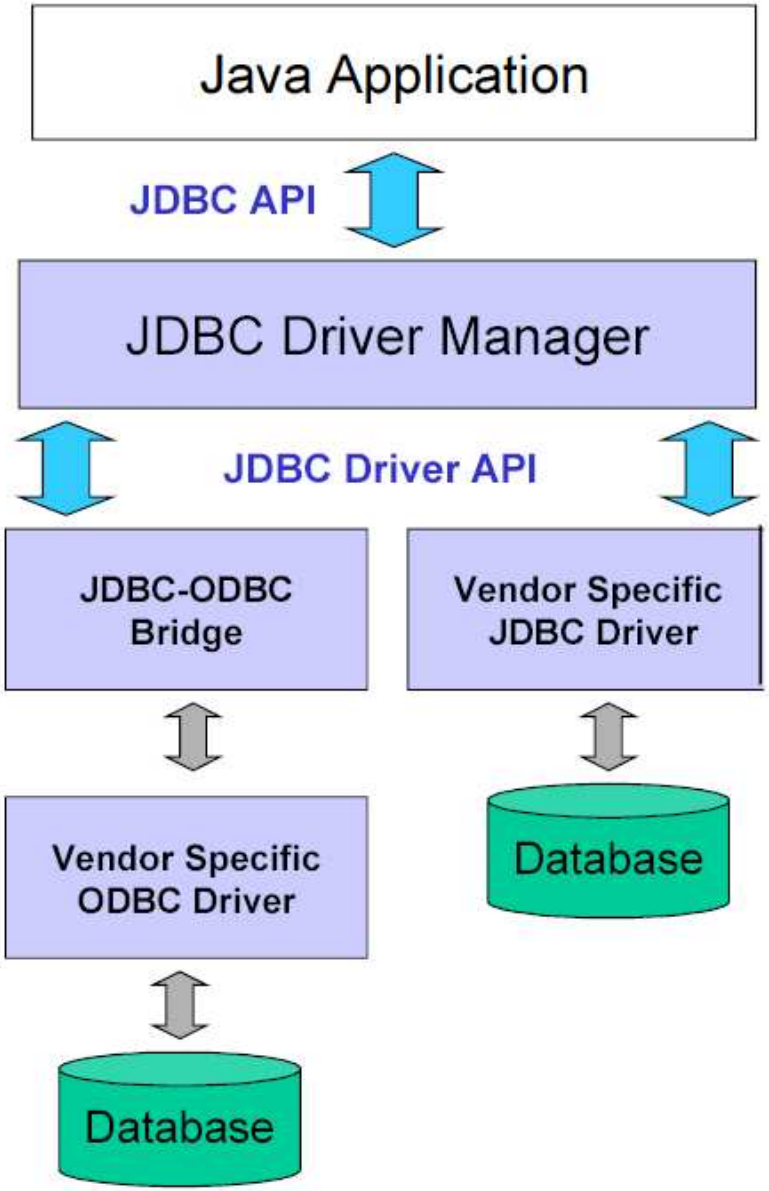
JDBC

Úvod

JDBC API poskytuje základní rozhraní pro unifikovaný přístup k databázím

- Programátor je odstíněn od specifického API databáze
- Jednotné rozhraní JDBC pro všechny DB
- JDBC není určeno pouze pro přístup k relačním databázím
  $\longrightarrow$ k libovolnému formátu dat ve „sloupcové podobě"
- Inspirováno ODBC standardem

Architektura JDBC



JDBC ovladače

- **Typ 1** - lokální ODBC, instalace, součást JDK
- **Typ 2** - JDBC-nativní ovladač, instalace
- **Typ 3** - Čistá Java, síťová komunikace, centrální server
- **Typ 4** - Čistá Java, žádná instalace, optimalizace výrobcem DB

Použití JDBC

1. Nahrát ovladač
2. Definovat URL pro připojení
3. Navázat spojení
4. Vytvoření objektu třídy *Statement*
5. Dotaz
6. Zpracování výsledku
7. Zavření spojení

Použití JDBC - Nahrání ovladače

- Specifická třída (obvykle dodána výrobcem DB)
- Ovladač je nutné nahrát
- *DriverManager*
- classpath

```
try {
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Class.forName("org.qjt.mm.mysql.Driver");
} catch(ClassNotFoundException e) {
  System.err.println("Nelze nahrat DB driver : "
                     +e.getMessage());
}
```

Použití JDBC

1. Nahrát ovladač
2. Definovat URL pro připojení
3. Navázat spojení
4. Vytvoření objektu třídy *Statement*
5. Dotaz
6. Zpracování výsledku
7. Zavření spojení

Použití JDBC - Definice URL

- *String*
- Podle URL se vybere ovladač

```
String host = "muj.server.cz";
String dbName = "ijaTest";
int port = 1234;

String oracleURL = "jdbc:oracle:thin:@"+
                   host+":"+port+":"+dbName;

String mySqlURL = "jdbc:mysql://"+host+":"+
                  port+"/"+dbName;
```

Použití JDBC

1. Nahrát ovladač
2. Definovat URL pro připojení
3. Navázat spojení
4. Vytvoření objektu třídy *Statement*
5. Dotaz
6. Zpracování výsledku
7. Zavření spojení

