LightHouseIT

2章 JDBC

JDBC

- JDBC入门　JDBC入門
- Statement　Statementとは
- 事務　トランザクション
- DAO/DTO　DAO/DTOパターン

目录

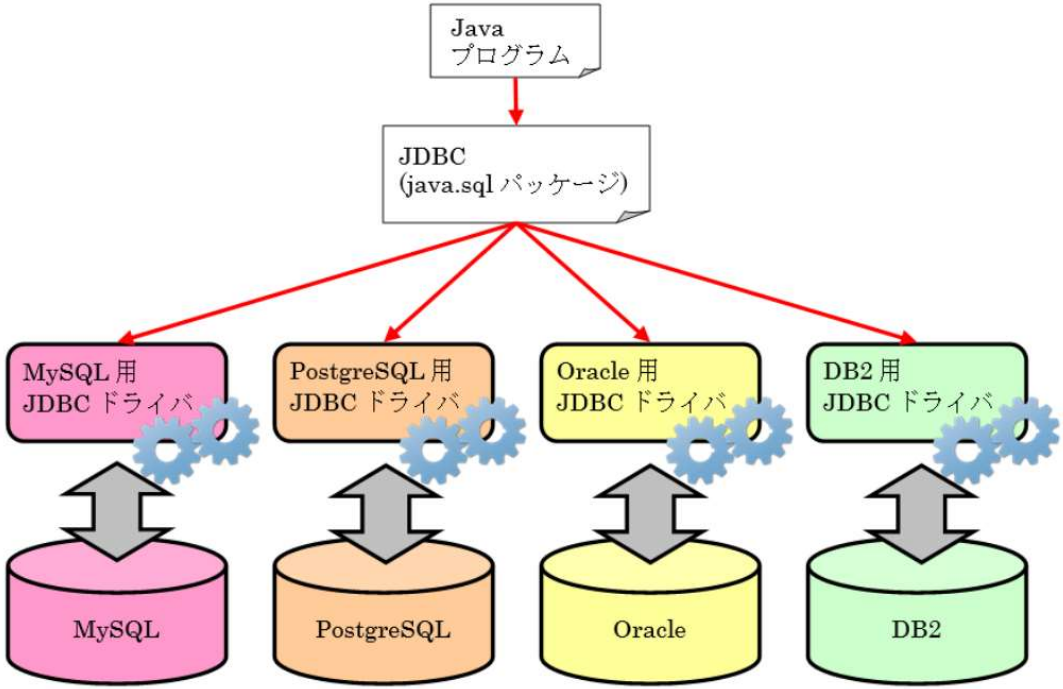
LightHouseIT

JDBC概述

- 基本介绍

JDBC是为了访问不同的数据库提供的同一接口。并由各个数据库厂商提供实现（类），一般以.jar文件提供，被称为驱动。
Java程序员使用JDBC，可以链接任何提供了JDBC驱动程序的数据库系统，从而完成对数据库的各种操作。

使用以下代码
模拟JDBC逻辑
JdbcInterface.java
MysqlJdbcImpl.java
OracleJdbcImpl.java
TestJDBC.java



