第4章　Servletからのデータベース操作

これまでのサーブレットでは通常のファイルに永続的なデータを記録していたが、実際には、データベースを利用すると便利な場合が多い。この章では、Java プログラムからデータベースを利用する方法を紹介する。

4.1　JDBC とは

JDBC とは **J**ava **D**ata**B**ase **C**onnectivity の略で、Java からデータベースを利用するための統一インタフェース（API）である。JDBC を利用することにより、利用するデータベースサーバーに依存せずにデータベースを利用する Java プログラムを作成することが可能になる。

Oracle, PostgreSQL, MySQL など、ほとんどのメジャーなデータベースサーバーは Java から JDBC 経由で利用可能である。

本講習では Servlet から利用するデータベースサーバーとして、OpenOffice.org（ver. 2.0 以降）でも採用されている HSQLDB を利用する。HSQLDB は 100% Java で記述された軽量でインストールが容易なデータベースサーバーである。なお、他のデータベースサーバーを利用する場合も、プログラム自体はほとんど変更する必要はない。

HSQLDB のインストール方法については別ドキュメントで説明する。また、SQL の基本的な書き方など関係データベースそのものに関する事柄は既知のものとして扱う。

4.2　JDBC の利用方法

Java のプログラムからデータベースサーバーを利用するためには、通常次のような手順を踏む必要がある。

1. JDBC ドライバーのロード
2. データベースサーバーへの接続を確立
3. SQL 文の送信
4. SQL 文実行結果の取出し
5. データベースサーバーへの接続を切断

次のサーブレットプログラム（DBSelect.java）でこれらを順に説明する。この DBSelect.java は customer というテーブルから`"SELECT * FROM customer"`とい

う SQL コマンドですべての内容を取り出し、それを単に HTML の表として整形するプログラムである。

まず、java.sql パッケージのいくつかのクラスを import していることに注意する。JDBC 関係のクラスは主にこのパッケージに定義されている。

ファイル DBSelect.java

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.sql.Statement;
8
9  import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12
13 public class DBSelect extends HttpServlet {
14   @Override
15   public void doGet(HttpServletRequest request,
16                     HttpServletResponse response)
17   throws IOException {
18
19     Connection conn = null;
20
21     response.setContentType("text/html;␣charset=Windows-31J");
22     PrintWriter out = response.getWriter();
23
24     out.println("<html><head></head><body>");
25     out.println("<table␣border='true'>");
26     out.println("<tr><th>id</th><th>first</th><th>last</th>"
27                 +"<th>street</th><th>city</th></tr>");
28
29     try {
30       // JDBC ドライバーのロード
31       Class.forName("org.hsqldb.jdbcDriver");
32       // HSQLDB 用の URL
33       String url = "jdbc:hsqldb:hsql://localhost";
34       // HSQLDB の既定値
35       String user = "sa", password="";
36       // 接続の確立
37       conn = DriverManager.getConnection(url, user, password);
38       Statement stmt = conn.createStatement();
39       // SQL の送信
40       ResultSet rs = stmt.executeQuery("SELECT␣*␣FROM␣customer");
41
42       while (rs.next()) {
43         out.print("<tr>");
```

```
44          out.print("<td>"+rs.getString("id")+"</td>");
45          out.print("<td>"+rs.getString("firstname") + "</td>");
46          out.print("<td>"+rs.getString("lastname") + "</td>");
47          out.print("<td>"+rs.getString("street") + "</td>");
48          out.print("<td>"+rs.getString("city") + "</td>");
49          out.print("</tr>");
50        }
51      } catch (ClassNotFoundException e) {
52        out.println("クラスが見つかりません。");
53      } catch (SQLException e) {
54        out.println("データベース操作中にエラーがありました。");
55        out.println("<pre>");
56        e.printStackTrace(out);
57        out.println("</pre>");
58      } finally {
59        try {
60          if (conn != null) { conn.close(); /* 接続の切断 */ }
61        } catch (SQLException e) {}
62      }
63      out.println("</table>");
64      out.println("</body></html>");
65      out.close();
66    }
67  }
```

4.2.1 JDBCドライバーのロード

JDBC を利用するプログラムでは、通常、JDBCドライバー（個々のデータベースサーバーにアクセスするための固有のライブラリ）を動的に（つまりプログラム実行時に）ロードする。クラスを動的にロードするためには Class.forName というクラスメソッドを使用する。

HSQLDB の場合、動的にロードすべきクラス名は"org.hsqldb.jdbcDriver"である[1]ので、

