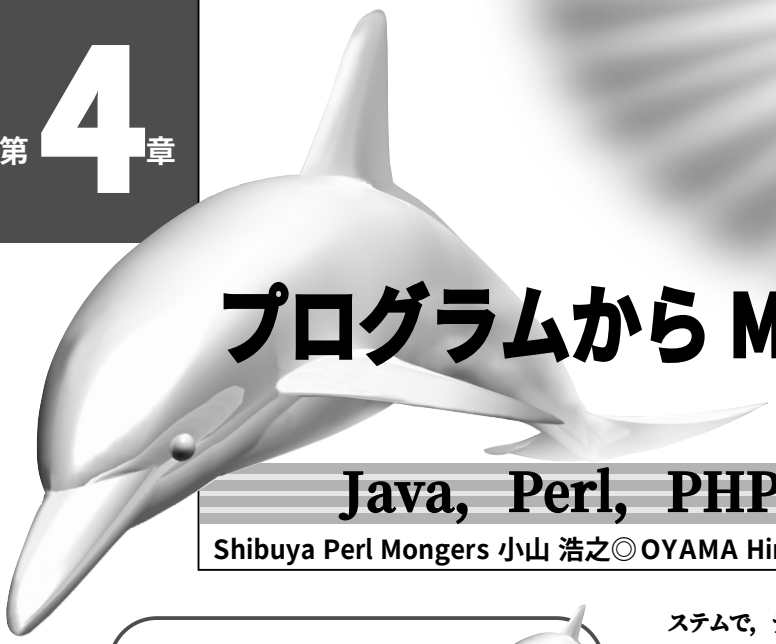
第4章

プログラムからMySQL操作

Java，Perl，PHPからの利用方法

Shibuya Perl Mongers 小山 浩之◎OYAMA Hiroyuki●oyama@cpan.org



はじめに

本章では，Java，Perl，PHPからMySQLを操作するプログラミングインタフェースを解説します．使用するメソッドや属性の名前はプログラミング言語とインタフェースごとに異なりますが，RDBMSを利用するプログラムは，

- 接続／切断
- エラー処理
- クエリ送信
- レスポンス受信

の4つの要素を把握していれば記述することができます．本章は各言語に用意されているプログラミングインタフェースを，これら4つの要素ごとに比較しながら解説します．

クライアントライブラリ

MySQLはクライアント／サーバ型のデータベースシステムで，データを管理するサーバと，それを利用するクライアントが協調して動作します．通常MySQLを利用する場合は，クライアント側のプログラムを記述することになります．

表1は，クライアントライブラリ各言語に用意されているプログラミングインタフェースの概要です．Java用に提供されているConnector/Jを除き，PerlのDBI，PHPのDBや他の多くの実装ではMySQLのネイティブライブラリであるlibmysqlclientをラッピングして使用する構成をとっています[注1]．

Connector/Jのセットアップ

Java用の実装の1つであるMySQL Connector/Jは，MySQLの開発元であるMySQL ABによってメンテナンスされているJDBC 3.0準拠のドライバです．現在安定版のバージョン3.0系と開発版の3.1系が配付されており，安定版の最新バージョンは3.0.8です[注2]．

Connector/Jの前身であるMM.MySQLドライバと同様にソースコードが公開されており，ライセンスはGNU General Public Licenseか，商用ライセンスのいずれかを選択できます．

▼表1　APIの概要

言語	API	実装	通信の特徴
Java	JDBC	MySQL Connector/J	通信プロトコルを独自実装
Perl	DBI	DBD::mysql	通信はlibmysqlclientを使用
PHP	DB	DB_mysql	通信はlibmysqlclientを使用

注1）現在はメンテナンスされていませんが，Javaと同様にPerlだけで記述されたライブラリNet::MySQLとDBD::mysqlPPという選択肢もあります．これらのライブラリは基本的にPerlとSocketが利用できる，あらゆる環境からMySQLを操作できます．

