スッキリわかるサーブレット&JSP 入門 第 13 章を MySQL を使ってやってみる

Seiichi Nukayama

2020-07-26

目次

1　JDBC ドライバーをインストールする

JDBC ドライバーを Windows にインストールするのは、ネットで調べてもちょっとわかりにくいかもしれない。情報が古いこともある。また、Oracle が Web サイトをけっこう頻繁に模様替えしてるのもある。

1.1　ダウンロード

まず、ここにいく。

https://dev.mysql.com/downloads/

“Connector/j” をクリックする。

https://dev.mysql.com/downloads/connector/j/ のページに遷移するので、ここのところからインストールするのだが、“Connector/J 8.0.21” と書かれているところの “Select Operating System...” には、Windows はない。

Windows は、その下の “MySQL Installer for Windows” をインストールして、それを使ってインストールすることになる。

大きなバナーをクリックするか、その下の “Windows (x86, 32 & 64-bit), MySQL Installer MSI” の横の “Go to Download Page >” をクリックする。

https://dev.mysql.com/downloads/connector/j/ というページが開き、少し下に “MySQL Installer 8.0.21” というコラムがある。“Generally Available (GA) Releases” とある。

その中に二つの Download ボタンがある。

上は、“Windows(x86, 32-bit), MSI Installer 8.0.21 24.5M” とある。ダウンロードファイルは、“mysql-installer-web-community-8.0.21.0.msi”

下は、“Windows(x86, 32-bit), MSI Installer 8.0.21 427.6M” とある。ダウンロードファイルは、“mysql-installer-community-8.0.21.0.msi”

結論から言うと、下の方をダウンロードしたほうがうまくいった。でも、どちらを選んでもできることは同じようである。

1.2　インストール

上記ファイルをクリックすると、“Login Now or Sign Up for a free account” とあって、ユーザー認証/登録を促される。別に登録してもかまわない。

ここでは、左下の “No thanks, just start my download” を選択する。

ダウンロードフォルダに “mysql-installer-community-8.0.21.0.msi” がダウンロードされる。このファイルをダブルクリックして **インストール** を始める。

「このアプリがデバイスに変更を加えることを許可しますか？」と聞かれるので、「はい」をクリックする。

“Choosing a Setup Type” のダイアログが開く。ここではインストールするものを選べる。通常は、“Developer Default” を選択するのだが、今回は、「JDBC ドライバー」のみ必要なので、一番下の “Custom” を選択して、“Next”。

“Select Products and Features” のダイアログが開く。“MySQL Connectors” の項目の「＋」をクリックする。

“Connector/J” − “Connector/J 8.0” − “Connector/J 8.0.21-X86” とクリックすると、右側の矢印が緑色に変わるので、その矢印をクリックする。

すると、右側に “Connector/J 8.0.16-X86” が表示される。“Next” をクリックする。

“Installation” のダイアログが開く。“Connector/J 8.0.21” が表示されており、“Ready to Install” となっている。下の “Execute” ボタンをクリックする。

“Complete” となる。 “Next” をクリックする。

“Installation Complete” となり、 “Copy Log to Clipboard” ボタンを押しておく。“Finish” ボタンをクリックして終了。

コピーしておいたログをメモ帳などで見てみると、どこにインストールされたかわかるので、のちにクラスパスを指定するときに役立つ。このログは “connector-j.log” とでも名前をつけて保存しておく。

1.3 クラスパスの指定

たとえば、以下のようにする。

```
> set CLASSPATH=.;C:\Program Files (x86)\MySQL\Connector J 8.0\mysql-connector-java-8.0.16.jar
```

こののち、コンパイル時に jar ファイルも読み込んでくれるようになる。

1.4 補足

MySQL Connector/J 8.0 については、以下のように書かれてある。

MySQL Connector/J 8.0 is highly recommended for use with MySQL Server 8.0, 5.7, 5.6, and 5.5. Please upgrade to MySQL Connector/J 8.0. (MySQL Connector / J 8.0 は、MySQL Server 8.0、5.7、5.6、および 5.5 で使用することを強くお勧めします。 MySQL Connector / J 8.0 にアップグレードしてください。by Google 翻訳)