```
Class.forName("org.hsqldb.jdbcDriver");
```

という呼出しをする。これで、JDBCドライバーがロード・登録され、利用可能な状態になる。

このメソッドは与えられたクラス名に対応する class ファイルを見付けられなかった場合、ClassNotFoundException を発生する。そのため ClassNotFoundException を catch ブロックで処理している。

[1]他のデータベースサーバーを利用する場合のクラス名は、対応する JDBCドライバーのドキュメントを参照する必要がある。

4.2.2　接続の確立

次にデータベースサーバーとの通信路を確立する。このためには、DriverManager.getConnection というクラスメソッドを利用する。このメソッドの第 1 引数は、接続のための URL 表記で、データベースを利用するための JDBC ドライバーの種類とデータベースのある場所を示している。ローカルホスト（Servlet コンテナと同一のホスト）で動作している HSQLDB サーバーにアクセスする場合は、この URL は "jdbc:hsqldb:hsql://localhost"とする。第 2 引数はユーザー名、第 3 引数はパスワードで、HSQLDB の場合、既定値のユーザー名・パスワードはそれぞれ"sa"と ""である。データベースへの接続を表すメソッドの戻り値は java.sql.Connection 型であり、以降はこの Connection オブジェクトを利用してデータベースを操作する。

> **参考:** データベースへの接続は重い処理なので、サーブレットの場合、プーリングという技法を用いて、一度確立した接続を何度も再利用するのが普通である。この場合、サーブレットの設定ファイルに、JDBC ドライバーのクラス名や URL を記述する。本講習ではこの方法の説明は割愛する。

4.2.3　SQL 文の送信と結果の取出し

データベースサーバーに SQL 文を送信するためにまず、java.sql.Statement クラスのオブジェクトを用意する。この Statement オブジェクトは Connection オブジェクトの createStatement メソッドで生成される。次に Statement オブジェクトの executeQuery メソッドの引数に SQL の SELECT 文を文字列として渡し、データベースサーバーに送信する。データベースサーバーでこの SQL 文が実行され、結果が Java のオブジェクトに変換されて Java 側に返される。この executeQuery メソッドの戻り値は java.sql.ResultSet というクラスのオブジェクトである。

なお、これらのデータベース操作はエラーを起こす可能性がある。データベース操作のときのエラーは、ほとんどの場合 SQLException クラスに属するので、catch ブロックでこの例外を処理している。

ResultSet オブジェクトはクエリの結果の表に対応するデータを保持する。ResultSet オブジェクトの next メソッドは現在行を進める。初期状態では現在行は最初の行の前に位置付けられていて、next メソッドの最初の呼出しによって、最初の行が現在行になる。また、これ以上、行がない場合は、next メソッドは false を返す。この ResultSet オブジェクトの getString メソッドを、列名を引数として呼び出すと、現在行の当該の列の内容が String 型として取り出される。この例題の場合、HSQLDB のテストデータ中の customer というテーブルをアクセスする。このテーブルには、id, firstname, lastname, street, city という 5 つの列がある。ResultSet クラスには getString 以外にも getInt など他の型のデータを取り出す getter メソッドが多く存在する。またそれぞれの getter には列を名前 (文字列) ではなく列の

インデックス(整数)で指定するバージョンもある。詳しくは java.sql.ResultSet クラスのドキュメント (***(J2SEAPI)***/java/sql/ResultSet.html) を参照すること。

問 4.2.1 java.sql.PreparedStatement クラスの使用法を調べよ。

4.2.4 接続の切断

データベースを利用する場合、接続の切断は確実に行なう必要がある。(そうしないとネットワークその他の資源が無駄使いされてしまう可能性がある。) DBSelect.java では切断を確実に行なうために、データベースに関する操作をすべて **try** ブロックの中で行ない、それに対応する **finally** ブロック内で接続の切断 (conn.close()) を行なっている。

4.3 データベースの更新

次に示す例はデータベースの内容を更新するサーブレットの例である。ここでは前節で利用したものと同じ customer テーブルを用い、顧客の住所を変更するサーブレットを示す。

このサーブレットは次のようなフォームから利用することを想定している。

ファイル DBUpdate.html