注2）http://www.mysql.com/products/connector-j/

Connector/Jをセットアップするには，http://www.mysql.com/downloads/api-jdbc-stable.htmlから“Source and Binaries (tar.gz)”をダウンロード・展開し，パッケージに含まれるmysql-connector-java-3.0.8-stable-bin.jarへのパスをCLASSPATHに追加するか，$JAVA_HOME/jre/lib/extディレクトリに設置するだけです（図1）．

DBD::mysqlのセットアップ

Perl用の実装の1つであるDBD::mysqlは，Jochen Wiedmann氏によってメンテナンスされているDBI用のドライバです．現在バージョン2.9002が配付されており，ライセンスはPerlと同じくGNU General Public LicenseかArtistic Licenseのいずれかを選択できます[注3]．

DBD::mysqlをセットアップする前に，まずlibmysqlclientライブラリと，そのヘッダファイルがインストールされている必要があります．

パッケージマネージャにRPMを使用している場合は，MySQL-develパッケージで必要なファイルをインストールできます．http://www.mysql.com/downloads/mysql-4.0.htmlから，使用しているアーキテクチャに対応したパッケージをダウンロードし，インストールしてください（図2）．

www.mysql.comでは，RPM以外にもSolaris，FreeBSD，Mac OS Xなど一般的なプラットフォーム向けのバイナリパッケージとソースパッケージを配付していますので，インストールする環境に合わせて選択してください．

次にDBIとDBD::mysqlモジュールをインストールします．これらはCPANモジュールを使用して，ネットワークインストールができます（図3）．

DB_mysqlのセットアップ

PHP用の実装であるDB_mysqlは，PEARのDBパッケージに含まれており，Stig Bakken氏によってメンテナンスされています[注4]．PerlのDBD::mysqlと同様にlibmysqlclientライブラリを使用するので，PHPのセットアップの前にlibmysqlclientライブラリと，そのヘッダファイルがインストールされている必要があります[注5]．

PHP4系列の場合は，libmysqlclientライブラリがインストールされた状態で，PHPのconfigureスクリプトに--with-mysqlオプションを指定し，PHPをビルドすればDB_mysqlが利用できるようになります（図4）．

現在開発中のPHP5系列の場合は，ライセンスの相違の問題でMySQLのクライアントライブラリはバンドルされていません．これについては第1章を参照してください．

◇ ◇ ◇

各プログラミングインタフェースのセットアップの詳細については，各パッケージのドキュメントを参照してください．

接続／切断

MySQLサーバへ接続し切断する手続きは，いずれ

図2●MySQL-develパッケージのインストール

```
% sudo rpm -i MySQL-devel-4.0.14-0.i386.rpm
```

図3●DBD::mysqlのインストール

```
% sudo perl -MCPAN -e 'install "DBD::mysql"'
```

図4●DB_mysqlのセットアップ

```
% cd php-4.3.2/
% ./configure --with-mysql
% make
% sudo make install
```

図1●Connector/Jのセットアップ

```
% tar zxvf mysql-connector-java-3.0.8-stable.tar.gz
% cd mysql-connector-java-3.0.8-stable/
% cp mysql-connector-java-3.0.8-stable-bin.jar $JAVA_HOME/jre/lib/ext/
```

注3）http://search.cpan.org/author/RUDY/DBD-mysql/
注4）http://pear.php.net/package-info.php?package=DB
注5）libmysqlclientライブラリのセットアップは，前項「DBD::mysqlのセットアップ」の説明を参照してください．

[特集3] 世界最速データベース MySQL 徹底攻略

のプログラミングインタフェースも同じような手続きをふみます．各言語で接続し切断するだけのプログラムは，リスト1～3のようになります．

いずれのプログラミングインタフェースも接続するデータベースの種類と場所，さらに細かなパラメータを文字列DSN（Data Source Name）で指定します．各インタフェースの接続用メソッドの引数を表2に，使用するDSNの例を表3にまとめます．

▼リスト1　Javaでの接続／切断

```
import java.sql.*;

public class ConnectDemo {
    public static void main(String argv[]) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost/dbname",
                    "user", "password");
            conn.close();
        }
        catch (SQLException e) {
            /* エラー処理 */
            e.printStackTrace();
        }
        catch (ClassNotFoundException e) {
            /* エラー処理 */
            e.printStackTrace();
        }
    }
}
```