MySQL5.7 などでも使えるようである。

2　データを準備する

p383 のコードを入力する前に、このコードで使っているデータを MySQL で準備しておかなくてはならない。

2.1　MySQL:ユーザーの作成

XAMPP などで、MySQL を動作させておく。

まず、MySQL で使用するユーザーを用意しなくてはならない。ROOT のままだと、他のデータベースにもアクセス可能なので、実際にはそのデータベースにのみアクセス権限が与えられているユーザーを作成することになる。しかし今はお試しでプログラムを作成しているので、ユーザー名とパスワードは簡単なものにしておく。

ユーザー名: sa
パスワード: sa

本 (p383) ではパスワードは設定されていないが、ユーザー名と同じにしておいたら忘れることはないだろう。

1. MySQL にルートでログインする。
   Windows のコマンドプロンプトで、以下のようにする。
   ```
   > mysql -u root -p
   Password: ****
   ```
   (多くの場合、パスワードは設定されていないか、もしくは root である)

2. ユーザーを作成する。
   ```
   mysql> create user 'sa'@'localhost' identified by 'sa';
   ```
   (ユーザー"sa" を作成し、パスワードを"sa" としている) [*1][*2]

3. そのユーザーにこれから作成するデータベースへの権限を与える。
   ```
   mysql> grant all privileges on example.* to 'sa'@'localhost';
   ```
   (データベース名は "example" で、それに関連する全てのファイルにアクセス権を与える)

4. `mysql> quit;` (MySQL をログアウトする)

以下のようにすると、ユーザーの作成とデータベースへの権限付与は同時に行うことができる。

```
mysql> grant all on example.* 'sa'@'localhost' identified by 'sa';
```

[*1] シングルクォーテーションは無くてもいけるみたい。ただし、パスワードにはシングルクォーテーションはあった方がよい。シングルクォーテーションはそれが文字列であることを明示している。

[*2] localhost の指定を無しにすることもできる。その場合、すべてのホストからそのデータベースに接続できることになる。多くの場合、あるホストで動くプログラムから接続するだけだし、そのプログラムと MySQL は同じホストで動作しているだろうから、ここは localhost という指定があったほうが、セキュリティ上好ましい。

2.2 MySQL の文字コードの確認

一応、MySQL の文字コードがどういう設定になっているかを確認しておく。インターネットでは UTF-8 を使うのだが、Windows では Shift-JIS(cp932) だからである。

以下は、XAMPP の場合である。XAMPP をインストールしたときの初期設定である。

```
mysql> show variables like '%char%'; \\
+--------------------------+---------------------------------+
| Variable_name            | Value                           |
+--------------------------+---------------------------------+
| character_set_client     | cp932                           |
| character_set_connection | cp932                           |
| character_set_database   | utf8mb4                         |
| character_set_filesystem | binary                          |
| character_set_results    | cp932                           |
| character_set_server     | utf8mb4                         |
| character_set_system     | utf8                            |
| character_sets_dir       | C:\xampp\mysql\share\charsets\ |
+--------------------------+---------------------------------+
8 rows in set (0.00 sec)
```

2.3 データベースの作成

先ほど作成したユーザーで MySQL にログインする。

```
> mysql -u sa -p
  Password: **  (sa)
```

データベース (example) を作成する。使用する文字コードも指定しておく。[*3]

```
mysql> create database example default character set=utf8;
```

テーブル (employee) を作成する。使用する文字コードも指定しておく。[*4]

```
mysql> create table employee (
    -> id char(6) not null primary key,
    -> name varchar(40),
    -> age int ) defalult character set=utf8;
```

[*3] データベース作成後に文字コードを設定する場合は、mysql>alter database example default character set=utf8; とする。

[*4] テーブル作成後に文字コードを設定する場合は、mysql>alter table employee default character set=utf8;

2.4 サンプルデータを入れる

サンプルデータを入れる。データベースアプリを作るときは、最初にサンプルデータを入れておくようにする。

```
mysql> insert into employee values ( 'EMP001', '湊 雄輔', 23);
mysql> insert into employee values ( 'EMP002', '綾部 みゆき', 22) ;
```

確認する。

```
mysql> select * from employee;
+--------+----------------+------+
| id     | name           | age  |
+--------+----------------+------+
| EMP001 | 湊 雄輔        |   23 |
| EMP002 | 綾部 みゆき    |   22 |
+--------+----------------+------+
2 rows in set (0.00 sec)
```

データがちゃんと入っている。[*5]