JDBC使用步骤

1. 注册驱动 - 加载Driver类
2. 获取链接 – 得到Connection
3. 执行增删改查 – 发送SQL至数据库
4. 释放资源 – 关闭链接

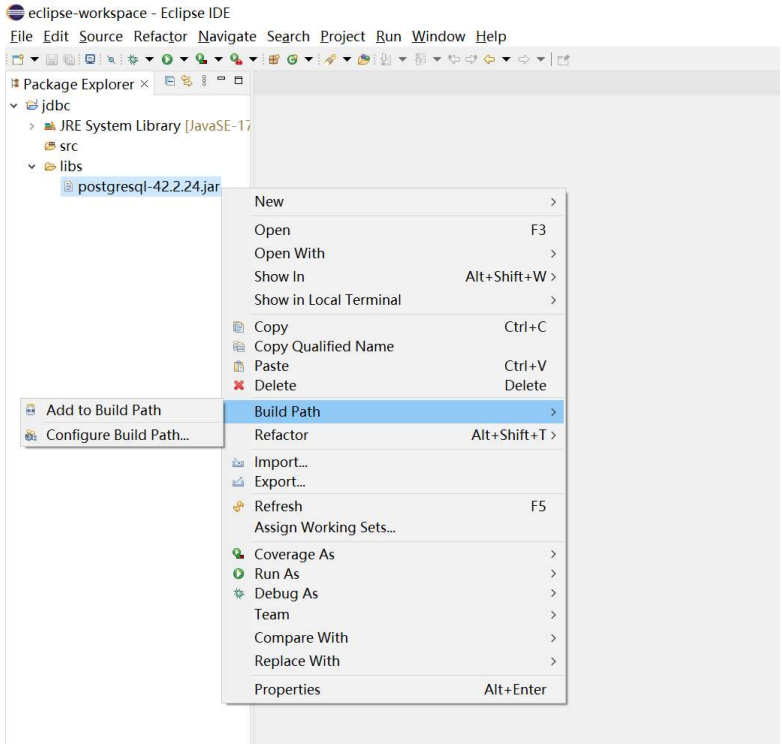
使用代码Jdbc01.java尝试链接数据库并获取所有数据。

JDBC快速入门

1. 注册驱动 - 加载Driver类

复制postgresql-42.2.24.jar到项目的libs文件夹中，
右键点击Build Path→Add to Build Path 将驱动添加到项目当中。

```
//使用postgresql驱动
import org.postgresql.Driver;
    //前置工作:  在项目下创建一个文件夹比如 libs
    // 将 postgresql-42.2.24.jar 拷贝到该目录下，点击 add to project ..加入到项目中
    //1. 注册驱动
    Driver driver = new Driver(); //创建driver对象
```



代码
加载Driver的不同方法
JdbcConn.java

Properties配置文件

- 例如java应用通过JDBC连接数据库时，通常需要在代码中写数据库连接字符串，让我们看看下面的代码：

```
//创建url 和 user 和 password
        String url = "jdbc:postgresql://localhost:5432/postgres";
        String user = "postgres";
        String password = "123456";
        Connection connection = DriverManager.getConnection(url, user, password);
```

在代码中，我们写入了数据库的登录用户名以及密码，而将重要信息直接写在代码中的做法是非常不安全的！因此，我们需要一种更安全的办法，那就是把这些信息保存在别的安全的地方，比如说某些文件里，比如说properties文件。

Properties配置文件

Properties文件是java中很常用的一种配置文件，文件后缀为“.properties”，属文本文件，文件的内容格式是“键=值”的格式，可以用“#”作为注释，java编程中用到的地方很多，运用配置文件，可以便于java深层次的解耦。让我们看看下面的代码：

```
//通过Properties对象获取配置文件的信息
        Properties properties = new Properties();
        properties.load(new FileInputStream("src\\postgresql.properties"));
        //获取相关的值
        String user = properties.getProperty("user");
        String password = properties.getProperty("password");
        String driver = properties.getProperty("driver");
        String url = properties.getProperty("url");
```

Properties文件中我们写入了一下信息：

```
user=postgres
password=123456
url=jdbc:postgresql://localhost:5432/postgres
driver=org.postgresql.Driver
```

这会让我们的程序更安全且更方便，比如数据库密码修改的时候，或是我们切换链接别的数据库等情况时，我们只需要修改文件，不需要修改代码就可以了。

JDBC快速入门

2. 获取链接 – 得到Connection

准备好数据库链接需要的信息：数据库的URL，用户信息

通过驱动的connect方法创建链接。

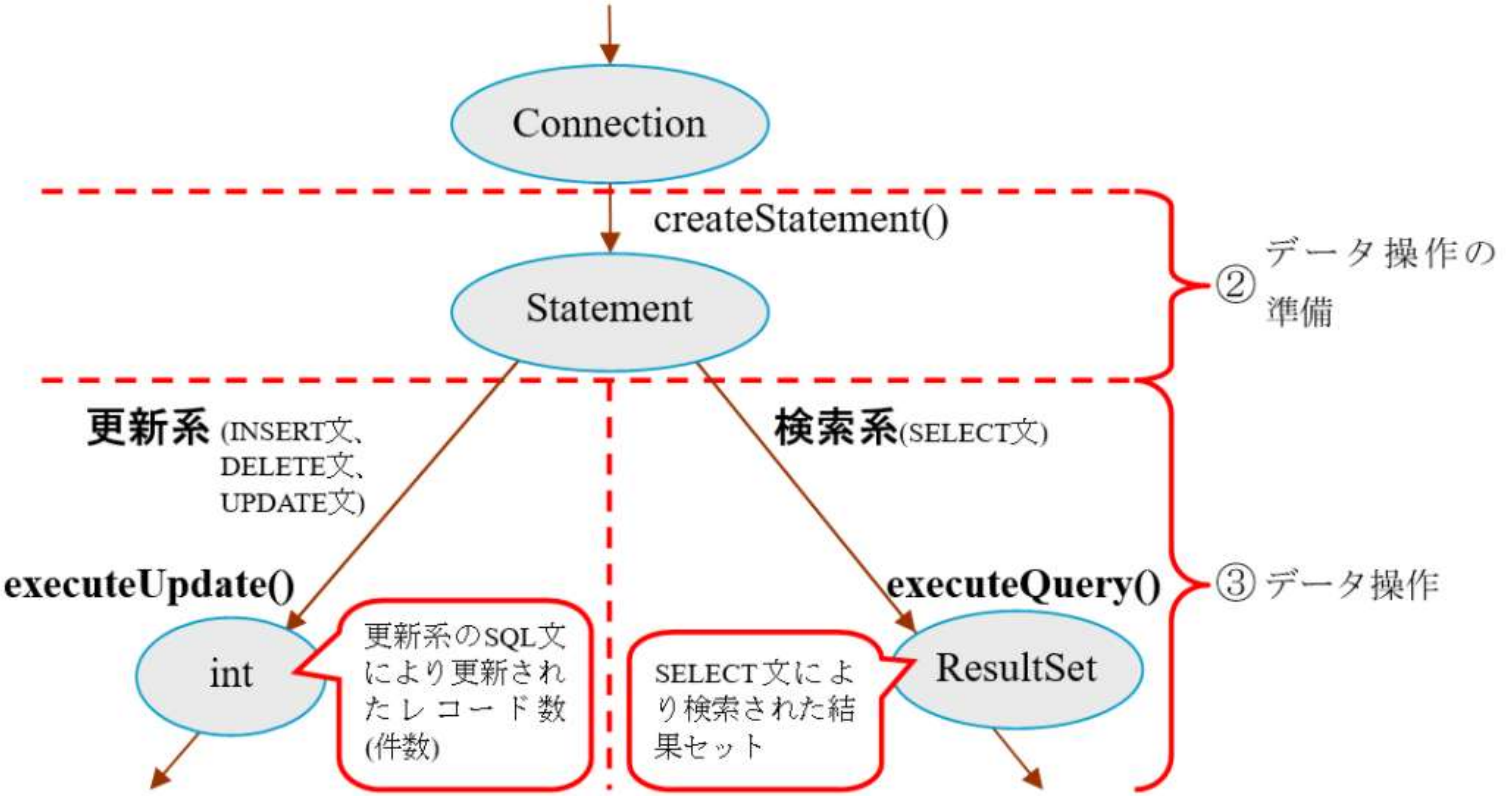
```
//(1) jdbc:postgresql:// 规定好表示协议，通过jdbc的方式连接postgresql
//(2) localhost 主机，可以是ip地址
//(3) 5432 表示postgresql监听的端口
//(4) postgres 连接到的哪个数据库
String url = "jdbc:postgresql://localhost:5432/postgres";

//将用户名和密码放入到Properties对象
Properties properties = new Properties();

//user 和 password 是规定好的，后面的值根据实际情况写
properties.setProperty("user", "postgres");// 用户名
properties.setProperty("password", "123456"); //密码
Connection connect = driver.connect(url, properties);
```

