Getting Connection

JDBC

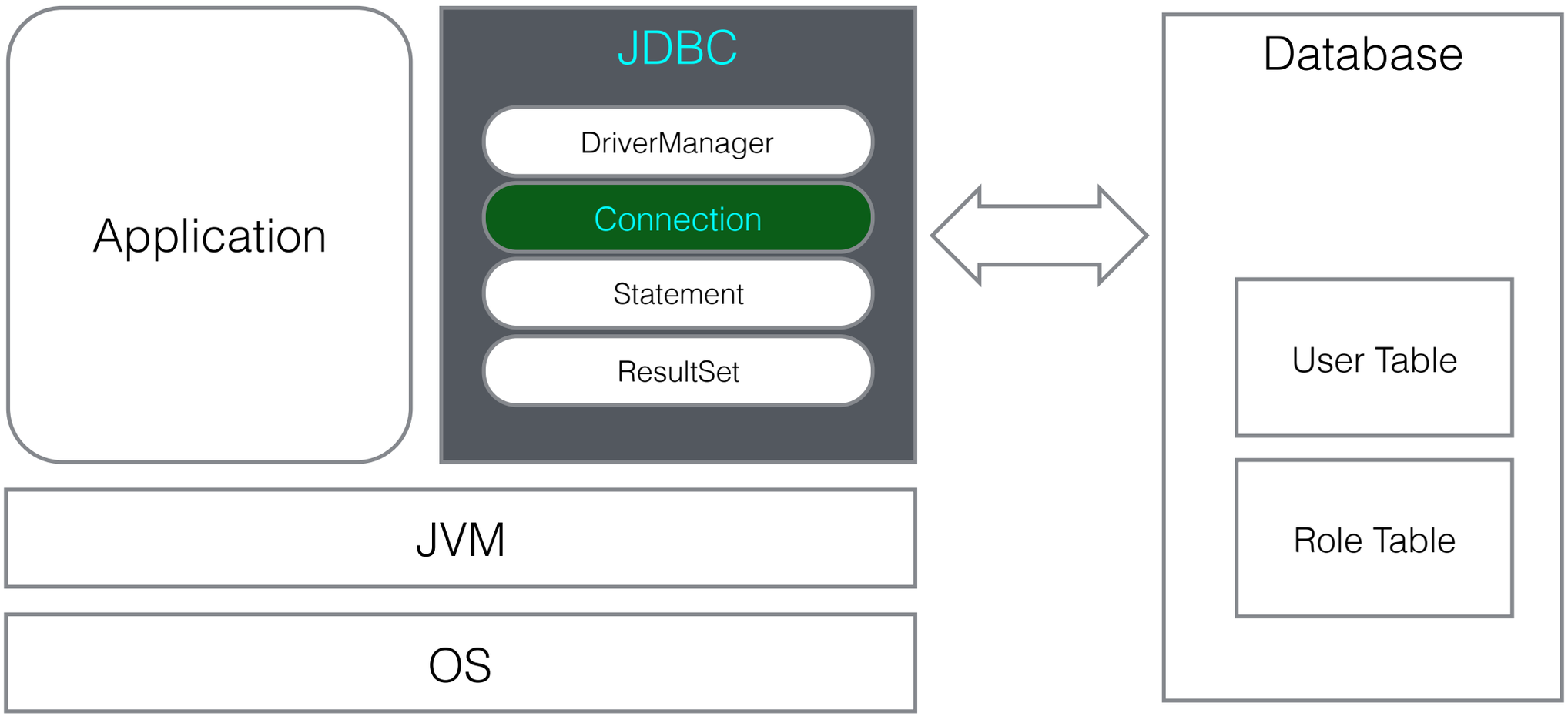
Contents

- What is Connection?
- DriverManager Class
- What can we do with Connection?
- How to get Connection?
- Closing Connection and Closable Interface

Connection

- Enterprise Application အတော်များများမှာ Data များကို အသုံးပြုရလွယ်ကူစေရန် DBMS များဖြင့် Management ပြုလုပ်လေ့ရှိကြသည်
- Application များမှ Database များနှင့် ချိတ်ဆက် အသုံးပြုနိုင်ရန် Connection Object များကို အသုံးပြုပါသည်
- အဆိုပါ Connection Object အားအသုံးပြုကာ လိုအပ်သော လုပ်ဆောင်ချက်များကို ဆောင်ရွက်နိုင်ပါသည်