[*5] Windows では文字コードが Shift-JIS なので、漢字がうまく表示できない場合がある。そのときは、とりあえずローマ字など英字で入れておく。プログラムを実行したときに入力に漢字 (UTF-8) が使えればよい。

3　サンプルプログラム (p383) の作成

p383 に掲載されているサンプルプログラムの MySQL 版を書いてみる。

仮にこの本のプログラムコードを入れているフォルダを ”sukkiri” とする。この中に ”chap13-mysql” というフォルダを作成する。このフォルダの中に入り、”terminal” というフォルダを作成する。この中に更に ”src” と ”classes” と ”lib” というフォルダを作成する。

```
chap13-mysql/
└── terminal/
    ├── classes/
    ├── lib/
    └── src/
```

”chap13-mysql” フォルダには、この章で作成するサーブレット JSP ファイルを置く。

”terminal/src” フォルダには、p383 のコードを置く。

”terminal/classes” フォルダには、p383 のコードをコンパイルしてできた class ファイルを置く。

”terminal/lib” フォルダには、mysql-connector-java-8.0.21.jaq を置く。

”src” フォルダで以下のコードを書く。

Listing 1　src/SelectEmployeeSample.java

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.PreparedStatement;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6
7  public class SelectEmployeeSample {
8
9    static final String USERNAME = "sa";
10   static final String PASSWORD = "sa";
11   static final String CONNECT =
12               "jdbc:mysql://localhost:3306/example?serverTimezone=JST";
13
14   public static void main( String[] args ) {
15     // データベースに接続
16     try (Connection conn =
17        DriverManager.getConnection( CONNECT, USERNAME, PASSWORD )) {
18       // select 文
19       String sql = "select␣id,␣name,␣age␣from␣employee";
20       PreparedStatement pStmt = conn.prepareStatement( sql );
21
22       ResultSet rs = pStmt.executeQuery();
23
24       while( rs.next() ) {
25         String id = rs.getString("id");
26         String name = rs.getString("name");
27         int age = rs.getInt("age");
```

```
28
29          System.out.println("ID:" + id);
30          System.out.println("名前:" + name);
31          System.out.println("年齢:" + age + "\n");
32        }
33      } catch (SQLException e) {
34        e.printStackTrace();
35      }
36    }
37  }
```

“?serverTimezone=JST” は、これを書かないと、Timezone が設定されていないというエラーが出る場合がある (XAMPP の場合?)。

”terminal” フォルダにて、次の build.xml を書く。

Listing 2　build.xml

```
1  <?xml version="1.0" ?>
2  <project name="sample" default="compile" basedir=".">
3    <target name="compile">
4      <javac includeAntRuntime="false"
5             srcdir="./src"
6             destdir="./classes"
7             classpath="./lib/mysql-connector-java-8.0.21.jar"
8             />
9    </target>
10 </project>
```

コンパイルする。

```
> ant
compile:

BUILD SUCCESSFUL
```

実行する。

```
> cd classes
> java -cp .;../lib/mysql-connector-java-8.0.20.jar SelectEmployeeSample
```

-cp オプションで、現在のフォルダ「.」と「lib」フォルダの jar ファイルを指定している。
mysql-connector-java-8.0.21.jar は、コンパイル時にも、実行時にも、-classpath 指定が必要である。

```
ID:EMP001
名前:湊　雄輔
年齢:23

ID:EMP002
名前:綾部　みゆき
年齢:22
```

しかし、毎回このオプションを入力するのはとても邪魔くさいので、バッチファイルを作成して、ちょっとでも楽をする。

class フォルダにて、“run.bat”というファイルを作成し、内容を以下とする。

```
1 java -cp .;../lib/mysql-connector-java-8.0.21.jar %1
```

そして、class フォルダにてコマンドプロンプトを起動し、以下のようにする。

```
>run SelectEmployeeSample
```

4　DAO パターンのサンプルプログラム

本 p390 からのサンプルプログラムを書いてみる。

4.1　プロジェクトフォルダの設定

先ほど作った “chap13-mysql” フォルダの中に “dao” というフォルダを作成する。
そして、以下のようにフォルダを作成する。