▼リスト2　Perlでの接続／切断

```
#!/usr/bin/perl
use DBI;
use strict;

eval {
    my $conn = DBI->connect(
        'dbi:mysql:hostname=localhost;
                database=dbname',
        'user', 'password',
        { RaiseError => 1, PrintError => 0 }
    );
    $conn->disconnect();
};
if ($@) {
    # エラー処理
    warn $DBI::errstr;
}
__END__
```

▼リスト3　PHPでの接続／切断

```
<?php
require_once 'DB.php';

$conn = DB::connect(
    'mysql://user:password@localhost/dbname');
if (DB::isError($conn)) {
    /* エラー処理 */
    die($conn->getMessage());
}
$conn->disconnect();
?>
```

表3で示した例の他にも指定可能なパラメータが存在しますので，詳しくは各ドライバのマニュアルを参照してください．

エラー処理

エラー処理はプログラミングインタフェースの差異よりも，プログラミング言語の差異が顕著に表れます．大きく分類すると例外処理をサポートする言語と，サポートしない言語に分類することができます（表4）．

JavaやPerlのように例外処理をサポートする言語では，ブロック単位にエラー処理を記述するスタイルをとるので簡潔なコードになりますが，PHPのように例外処理をサポートしない言語では，各処理単位でその都度エラー処理を記述する必要があります．なおPHPのDBクラス（と親のPEARクラス）は，エラー時にコールバックされる関数を設定できますので，表4のように例外処理的な記述は可能です．

▼表2　接続メソッドの引数

言語	接続メソッドの引数
Java	java.sql.DriverManager.getConnection(String DSN, String USER, String PASSWD);
Perl	DBI->connect($DSN, $USER, $PASSWD, \%OPTIONS);
PHP	DB::connect($DSN, $OPTIONS);

▼表3　DSNの差異

言語	DSNの書式
Java	"jdbc:mysql:///dbname" "jdbc:mysql://hostname:port/dbname"
Perl	'dbi:mysql:dbname' 'dbi:mysql:dbname@hostname:port' 'dbi:mysql:database=dbname;host=hostname;port=port'
PHP	'mysql:///dbname' 'mysql://hostname:port/dbname' 'mysql://username:password@hostname:port/dbname'

▼表4　エラーハンドリングの作法

言語	例外処理	例
Java	○	try { code; } catch (Exception) { error_code; }
Perl	○	eval { code; }; if ($@) { error_code; }
PHP	△	code; if (DB::isError($obj)) { error_code; }

クエリ送信

RDBMSへ送信するクエリには，UPDATE文やINSERT文などの更新系SQL，SELECT文などの参照系SQLの2種類があります．更新系SQLの場合はクエリの送信後，影響を受けたレコード数などのチェックを行います．参照系SQLの場合はクエリの送信後，検索結果のレコードセットを受信します．

更新系SQL

UPDATE文やINSERT文などの更新系SQLを実行した後は，クエリが成功したか否か，また影響を受けたレコード数はいくつあったかを調べます．

表5は更新系SQLを実行する手続きの例です．各プログラミングインタフェース共に，SQLをもとに，SQLとSQLの状態を抽象化したステートメントハンドルを生成し，そのハンドルを通じてSQLを実行します．いずれの言語のメソッドも戻り値として，実行したSQLによって影響を受けたレコード数を返します．

参照系SQL

SELECT文などの参照系SQLを実行した後は，検索結果のレコードセットを受信します．

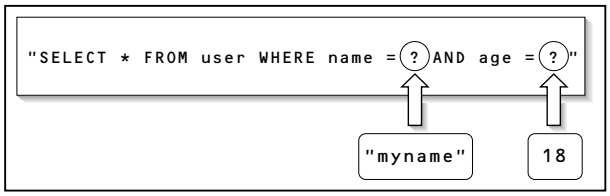
表6は各インタフェースで参照系SQLを実行する例です．メソッド名や手続きは似通っていますが，JavaとPHPはステートメントハンドルから生成した，レコードセットオブジェクトから値を参照しているのに対し，PerlのDBIはレコードセットオブジェクトを生成せずに，ステートメントハンドルから値を参照している点が微妙ながら大きく異なるポイントです．