JDBC快速入门

3. 执行增删改查 – 发送SQL至数据库
   使用createStatement()方法，为发送SQL语句做准备。
   查询：executeUpdate()，返回结果为ResultSet类型，是查询结果
   增删改：executeQuery()，返回结果为Int类型，为操作行数。



JDBC快速入门（查询例）

3. 执行增删改查 – 发送SQL至数据库
   查询：executeUpdate()，返回结果为ResultSet类型。
   通过getXXXXX方法来获取指定列的值，这里用了getString方法。

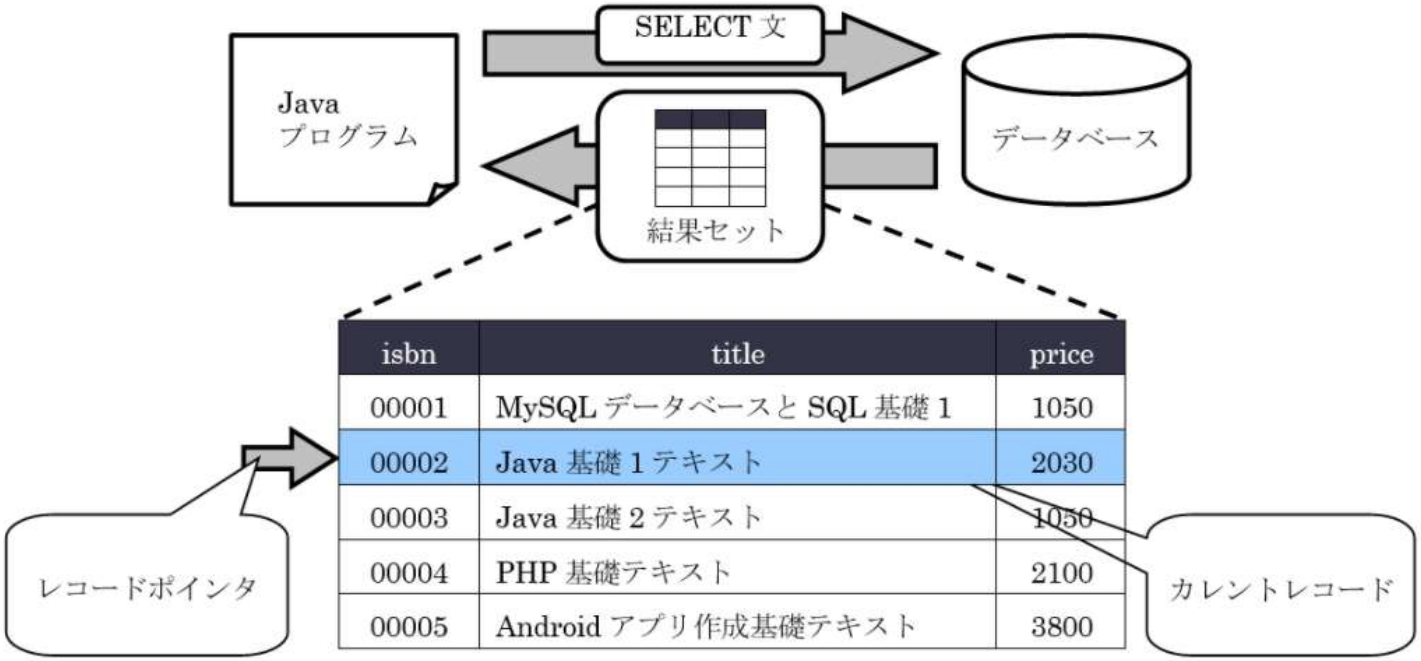
```
//3. 执行sql
String sql = "select * from student";
//statement 用于执行静态SQL语句并返回其生成的结果的对象
Statement statement = connect.createStatement();
ResultSet rs = statement.executeQuery(sql);

//输出全部查询结果的id，name，age
while (rs.next()) {
    System.out.println("id -> " + rs.getString("id") +
            " name -> " + rs.getString("name") +
            " age-> " + rs.getString("age"));
}
```