```
dao
├── build.xml
├── classes/
├── lib/
│   └── mysql-connector-java-8.0.21.jar
└── src/
    ├── SelectEmployeeSample.java
    ├── dao/
    │   └── EmployeeDAO.java
    └── model/
        └── Employee.java
```

“build.xml”、“SelectEmployeeSample.java”、“EmployeeDAO.java”、“Employee.java” は、これから作成するファイルである。

“mysql-connector-java-8.0.21.java” は、先ほどダウンロードしたファイルである。

4.2　プログラムコードを書く

Listing 3　Employee.java

```
1  package model;
2
3  public class Employee {
4    private String id;
5    private String name;
6    private int age;
7
8    public Employee( String id, String name, int age ) {
9      this.id = id;
10     this.name = name;
11     this.age = age;
12   }
13
14   public String getId() { return id; }
15   public String getName() { return name; }
16   public int getAge() { return age; }
17 }
```

Listing 4　EmployeeDAO.java

```
1  package dao;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.util.ArrayList;
9  import java.util.List;
10
11 import model.Employee;
12
13 public class EmployeeDAO {
14     private final String JDBC_URL =
15         "jdbc:mysql://localhost:3306/example?serverTimezone=JST";
16     private final String DB_USER = "sa";
17     private final String DB_PASS = "sa";
18
19     /**
20      * 一覧表示メソッド
21      * @Param: none
22      * @Return:
23      * List<Employee> empList -- Employee型のリスト
24      */
25     public List<Employee> findAll () {
26         List<Employee> empList = new ArrayList <> ();
27
28         try (Connection conn =
29              DriverManager.getConnection( JDBC_URL, DB_USER, DB_PASS )) {
30
31             String sql = "select␣id,␣name,␣age␣from␣employee";
32             PreparedStatement pStmt = conn.prepareStatement( sql );
33
34             ResultSet rs = pStmt.executeQuery();
35
36             while (rs.next()) {
37                 String id = rs.getString("id");
38                 String name = rs.getString("name");
39                 int age = rs.getInt("age");
40                 Employee employee = new Employee( id, name, age );
41                 empList.add( employee );
42             }
43         } catch (SQLException e) {
44             e.printStackTrace();
45             return null;
46         }
47         return empList;
48     }
49
50     /**
51      * 削除処理
52      * @Param:
53      * String id -- ex.EMP001
54      * @Return:
```

```
55      * boolean -- true: 削除成功、 false: 削除失敗l
56      */
57     public boolean remove ( String id ) {
58         try (Connection conn =
59              DriverManager.getConnection( JDBC_URL, DB_USER, DB_PASS )) {
60 
61             String sql = "delete␣from␣employee␣where␣id␣=␣?";
62             PreparedStatement pStmt = conn.prepareStatement( sql );
63             pStmt.setString(1, id);
64             int result = pStmt.executeUpdate();
65             if (result < 1) {
66                 return false;
67             }
68         } catch (SQLException e) {
69             e.printStackTrace();
70             return false;
71         }
72         return true;
73     }
74 }
```

削除処理も入れてみた。

Listing 5　SelectEmployeeSample.java

```
1  import java.util.List;
2  import model.Employee;
3  import dao.EmployeeDAO;
4 
5  public class SelectEmployeeSample {
6      public static void main( String[] args ) {
7 
8          EmployeeDAO empDAO = new EmployeeDAO();
9          List<Employee> empList = empDAO.findAll();
10 
11         // 一覧
12         for (Employee emp : empList) {
13             System.out.println("ID:" + emp.getId());
14             System.out.println("名前:" + emp.getName());
15             System.out.println("年齢:" + String.valueOf(emp.getAge()) + "\n");
16         }
17 
18         // 削除
19         String id = "EMP003";
20         if (empDAO.remove(id)) {
21             System.out.println(id + "を削除しました。");
22         }
23     }
24 }
```

プロジェクト dao の先頭でコンパイルをおこなう。そして classes フォルダに移動する。

```
chap13-mysql\dao> ant
chap13-mysql\dao> cd classes
```

サンプルデータをひとつ増やしておく。

```
> mysql -u sa -p example
Password: **    (sa)

mysql> insert into employee values ('EMP003', '猿飛佐助', 25);
mysql> select * from employee;
  ( データの表示 )
```

ここで実行する。

```
classes> java -cp .;../lib/mysql-connector-java-8.0.21.jar SelectEmployeeSample
```

3 つのデータが表示されたあと、“EMP003 を削除しました。“ と表示されたら成功である。
前回で作成した “run.bat” をここにコピーして使うと楽ができる。

Listing 6　run.bat