```
1
2  <html><head></head>
3  <body>
4  <form action='DBUpdate' method='POST'>
5  更新する値を入力してください。<br>
6  <table border>
7  <tr><th>列名</th>
8      <th>値</th></tr>
9  <tr><td>ID</td>
10     <td><input type='text' size='5' name='id'></td></tr>
11 <tr><td>STREET</td>
12     <td><input type='text' size='32' name='street'></td></tr>
13 <tr><td>CITY</td>
14     <td><input type='text' size='32' name='city'></td></tr>
15 </table>
16 <input type='submit' value='送信'>
17 </form>
18 </body>
19 </html>
```

ファイル DBUpdate.java

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3  import java.sql.Connection;
4  import java.sql.DriverManager;
```

```
 5 import java.sql.SQLException;
 6 import java.sql.Statement;
 7
 8 import javax.servlet.http.HttpServlet;
 9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 public class DBUpdate extends HttpServlet {
13   @Override
14   public void doPost(HttpServletRequest request,
15                      HttpServletResponse response)
16               throws IOException {
17
18     Connection conn = null;
19
20     response.setContentType("text/html; charset=Windows-31J");
21     PrintWriter out = response.getWriter();
22
23     out.println("<html><head></head><body>");
24
25     String id     = request.getParameter("id");
26     String street = request.getParameter("street");
27     String city   = request.getParameter("city");
28
29     try {
30       String user = "sa", password="";
31
32       Class.forName("org.hsqldb.jdbcDriver");
33       String url = "jdbc:hsqldb:hsql://localhost";
34       conn = DriverManager.getConnection(url, user, password);
35       Statement stmt = conn.createStatement();
36       stmt.executeUpdate("UPDATE␣customer␣SET␣street␣=␣'"+street
37                          +"',␣city␣='"+city+"'␣WHERE␣id␣=␣"+id);
38
39       out.println("テーブルを更新しました。<br>");
40       out.print("id="+id+",␣street='"+street+"',␣")
41       out.println("city='"+city+"'");
42     } catch (ClassNotFoundException e) {
43       out.println("クラスが見つかりません。");
44     } catch (SQLException e) {
45       out.println("テーブルの更新に失敗しました。");
46       out.println("<pre>");
47       e.printStackTrace(out);
48       out.println("</pre>");
49     } finally {
50       try {
51         if (conn != null) { conn.close(); }
52       } catch (SQLException e) {}
53     }
```

```
54      out.println("</body></html>");
55      out.close();
56    }
57  }
```

単なるクエリではなく、データベースの内容を書き換える時には executeQuery メソッドの代わりに、executeUpdate メソッドを用いる。このプログラムでは SQL の UPDATE 文を使っているが、行を挿入する INSERT 文や削除する DELETE 文などを使う場合でも同様に executeUpdate メソッドを用いる。executeUpdate メソッドは実行した SQL が対象とした行数を戻り値とする (ただし DBUpdate.java ではこの戻り値は利用していない)。それまでのデータベースへ接続する部分などは、DBSelect.java とまったく変わらない。

このサーブレットはデータベースを操作したあと、「テーブルを更新しました」というメッセージと更新内容 (失敗したときは「テーブルの更新に失敗しました」というメッセージ) を表示する。

問 4.3.1 X 国では price が単価 15 以上のものに対して 10% の間接税がかかるという。HSQLDB のテストデータの中の product テーブルの price は税抜の単価を示すものとする。product テーブルの内容を X 国での間接税を加えた値段を表に成形して出力するサーブレットを作成せよ。
（当然、データベース中のデータは変更してはいけない。）

問 4.3.2 HSQLDB のテストデータの中の customer テーブルに対して、既存の顧客のデータを変更するだけでなく、新規の顧客を追加する／顧客を削除することも可能な入力用の HTML ページと処理用のサーブレットを作成せよ。

キーワード :

JDBC, Class.forName メソッド, DriverManager.getConnection メソッド, Connection クラス, createStatement メソッド, Statement クラス, executeQuery メソッド, executeUpdate メソッド, ResultSet クラス, next メソッド, getString メソッド,