Chương 6: LẬP TRÌNH CSDL VỚI JDBC

Khoa CNTT

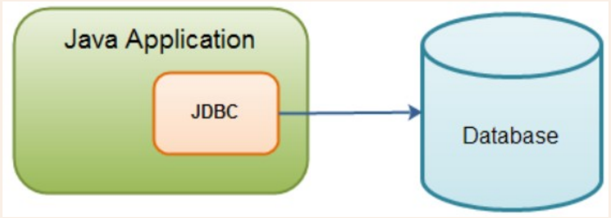
ĐH GTVT TP.HCM

Nội dung

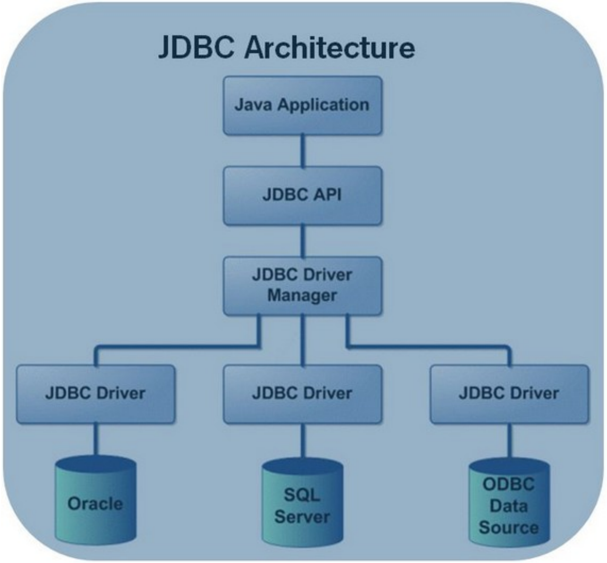
JDBC là gì?

Định nghĩa

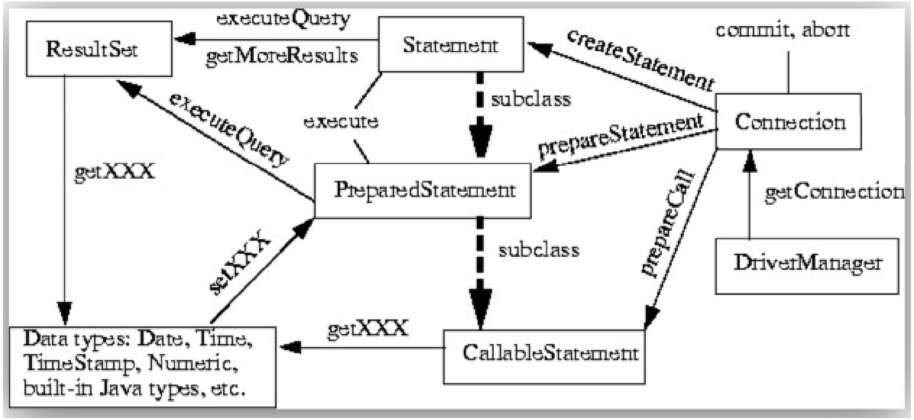
JDBC (Java Database Connectivity), which is a standard Java API for database-independent connectivity between the Java program and a wide range of databases.



Kiến trúc JDBC



JDBC Application Program Interface (API)



JDBC Application Program Interface (API)

Classes & Interfaces

* **DriverManager**: This class loads JDBC drivers in memory. It is a “factory” class and can also be used to create java.sql.Connection objects to data sources (such as Oracle, MySQL, etc.).
* **Connection**: This interface represents a connection with a data source. The Connection object is used for creating Statement, PreparedStatement, and CallableStatement objects.
* **DatabaseMetaData**: This interface provides detailed information about the database as a whole. The Connection object is used for creating Database MetaData objects.
* **Statement**: This interface represents a static SQL statement. It can be used to retrieve ResultSet objects.