JDBC API

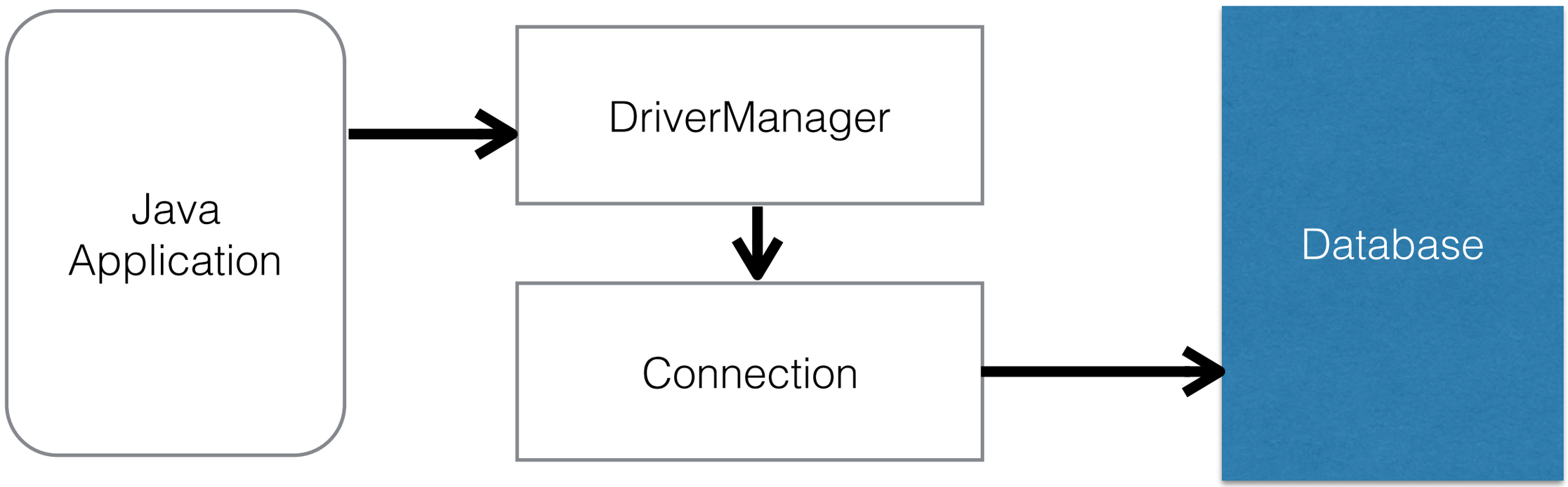

Application
JDBC
DriverManager
Connection
Statement
ResultSet
Database
User Table
Role Table
JVM
OS

What can we do?

- Connection Object ဟာ Database ကို အသုံးပြုဖို့ အတွက် လိုအပ်တဲ့ အဓိက Interface Object ဖြစ်ပါတယ်
- Connection ကို အသုံးပြုပြီး အောက်ပါ လုပ်ဆောင်ချက်များကို လုပ်ဆောင်နိုင်ပါတယ်
  - Statement Object များကို ရယူခြင်း
  - Database Metadata တွေကို ရယူနိုင်တယ်
  - Transaction Control ကို လုပ်ဆောင်နိုင်တယ်
  - Transaction Isolation Level တွေကို သတ်မှတ်နိုင်တယ်

DriverManager

- Java Application ထဲကနေ Connection Object ကို ရယူဖို့ အတွက် DriverManager Class ကို အသုံးပြုနိုင်



Informations that used to connect Database

- URL
  Database Application ကို ချိတ်ဆက်နိုင်မည့် URL

```
jdbc:mysql://localhost:3306/mydb
[protocol] //[host]:[port]/[schema]
```

- User
  Database User Name

- Password
  Database User's Password

How to get

- DriverManager ရဲ့ getConnection Method ကို အသုံးပြုနိုင်ပါတယ်

- getConnection method မှာ Argument မတူပဲ Overload လုပ်ပြီး ရေးထားတဲ့ static method သုံးခုရှိပါတယ်

    - String Object တစ်ခုကို အသုံးပြု

    - Properties Object ကို အသုံးပြု

    - String Object သုံးခုကို အသုံးပြု

Get Connection with one String Object

```
String url = “jdbc:mysql://localhost:3306/mddb”;
String query = “?user=root&password=admin”;

Connection conn =
    DriverManager.getConnection(url + query);
```

Get Connection with Properties Object

```
String url = “jdbc:mysql://localhost:3306/mddb”;
Properties props = new Properties();
props.put(“user”, “root”);
props.put(“password”, “admin”);

Connection conn =
    DriverManager.getConnection(url, props);
```

Get Connection with three String Object

```
String url = “jdbc:mysql://localhost:3306/mddb”;
String user = “root”;
String pass = “admin”;

Connection conn =
    DriverManager.getConnection(url, user, pass);
```

Closing Resources

- Connection, Statement တွေနဲ့ ResultSet Object တွေ ဟာ ပြင်ပမှ Resource တွေနဲ့ ချိတ်ဆက် ထားတဲ့ Resource Object တွေဖြစ်တယ်

- အဲ့ဒီ Object တွေကို အသုံးပြုပြီးရင် ပြန်ပိတ်ပေးဖို့ လိုအပ်ပါတယ်

- Close မလုပ်ပေးရင် Insert လုပ်ထားတာ တွေ Update လုပ်ထားတာတွေ ဟာ Database ထဲကို တကယ် ရောက်ရှိသေးမှာ မဟုတ်ဘူး

Closing Resource

```
try {
  Connection conn = DriverManager.getConnection(url);
  Statement stmt = conn.createStatement(sql);

  // codes

  stmt.close();
  conn.close();

} catch(Exception e) {
  e.printStackTrace();
}
```

Closing Resources

Closable Interface

- Resource Object တွေကို Close လုပ်တဲ့အခါမှာ အဲ့ဒီ Object တွေရဲ့ close() method ကို ခေါ်ပေးရမှာ ဖြစ်ပါတယ်

- close() method ကို ခေါ်တဲ့အခါမှာ Check Exception တွေကို ဖြစ်ပေါ်စေမှာ ဖြစ်တယ်

- ဒါကြောင့် JDBC Code တွေကို ရေးတဲ့အခါမှာ Business Logic နဲ့မဆိုင်တဲ့ Exception Control ကုဒ်တွေကို ရေးနေရတာမျှားပါတယ်

- Java SE 7 ကနေစပြီး try with resource statement အသစ်ကို ဖြည့်စွက်လာပြီး ၎င်းကို အသုံးပြုပါက close method ကို အသုံးပြုပြီးတဲ့အခါမှာ အလိုအလျောက် ခေါ်ဆိုပေးနိုင်မှာ ဖြစ်တယ်

After Java SE 7

Try with resources

```
try (Connection conn = DriverManager.getConnection(url);
  Statement stmt = conn.createStatement(sql)){

  // codes


} catch(Exception e) {
  e.printStackTrace();
}
```