```
1 java -cp .;../lib/mysql-connector-java-8.0.21.jar %1
```

```
> run SelectEmployeeSample
```

5　どこつぶでデータベースを利用する

5.1　サンプルデータの作成

MySQL でプログラムで使うためのサンプルデータを作る。

前回までで “sa” というユーザーを作成しているので、そのユーザーに、これから作成するデータベースについての権限を与えておく。

```
> mysql -u root -p
Password:       (なし)

mysql> grant all privileges on docoTsubu.* to 'sa'@'localhost';
mysql> quit;
```

次に新しいデータベース docoTsubu を作成する。

```
> mysql -u sa -p
Password: **      (sa)

mysql> create database docoTsubu default character set=utf8;
mysql> use docoTsubu;
mysql> create table mutter (
    -> id int primary key auto_increment,
    -> name varchar(100) not null,
    -> text varchar(255) not null
    -> ) default character set=utf8;
mysql> insert into mutter ( name, text ) values ( '湊', '今日は休みだ' ) ;
mysql> insert into mutter ( name, text ) values ( '綾部', 'いいな?' ) ;
mysql> select * from mutter;

+----+------+--------------+
| id | name | text         |
+----+------+--------------+
|  1 | 湊   | 今日は休みだ |
|  2 | 綾部 | いいな～     |
+----+------+--------------+
2 rows in set (0.00 sec)
```

insert 文でデータを追加する時、id は auto_increment (自動追加) なので指定しなくてよい。

5.2 プログラムの作成

フォルダの構成は以下のとおり。

```
docoTsubuDao/
├── WEB-INF/
│   ├── classes/
│   └── jsp/
└── src/
    ├── dao/
    ├── model/
    └── servlet/
```

”sukkiri” フォルダの中に ”docoTsubuDao” フォルダを作成し、その中に上記のフォルダを作成する。
p399 からのコードを入力していく。
これは本のコードのままである。

Listing 7　model/Mutter.java

```
1  package model;
2
3  import java.io.Serializable;
4
5  public class Mutter implements Serializable {
6    private int id;
7    private String userName;
8    private String text;
9
10   public Mutter () {}
11
12   public Mutter (String userName, String text) {
13     this.userName = userName;
14     this.text = text;
15   }
16
17   public Mutter (int id, String userName, String text) {
18     this.id = id;
19     this.userName = userName;
20     this.text = text;
21   }
22
23   public int getId () { return id; }
24   public String getUserName () { return userName; }
25   public STring getText () { return text; }
26 }
```

次のコードは、jdbs ドライバを読み込んでいる部分が違うだけで、あとは同じ。
ただ、プログラムがちゃんと動いているかをチェックするためのコードを入れておいた。

Listing 8　dao/MutterDAO.java

```java
package dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import model.Mutter;

public class MutterDAO {
  // info about database
  private final String JDBC_URL =
      "jdbc:mysql://localhost:3306/docoTsubu?serverTimezone=JST";
  private final String DB_USER = "sa";
  private final String DB_PASS = "sa";

  public List<Mutter> findAll () {
    List<Mutter> mutterList = new ArrayList <> ();

    // connect
    try (Connection conn =
        DriverManager.getConnection( JDBC_URL, DB_USER, DB_PASS )) {

      String sql =
          "select id, name, text from mutter order by id desc";
      PreparedStatement pStmt = conn.prepareStatement( sql );

      ResultSet rs = pStmt.executeQuery();

      while (rs.next()) {
        int id = rs.getInt("id");
        String userName = rs.getString("name");
        String text = rs.getString("text");
        System.out.println("CHECK: " + userName); // チェック用
        Mutter mutter = new Mutter( id, userName, text );
        mutterList.add(mutter);
      }
    } catch (SQLException e) {
      e.printStackTrace();
      System.out.println("エラーでっせ！");
      return null;
    }
    return mutterList;
  }

  public boolean create( Mutter mutter ) {
    try (Connection conn =
        DriverManager.getConnection(
              JDBC_URL, DB_USER, DB_PASS)) {
      String sql = "insert into mutter( name, text) values (?,?)";
      PreparedStatement pStmt = conn.prepareStatement( sql );
      pStmt.setString( 1, mutter.getUserName() );
      pStmt.setString( 2, mutter.getText() );
```