JDBC快速入门- 查询结果

使用 executeQuery() 方法查找数据时，结果集作为 ResultSet 类型对象的返回值返回，但在使用对象中的数据时使用了“指针”（被引用列的位置信息）的概念。

ResultSet 类型对象以类似于数据库表的二维表格式管理数据。因此，为了引用结果集中的数据，需要明确“引用了数据的哪一行、哪一列”。那时，指示它是哪一行的内部标记称为“记录指针”。此外，记录指针所在的可引用当前行称为“当前记录”。



JDBC快速入门 - 增

3. 执行增删改查 – 发送SQL至数据库
   增删改：executeQuery()，返回结果为Int类型，为操作行数。

```
//增添
    String sqlAdd = "INSERT INTO student(id,name,age,subject,gender) VALUES('5','jdbctest','1','jdbc','male')";
    int rs2 = statement.executeUpdate(sqlAdd);
    System.out.println(rs2);
```

	id [PK] integer	name character (50)	age integer	subject character (50)	gender character (50)
1	0	Mike	20	python	male
2	1	Susan	18	java	female
3	2	Caroline	20	python	female
4	3	Peter	21	java	male
5	4	David	22	java	male
6	5	jdbctest	1	jdbc	male

JDBC快速入门 - 改

3. 执行增删改查 – 发送SQL至数据库
   增删改：executeQuery()，返回结果为Int类型，为操作行数。

```
//修改
    String sqlUpdate = "UPDATE student SET age='10' WHERE id='5'";
    int rs3 = statement.executeUpdate(sqlUpdate);
    System.out.println(rs3);
```

	id [PK] integer	name character (50)	age integer	subject character (50)	gender character (50)
1	0	Mike	20	python	male
2	1	Susan	18	java	female
3	2	Caroline	20	python	female
4	3	Peter	21	java	male
5	4	David	22	java	male
6	5	jdbctest	10	jdbc	male

JDBC快速入门 – 删

3. 执行增删改查 – 发送SQL至数据库
   增删改：executeQuery()，返回结果为Int类型，为操作行数。

```
//修改
    String sqlUpdate = "DELETE from student WHERE id='5'";
    int rs4 = statement.executeUpdate(sqlUpdate);
    System.out.println(rs4);
```

	id [PK] integer	name character (50)	age integer	subject character (50)	gender character (50)
1	0	Mike	20	python	male
2	1	Susan	18	java	female
3	2	Caroline	20	python	female
4	3	Peter	21	java	male
5	4	David	22	java	male

JDBC快速入门

4. 释放资源 – 关闭链接

```
statement.close();
connect.close();
```

将数据库与 Statement 断开连接。当不再需要连接到数据库时，**一定要断开连接，释放资源**。

这里可以使用**Try catch-finnaly**语句进行数据库访问，确保资源释放。

总结

- 执行 SELECT 语句时，使用 executeQuery() 方法。
- 使用 executeUpdate() 方法执行 INSERT、UPDATE 和 DELETE 语句。
- 使用close()方法保证释放资源，因为它可能会影响系统性能。
- 为确保资源被释放，使用 finally 块。

JDBC練習問1

次の設問①～⑤について○か×で答えなさい。

設問
① JDBCはjava.sqlパッケージとJDBCドライバから構成されている。
② JDBCドライバはプロジェクトへコピーすれば利用することができる。
③ Class.forName()メソッドでJDBCドライバの読み込みを行うことができる。
④ DriverManager.getConnection()メソッドでデータベースとの接続を確立する。
⑤ DriverManager.getConnection()メソッドの引数には、URL,ユーザー名,パスワードが必須である。

JDBC練習1（回答）

① ×：JDBCは「java.sqlパッケージ」のことを指します。
② ×：ビルド・パスへ追加する必要があります。
③ ○
④ ○
⑤ ×：設定が無い場合はユーザー名、パスワードを省略することも可能です。