JDBC Application Program Interface (API)

Classes & Interfaces

* **PreparedStatement**: This interface extends Statement and represents a precompiled SQL statement. It can be used to retrieve ResultSet objects.
* **CallableStatement**: This interface represents a database stored procedure. It can execute stored procedures in a database server.
* **ResultSet**: This interface represents a database result set generated by using SQL's SELECT statement. Statement, PreparedStatement, CallableStatement, and other JDBC objects can create ResultSet objects.
* **ResultSetMetaData**: This interface provides information about the types and properties of the columns in a ResultSet object.
* **SQLException**: This class is an exception class that provides information on a database access error or other errors.

Kết nối đến CSDL

Connection

```
static private Connection con = null;
static String driver = "com.mysql.jdbc.Driver";
static String host = "jdbc:mysql://localhost:3306/banhang
?useUnicode=yes&characterEncoding=UTF-8";
static String uName = "root";
static String uPass = "root";
static private Connection createConnection() throws SQLException,
    ClassNotFoundException {
    //Class.forName(driver);
    DriverManager.registerDriver(new com.mysql.jdbc.Driver());
    return DriverManager.getConnection(host, uName, uPass);
}
```

Các thao tác cơ bản trên CSDL

Đọc dữ liệu

```
static public ResultSet getResultSet(String query, Object... params) throws
    SQLException, ClassNotFoundException {
        con = createConnection();
        PreparedStatement pst = con.prepareStatement(query);
        int j=1;
        for (int i = 0; i < params.length-1; i+=2) {
            pst.setObject(j, params[i], (int) params[i+1]);
            j++;
        }
        return pst.executeQuery();
}
```

Các thao tác cơ bản trên CSDL

Đọc dữ liệu

```java
try {
    ResultSet rs = DAL.getResultSet("select P.*, C.name as categoryname from
    product as P "
    + "inner join category as C on P.cat = C.id where P.cat=?", 1, Types.INTEGER);
    if (rs == null) {
        throw new SQLException();
    }
    while (rs.next()) {
        System.out.println(rs.getString("categoryname"));
    }
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Các thao tác cơ bản trên CSDL

Thêm / Xóa / Sửa trên CSDL

```
static public int executeNonQuery(String query, Object... params) throws
    SQLException, ClassNotFoundException {
        con = createConnection();
        PreparedStatement pst = con.prepareStatement(query);
        int j=1;
        for (int i = 0; i < params.length-1; i+=2) {
            pst.setObject(j, params[i], (int) params[i+1]);
            j++;
        }
        int kq = pst.executeUpdate();
        con.close();
        return kq;
}
```

Các thao tác cơ bản trên CSDL

Thêm / Xóa / Sửa trên CSDL

```
String query = "insert into product(name,price,description) values (?,?,?)";
try {
    int kq = DAL.executeNonQuery(query, "product demo", Types.VARCHAR, 100,
    Types.DECIMAL, "no description",        Types.LONGVARCHAR);
    if (kq == 1) {
        System.out.println("1 row effected");
    }
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Các thao tác cơ bản trên CSDL

Metadata - Get table names

```
static public ArrayList getTableMetadata() throws Exception {
        con = createConnection();
        DatabaseMetaData meta = con.getMetaData();
        String[] table = {"TABLE"};
        ResultSet rs = null;
        ArrayList tables = new ArrayList();
//          System.out.println(meta.getDriverName() + "\n" + meta.getURL());
        rs = meta.getTables(null, null, null, table);
        while (rs.next()) {
            tables.add(rs.getString("TABLE_NAME"));
        }
        con.close();
        return tables;
}
```

Các thao tác cơ bản trên CSDL

Metadata - Get column names

```
static public ArrayList getColumnsMetadata() throws Exception {
    con = createConnection();
    DatabaseMetaData meta = con.getMetaData();
    String[] table = {"TABLE"}; ResultSet rs = null;
    ArrayList tables = getTableMetadata();
    ArrayList columns = new ArrayList();
    for(Object t:tables){
        rs = meta.getColumns(null, null, (String) t, null);
        while(rs.next()){columns.add(rs.getString("COLUMN_NAME"));
        //"COLUMN_SIZE" or "TYPE_NAME"
        }
    }
    con.close();    return columns;
}
```

—Hết—