```
56
57        // resultには追加された行数が入る
58        int result = pStmt.executeUpdate();
59        if (result != 1) {
60          return false;
61        }
62      } catch (SQLException e) {
63          e.printStackTrace();
64          System.out.println("データの追加失敗");
65        return false;
66      }
67      return true;
68    }
69  }
```

Listing 9　servlet/Main.java

```
1   package servlet;
2
3   import java.io.IOException;
4   import java.util.ArrayList;
5   import java.util.List;
6
7   import javax.servlet.RequestDispatcher;
8   import javax.servlet.ServletContext;
9   import javax.servlet.ServletException;
10  import javax.servlet.annotation.WebServlet;
11  import javax.servlet.http.HttpServlet;
12  import javax.servlet.http.HttpServletRequest;
13  import javax.servlet.http.HttpServletResponse;
14  import javax.servlet.http.HttpSession;
15
16  import model.GetMutterListLogic;
17  import model.Mutter;
18  import model.PostMutterLogic;
19  import model.User;
20
21  @WebServlet("/Main")
22  public class Main extends HttpServlet {
23    private static final long serialVersionUID = 1L;
24
25    protected void doGet (HttpServletRequest request,
26                          HttpServletResponse response)
27      throws ServletException, IOException {
28
29      // つぶやきリストを取得して、リクエストスコープに保存
30      GetMutterListLogic getMutterListLogic =
31          new GetMutterListLogic();
32      List<Mutter> mutterList = getMutterListLogic.execute();
33      request.setAttribute("mutterList", mutterList);
34      if (mutterList.size() > 0) {
35          System.out.println("空ではない!");
36      }
37
38      // セッションスコープからユーザー情報(loginUser)を取得
39      HttpSession session = request.getSession();
```

```
40      User loginUser = (User) session.getAttribute("loginUser");
41
42      if (loginUser == null) {
43          // ユーザー情報が空なら新規作成画面へリダイレクト
44          response.sendRedirect("/docoTsubuDao");
45      } else {
46          RequestDispatcher dispatcher =
47              request.getRequestDispatcher("/WEB-INF/jsp/main.jsp");
48          dispatcher.forward( request, response );
49      }
50    }
51
52    @SuppressWarnings("unchecked")
53    public static List<Mutter> automaticCast( Object src ) {
54        List<Mutter> castedObject = (List<Mutter>) src;
55        return castedObject;
56    }
57
58    protected void doPost (HttpServletRequest request,
59                           HttpServletResponse response)
60      throws ServletException, IOException {
61
62      request.setCharacterEncoding("UTF-8");
63      String text = request.getParameter("text");
64
65      // check
66      if (text != null && text.length() != 0) {
67
68        // User info
69        HttpSession session = request.getSession();
70        User loginUser = (User) session.getAttribute("loginUser");
71
72        // added Tsubuyaki List
73        Mutter mutter = new Mutter( loginUser.getName(), text );
74        PostMutterLogic postMutterLogic = new PostMutterLogic();
75        postMutterLogic.execute(mutter);
76
77      } else {
78        // textが 空だった場合
79        request.setAttribute("errorMsg", "つぶやきが入力されていません");
80      }
81
82      GetMutterListLogic getMutterListLogic =
83          new GetMutterListLogic ();
84      List<Mutter> mutterList = getMutterListLogic.execute();
85      request.setAttribute("mutterList", mutterList);
86
87      // forward to main
88      RequestDispatcher dispatcher =
89          request.getRequestDispatcher("/WEB-INF/jsp/main.jsp");
90      dispatcher.forward( request, response );
91    }
92  }
```

Listing 10　model/GetMutterListLogic.java

```
 package model;

import java.util.List;
import dao.MutterDAO;

public class GetMutterListLogic {
	public List<Mutter> execute () {
		MutterDAO dao = new MutterDAO ();
		List<Mutter> mutterList = dao.findAll ();
		return mutterList;
	}
}
```

以下は、第 11 章で作成したものをそのまま使う。

Listing 11　model/LoginLogic.java

```
 package model;

public class LoginLogic {
    public boolean execute (User user) {
        if (user.getPass().equals("1234")) { return true; }
        return false;
    }
}
```

Listing 12　model/PostMutterLogic.java