JDBC練習問2

次の設問①～⑤について○か×で答えなさい。

設問

① データの検索を行う場合、executeUpdate()メソッドを利用する。
② SELECT文を実行した結果セットはStatement型のオブジェクトとして返される。
③ 例外が発生した場合、プログラムは終了するためリソースの開放は必要ない。
④ 更新系のSQLを実行するプログラムの場合、処理の中で異なるのはSQL文だけである。
⑤ executeUpdate()メソッドの戻り値はSQL文が成功したかどうかの真偽値である。

JDBC練習2（回答）

① ×：executeQuery ()メソッドを利用します。
② ×：ResultSet型のオブジェクトとして返されます。
③ ×：例外が発生した場合でもリソースの開放は確実に行う。
④ ○
⑤ ×：更新件数がint型の値として返されます。

目录

LightHouseIT

Statement

- Statement是一个接口，用于执行静态SQL语句并返回其生产的结果的对象。
- 在链接建立后，需要对数据库进行访问，执行命名或是SQL语句，可以通过以下几种方法。

1. Statement
2. PreparedStatement
3. CallableStatement

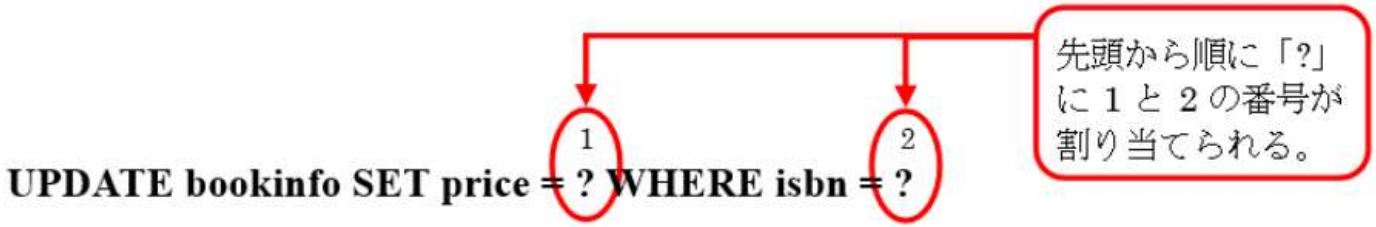
- Statement对象执行SQL语句，存在SQL注入风险，实际开发中不使用。
- SQL注入式利用某些系统没有对用户输入的数据经行筛选排查，恶意用户输入中输入SQL语句命令，达到访问攻击数据库的行为。

- 要防范SQL注入，只要用PreparedStatement即可。

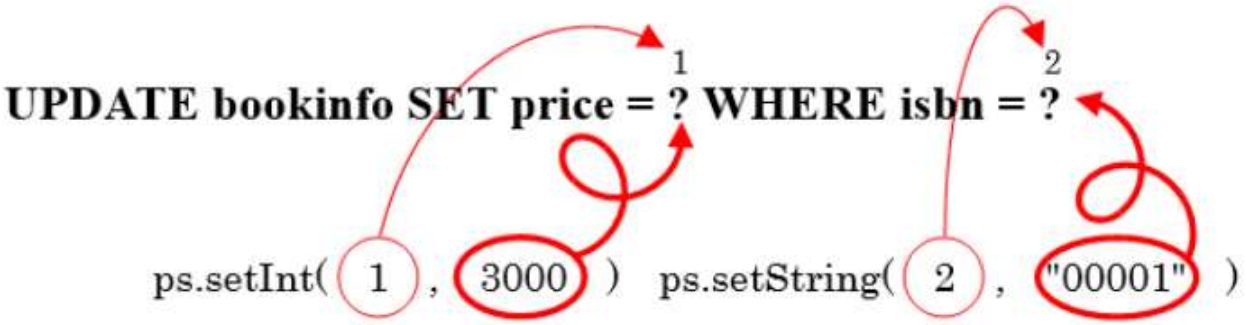
了解什么是SQL注入：
sql_injection.sql
Statement.java

PreparedStatement

- PreparedStatement执行的SQL语句中的参数用问号(?)来表示



- 调用PreparedStatement对象的setXxx()方法来设置这些参数，setXxx()方法由两个参数，第一个参数是要设置的SQL语句中的参数的索引（从1开始），第二个是设置SQL语句中的参数的值