Použití JDBC - spojení

- *DriverManager.getConnection(URL, uzivatel, heslo);*
- *Connection*
- Náročná operace

```
String user = "pepa";
String password = "heslo";

Connection connection =
    DriverManager.getConnection(URL, user, password);
```

Získání informací o DB

- *DatabaseMetaData md = connection.getMetaData();*
- *getDatabaseProductName(), getDatabaseProductVersion()*

Použití JDBC

1. Nahrát ovladač
2. Definovat URL pro připojení
3. Navázat spojení
4. Vytvoření objektu třídy *Statement*
5. Dotaz
6. Zpracování výsledku
7. Zavření spojení

Použití JDBC - Statement, dotaz

- Posílání SQL dotazů DB (3 třídy)
  - Statement – jednoduché SQL
  - PreparedStatement – předkompilované SQL, parametry
  - CallabeStatement – volání uložených procedur na SQL serveru
- Mnoho metod (API)
- *executeQuery()*
  - SQL dotazy, vrací data v *ResultSet*
  - Prázdná ano, null nikdy
- *executeUpdate()*
  - Pro modifikaci tabulky
  - Podpora DDL (CREATE, ALTER, . . .)

Použití JDBC - Statement, dotaz II

Statement statement = connection.createStatement();

```
Statement statement = connection.createStatement();

String sql = "SELECT * FROM mojetabulka";
ResultSet resultSet = statement.executeQuery(sql);

String sqlUpdate = "DELETE FROM mojetabulka";
statement.executeUpdate(sqlUpdate);
```

Použití JDBC - Statement, užitečné metody

- **execute** – uložené procedury
- **getMaxRows/setMaxRows** – maximální počet řádek ve výsledku (0 neomezeno)
- getQueryTimeout/setQueryTimeout – maximální čas po který bude driver čekat na výsledek

Použití JDBC

1. Nahrát ovladač
2. Definovat URL pro připojení
3. Navázat spojení
4. Vytvoření objektu třídy *Statement*
5. Dotaz
6. Zpracování výsledku
7. Zavření spojení

Použití JDBC - Výsledky

- *ResultSet* – Obsahuje výsledek SQL dotazu
  - „tabulka s řádky a sloupci"
- První sloupec ma index 1!
- metoda *next()* – pokus o přesun na další řádek
- *getString(int index), getString(String nazev)*

```
while (resultSet.next()) {
  String data1 = resutlSet.getString(1);
  String data2 = resutlSet.getString("jmeno");
}
```

- metody *getXXX* pro různé datové typy (int, double, float, . . .)

Použití JDBC - Výsledky II

- metoda *close()* – automatické uzavření pokud rodičovský Statement opět udělá dotaz
- metoda *getMetaDataObject* – vrátí *ResultSetMetaData*, informace o výsledku
  - Počet sloupců výsledků
  - Názvy sloupců
  - Typy sloupců

Použití JDBC

1. Nahrát ovladač
2. Definovat URL pro připojení
3. Navázat spojení
4. Vytvoření objektu třídy *Statement*
5. Dotaz
6. Zpracování výsledku
7. Zavření spojení

Použití JDBC - Zavření spojení

- *close();*

```
connection.close();
```

- Připojení je náročná operace → pooling

JDBC - Pooling

- Pomocí „poolingu" se citelně zvýší výkonnost aplikace
- Nezavírá se spojení s DB, uložení do „bazénku"
- Několik otevřených připojení
- Využití – web aplikace
- „sklad připojení"

PreparedStatement

- Předkompilované dotazy
- Opakované použití SQL dotazu
- Přehledné použití parametrů – složité SQL dotazy s parametry
- parametry dotazu se nastavují – **setXXX**

- metody **execute()**, **executeQuery()**, **executeUpdate()**
- **clearParameters** – vymaže nastavené parametry Pozn. :

Kontrola dat z www

PreparedStatement II

```
PreparedStatement statement =
   connection.prepareStatement(
      "UPDATE lidi SET plat=? WHERE id=?");

for (... pres vsechny lidi ...) {
   Clovek clovek = .....
   statement.setInt(1, clovek.getPlat());
   statement.setInt(2, clovek.getId());
   statement.executeUpdate();
}
```

Transakce

- Co to jsou transakce?

- Po každé operaci pomocí Statement automatické potvrzení změny dat
- Vypnutí automatického potvrzení **connection.setAutoCommit(false);**
- Nutně potvrzovat ručné – **connection.commit();**
- provedení „rollback" – **connection.rollback();**

Transakce - příklad

```
connection.setAutoCommit(false);
try {
  statement.executeUpdate(...);
  statement.executeUpdate(...);
} catch(SQLException e) {
  try {
    connection.rollback();
  } catch(SQLException er) {
    System.err.println("Nemohu provest rollback");
  }
} finally {
  try {
    connection.commit();
    connection.close();
  } catch(SQLException er) {
    System.err.println("Nemohu provest commit/close");
  }
}
```

Rekapitulace

- Připojení reprezentuje *Connection*
- Zvýšení výkonnosti – „pooling" připojení
- Zasílání dotazů, volání uložených procedur – *Statement*
- Zvýšení výkonnosti – předkompilované dotazy *PreparedStatement*
- Implicitně autoCommit = true
- SQLException jsou spojené do hromady