またPerlとPHPは直接レコードの値を参照できますが，Javaの場合は受け取りたいデータ型に応じたアクセッサメソッドで値を取り出す必要があります．例で挙げたgetString( )メソッドの他に，getInt( )，getDate( )，getObject( )などのデータ型に応じたメソッドが用意されています．

プレースホルダ

送信するSQLを動的に組み立てる場合はプレースホルダを使用します．SQL文に“？”でマーキングした場所をプレースホルダと呼びます．プレースホルダを使用するSQL文はprepare系のメソッドでステートメントを生成し，実行前（もしくは実行時）に設定した値でSQL文中のプレースホルダを置き換え，クエリを実行します（図5）．プレースホルダに値を埋め込む際にエスケープ処理などが自動的に行われますので，SQL的に不正な文字を埋め込む場合でも，適切にエスケー

図5 ●プレースホルダとバインド値



▼表5　更新系SQLの実行

言語	手続き

Java	PreparedStatement query = conn.prepareStatement(SQL);
	int result = query.executeUpdate();
	query.close();

Perl	my $query = $conn->prepare($SQL); my $result = $query->execute(@params);
PHP	$query = $conn->prepare($SQL); $result = $conn->execute($query, $params_array);

▼表6　参照系SQLの実行

言語	手続き

Java	PreparedStatement query = conn.prepareStatement(SQL);
	ResultSet result = query.executeQuery();
	while (result.next()) {
	　String row0 = result.getString(1);
	　String row1 = result.getString(2);
	　String row2 = result.getString(3);
	}
	result.close();
	query.close();

Perl	my $query = $conn->prepare($SQL);
	my $result = $query->execute(@params);
	while (my $row = $query->fetch()) {
	　$row->[0];
	　$row->[1];
	　$row->[2];
	}
	$query->finish();

PHP	$query = $conn->prepare($SQL);
	$result = $conn->execute($query, $params_array);
	while ($row = $result->fetchRow()) {
	　$row[0];
	　$row[1];
	　$row[2];
	}
	$result->free();

プされます．プレースホルダの処理は，各プログラミングインタフェースのドライバによって実装されます．

表7のように，各プログラミングインタフェース共に，プレースホルダは“?”でマーキングします．値の設定方法はそれぞれ異なり，JavaはsetString( )やsetInt( )メソッドでIndexを指定して値を設定し，PerlとPHPはexecute( )メソッドの引数に配列で値を設定します．

外部からの入力値をSQLに埋め込む際に入力値を適切にエスケープしていない場合，悪意を持ったユーザに不正なSQLの実行を許してしまうSQL Injectionと呼ばれる問題が発生します．このセキュリティ面での問題を簡単かつ確実に回避できますので，動的にSQL文を組み立てる場合は特別な理由がない限りプレースホルダを使用してください．

▼表7　プレースホルダと値のバインド

言語	手続き

Java	`PreparedStatement query = conn.prepareStatement(`
	`  "SELECT * FROM user WHERE name = ? AND age = ?");`
	`query.setString(1, "myname");`
	`query.setInt(2, 18);`
	`ResultSet result = query.executeQuery();`

Perl	`my $query = $conn->prepare(`
	`  'SELECT * FROM user WHERE name = ? AND age = ?'`
	`);`
	`my $result = $query->execute('myname', 18);`

PHP	`$query = $conn->prepare(`
	`  'SELECT * FROM user WHERE name = ? AND age = ?');`
	`$result = $conn->execute($query, array('myname', 18));`

まとめ

JavaのJDBC，PerlのDBI，PHPのDBの各ライブラリによる，MySQLへのプログラミングインタフェースの利用方法とそれらの差異を見てきました．どのインタフェースも同じような思想のもとで設計されているためにとてもよく似たAPIになっています．RDBMSのクライアントを実装する場合，プログラミング上の手続きは言語や環境を問わずどれも同じなので，使用するメソッドやオブジェクトの差異にさえ注意すれば，言語や環境が変わっても同じように開発することができるはずです．WD