- 调用executeQuery()，返回ResultSet对象
- 调用executeUpdate()，执行更新，增添，删除，修改

PreparedStatement

使用PreparedStatement方法有以下几点好处

1. 不再使用加号+拼接SQL语句，降低语法错误的可能性。
2. 有效解决SQL的注入问题。
3. 减少了编译次数，提高了代码效率。

使用以下代码尝试PreparedStatement的增删改查，并确认SQL注入是否被解决。

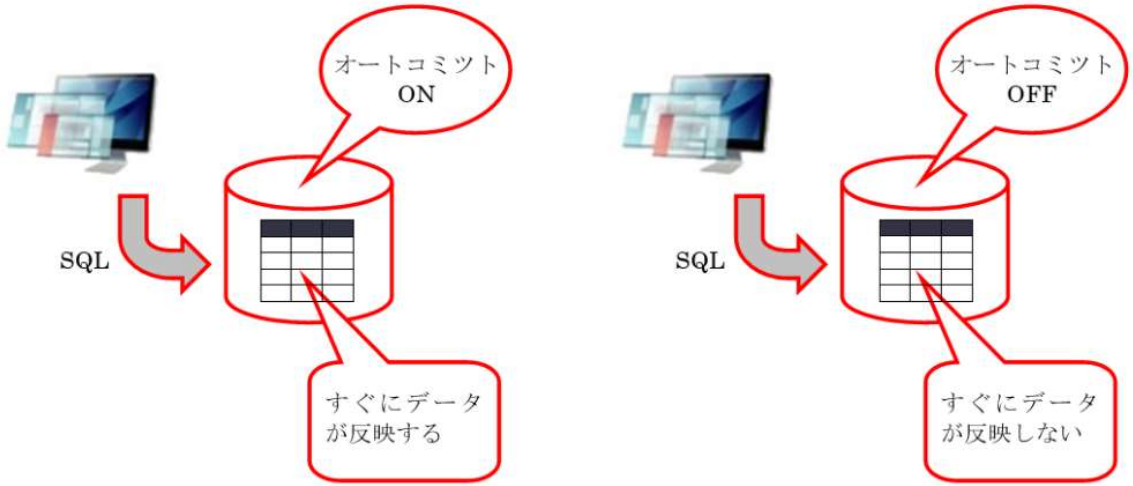
PreparedStatementQuery.java

PreparedStatementDML.java

LightHouseIT

事务[トランザクション]

- JDBC程序中当一个Connection对象创建时，默认情况下是自动提交事务：每次执行一个SQL语句时，如果执行成功，就会向数据库自动提交，不能回滚。



- JDBC程序中为了让多个SQL语句作为一个整体执行，需要使用事务。
- 调用Connection的setAutoCommit(false)可以取消自动提交事务。
- 在所有的SQL语句都执行成功后，调用commit()方法提交事务。
- 在其同某个操作失败或出现异常时，调用rollback()方法回滚事务。

代码Transaction_.java

JDBC練習問3

次の設問①～⑤について○か×で答えなさい。

設問

①　MySQLのオートコミット機能は初期状態では無効になっている。
②　トランザクション処理中は、何度SQL文を実行してもデータには反映されない。
③　PreparedStatement を使う場合、SQLのパラメータ変数には「$」マークを使用する。
④　パラメータ変数に値をセットする場合、セットする値の型によってメソッドが異なる。
⑤　基準となるSQLが複数ある場合、PreparedStatementは利用できない。

