<Java側>

DB Connector=自分で作るクラス

Driver managerに対してどんなものを使うのか等の細かい情報を示す。

どのドライバをつかって、どのmanagerを使うのかを定義して接続を確立させるためのもの。

1.DriverManager（どのドライバを使うのかを教えてあげる）

2.Connection

3.PreparedStatement

4.ResultSet

Java側で用意してあるDBとつなぐもの。自分で作る必要はない。

<データーベース側>

**MySQL用ドライバ(jar)=mysql\_connector\_java-xxx.bin.jar**

オラクル用(jar)

XX用(jar)

Jar=java archive(java ファイルを集めて圧縮したもの)

↓

DB（mySQL等に接続）

com.mysql.jdbc=パッケージ名

Driver＝クラス名

“jdbc:mysql//localhost/testdb?

Localhostはデーターベースのサーバーの場所

testdbはデータベースの自分でつけた名前

public Connection getConnection(){

//名前はgetConnectionoじゃなくてもいい。Connection型の何かを返すもの

　connectionは確立したコネクション自体を変数とするもの。

↓

Connection con=null;

try{

Class.forName(driverName);　//変数で指定したドライバーを引数としてロード（mysqlのドライバ）

//getConnectionメソッドで（）のなかの要素をもって接続

↓

Con=(Connection) DriverManager.getConnection(url,user,password);

}catch(ClassNotFoundException e){

e.printStackTrace();　　　//例外の流れをコンソール上に表示するための処理

}catch (SQLException e){

e.printStackTrace();

}

return con;

}

**try,catch構文**

try{想定外のことが起こる可能性のある処理を括弧内に記入}

これで中身の文でエラーが起きてもそのままいける。普通はエラーになる。

Catch(例外の型、変数名){例外が発生した際に行う処理}

例外=想定外の事態

DAO (data access object)

クラスファイルに

DAOからDBConnectorを呼び出して処理を実行させる。

<TestUser DAO>

prepared statement=予め準備されたSQL文

select\*from test\_table where user\_name=? and password=?

?として不完全なものを変数psに代入

?にしておくことで編集可能。

↓

Ps.setString(1,name);

Ps.setString(2,password)

ここで?に変数を代入している。

↓

ResultSet rs=ps.executeQuery(); //ここで完成したsql文を実行する。

//executeQuery=sqlを実行するためのメソッド

ResultSet rs=ps.executeQuery();

↓

・next()

・get型名

取り出すメソッド。

If(rs.next())

テーブルを指し示す矢印をひとつ下に動かすイメージ。

そこに値が存在するならばtrueとなる

if(rs.next())

System.out.println(rs.getString(“user\_name”);

//データベースのテーブルからuser nameを出力

try{

con.close //最後に接続を解除する

<補足>

executeQueryはSelectメソッドのみ

crudのreadにしか対応していない。

Createの場合 execute update()メソッドを使う。

Int i=ps.executedUpdate;

If(i>0)

System.out.println(i+”件更新しました”)

変数iを更新の件数としている。

Page10

普通のexecutequeryだけでif文と重複に対応できない。

名前が太郎が二人いてそれぞれ異なるIDを保つ場合など一つしか見れない。

そういうときは

while(rs.next())メソッドを使用することで繰り返し処理が行われる