JDBC練習3（回答）

① ×：初期状態のオートコミット機能は有効である。
② ○
③ ×：「?」(はてなマーク)を使用する。
④ ○
⑤ ○

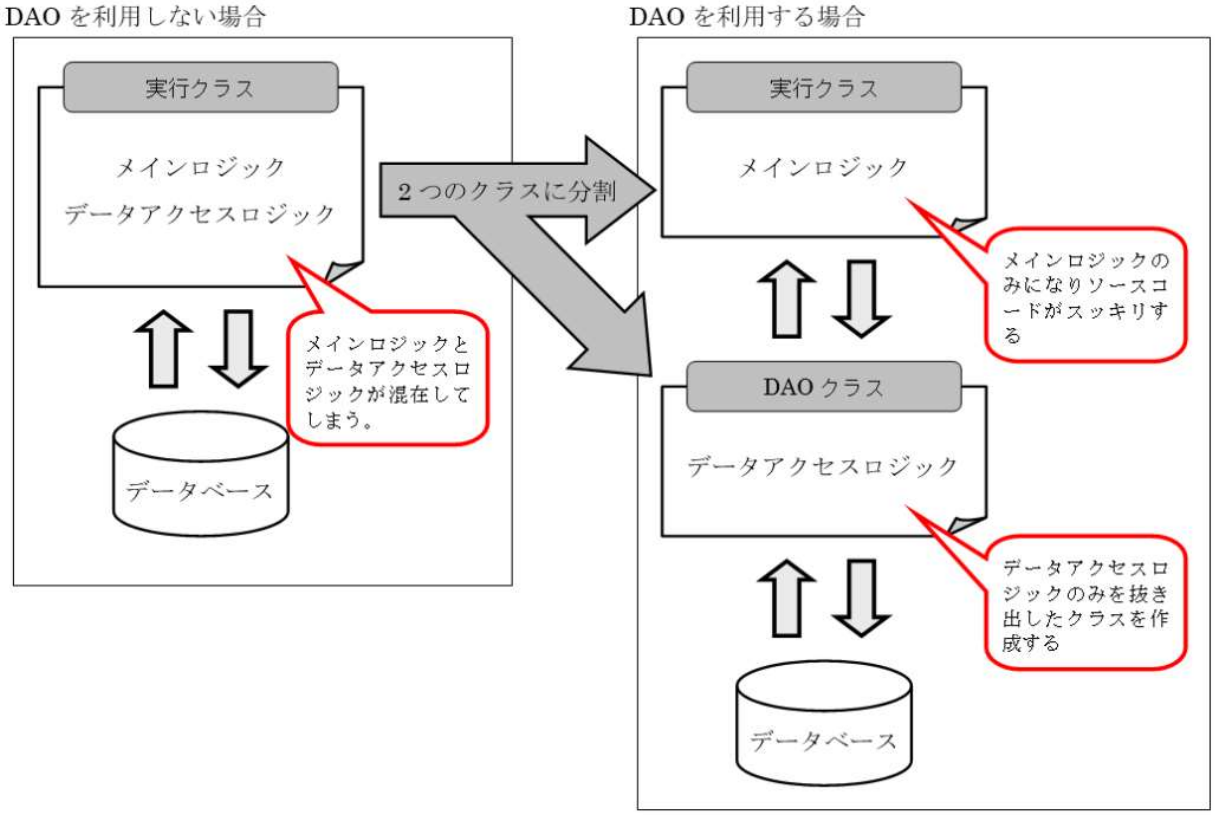
目录

LightHouseIT

DAO模式

在创建使用 JDBC 的应用程序时，经常使用称为 DAO 模式和 DTO 模式的两种程序设计模式。这两种设计模式通常是一起被使用的，我们先介绍DAO模式。

DAO（数据访问对象）模式是一种设计模式，它创建一个访问数据库的类，并通过该类访问数据库。将主逻辑中描述的访问部分的处理合并为一个类，具有数据库访问窗口的作用。



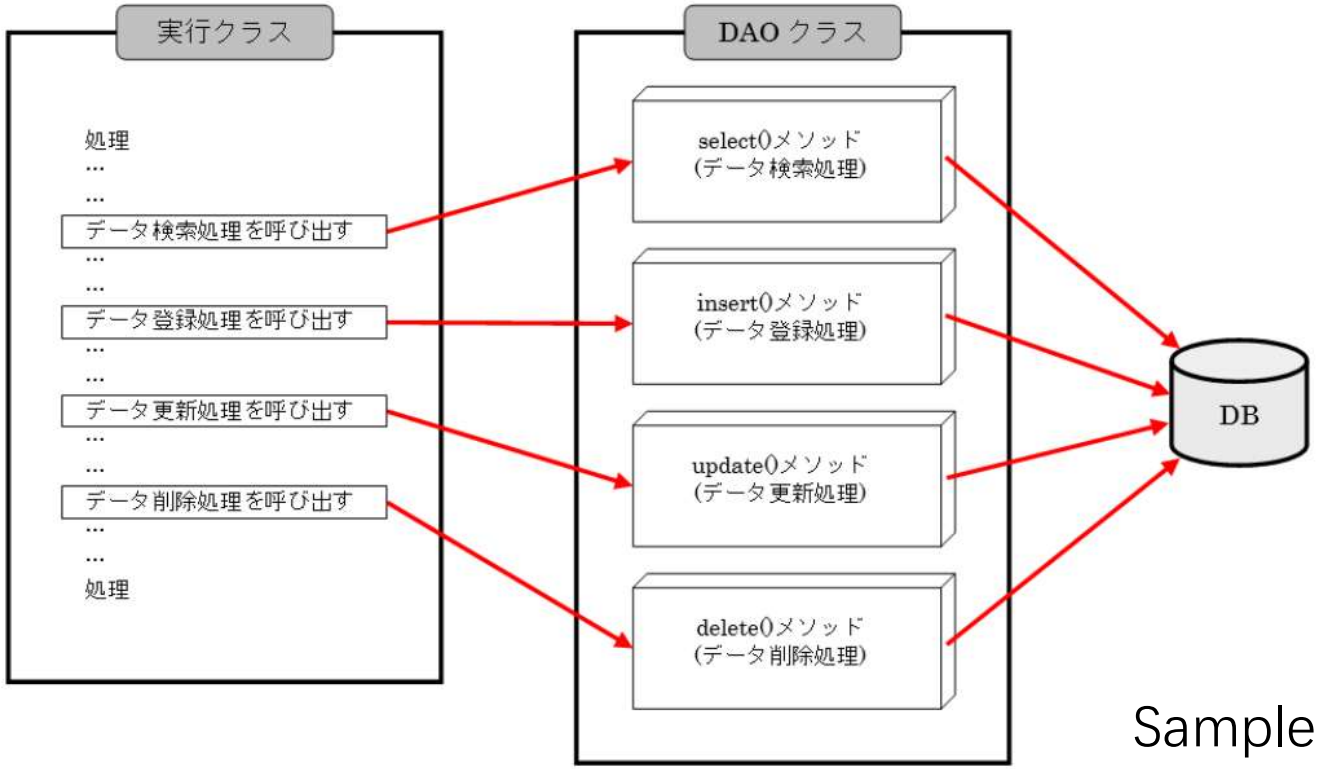
DAO模式的基本配置和好处

使用 DAO 模式的程序在设计时明确区分了主逻辑和数据访问过程。因此，它具有以下优点。

•通过将应用程序内部的主要逻辑和数据库访问处理明确分离，可以增加独立性和可扩展性，使更改和修改不会影响彼此的类。

•由于数据库访问功能被集成到一个类中，因此很容易识别访问位置。

•防止重复描述数据库访问处理并简化源代码。

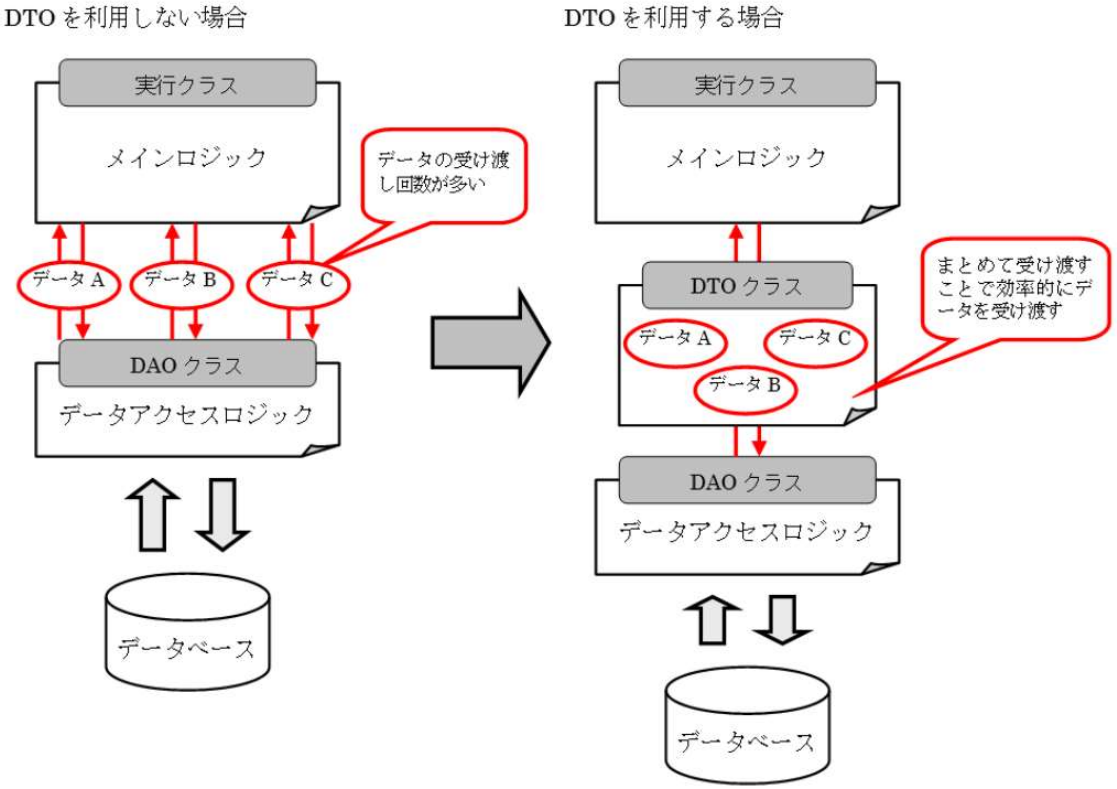


SampleDAO.java

DTO模式

如果只使用 DAO 模式，在主逻辑和数据访问中会出现一些低效的部分，比如说我们需要针对每一个成员变量定义一个方法。

DTO（Data Transfer Object）是基于JavaBeans（JavaBeans）概念的“专用于数据传输的类”。通常定义字段变量来管理数据以及与字段变量对应的 setter 和 getter。数据传输很容易，通过ArrayList或数组来管理不同类型的数据。另外可以减少数据库访问量而提高程序效率。



参考代码
SampleDTO.java
SampleDAO2.java

总结

- 使用 JDBC 的程序设计模式之一是称为 DAO/DTO 模式。
- DAO 是一种将数据访问逻辑聚合到一个类中并创建程序的设计模式。
- DTO 是一种设计模式，它定义了一个专用于数据传输的类。
- 通过结合使用ArrayList和DTO，SELECT语句的结果集可以由一个对象管理。

JDBC練習問4

次の設問①～⑤について○か×で答えなさい。

設問

①　DAOパターンとはデータアクセスロジックを1つのクラスに集約しプログラムするデザインパターンである。
②　DTOパターンとはデータの受け渡し専用のクラスを用意するデザインパターンである。
③　DAOクラスはJavaBeansの概念が基にしたデザインパターンである。
④　DAOパターンとDTOパターンはセットで使用してはならない。
⑤　DTOクラスとArrayListを組み合わせることで異なる型のデータをまとめて管理できる。

JDBC練習4（回答）

①　○
②　○
③　×：DTOクラスがJavaBeansの概念を基にしたデザインパターンである。
④　×：DAOパターンとDTOパターンはセットで利用するのが一般的である。
⑤　○