```
 package model;

import dao.MutterDAO;

public class PostMutterLogic {
    public void execute (Mutter mutter) {
        MutterDAO dao = new MutterDAO();
        dao.create( mutter );
    }
}
```

Listing 13　model/User.java

```
 package model;

import java.io.Serializable;

public class User implements Serializable {
    private String name;
    private String pass;

    public User () {}
    public User (String name, String pass) {
        this.name = name;
        this.pass = pass;
    }

```

```
15      public String getName () { return name; }
16      public String getPass () { return pass; }
17  }
```

Listing 14　servlet/Login.java

```
1   package servlet;
2
3   import java.io.IOException;
4
5   import javax.servlet.RequestDispatcher;
6   import javax.servlet.ServletException;
7   import javax.servlet.annotation.WebServlet;
8   import javax.servlet.http.HttpServlet;
9   import javax.servlet.http.HttpServletRequest;
10  import javax.servlet.http.HttpServletResponse;
11  import javax.servlet.http.HttpSession;
12
13  import model.LoginLogic;
14  import model.User;
15
16  @WebServlet("/Login")
17  public class Login extends HttpServlet {
18      private static final long serialVersionUID = 1L;
19
20      protected void doPost (HttpServletRequest request,
21                             HttpServletResponse response)
22          throws ServletException, IOException {
23
24          request.setCharacterEncoding("UTF-8");
25          String name = request.getParameter("name");
26          String pass = request.getParameter("pass");
27
28          User user = new User (name, pass);
29
30          /**
31           * ログイン処理
32           * LoginLogic.execute
33           * user.getPass() == 初期パスワード ==> true
34           * .. != .. ==> false
35           */
36          LoginLogic loginLogic = new LoginLogic();
37          boolean isLogin = loginLogic.execute (user);
38
39          if (isLogin) {
40              HttpSession session = request.getSession();
41              session.setAttribute("loginUser", user);
42          }
43
44          RequestDispatcher dispatcher =
45              request.getRequestDispatcher("/WEB-INF/jsp/loginResult.jsp");
46          dispatcher.forward( request, response );
47      }
48  }
```

Listing 15　servlet/Logout.java

```
 package servlet;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/Logout")
public class Logout extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet( HttpServletRequest request,
                          HttpServletResponse response )
        throws ServletException, IOException {

        HttpSession session = request.getSession();
        // セッションスコープの破棄
        session.invalidate();

        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/WEB-INF/jsp/logout.jsp");
        dispatcher.forward( request, response );
    }
}
```

Listing 16　common.jsp

```
 <%@ page language="java" contentType="text/html;␣charset=UTF-8"
        pageEncoding="UTF-8" %>
<%@ page import="java.util.Date,␣java.text.SimpleDateFormat" %>
<% String name = "猿飛佐助"; %>
```

Listing 17　footer.jsp

```
 <%@ page language="java" contentType="text/html;␣charset=UTF-8"
        pageEncoding="UTF-8" %>
<p>Copyright &copy; どこつぶ制作委員会 All Rights Reserved.</p>
```

Listing 18　index.jsp

```
 <%@ page language="java" contentType="text/html;␣charset=UTF-8"
        pageEncoding="UTF-8" %>

<%@ include file="/common.jsp" %>
<%
Date date = new Date();
SimpleDateFormat sdf = new SimpleDateFormat("YYYY年MM月dd日");
String today = sdf.format(date);
%>
```

```
<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>どこつぶ</title>
  </head>
  <body>
    <h1>どこつぶへようこそ</h1>
    <time><%= today %></time>
    <p>管理人:<%= name %></p>
    <form action="/docoTsubuDao/Login" method="post">
      ユーザー名:<input type="text" name="name"/><br/>
      パスワード:<input type="password" name="pass"/><br/>
      <input type="submit" value="ログイン"/>
    </form>
    <jsp:include page="/footer.jsp" />
  </body>
</html>
```

5.3 ビルド

build.xml を以下のようにする。

```
 <?xml version="1.0" ?>
<project name="docoTsubu_dao" default="compile" basedir=".">
  <target name="compile">
      <javac includeAntRuntime="false"
            encoding="UTF-8"
            srcdir="./src"
            destdir="./WEB-INF/classes"
            classpath="C:\pleiades\tomcat\9\lib\servlet-api.jar;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣C:\pleiades\tomcat\9\lib\jsp-api.jar;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣C:\pleiades\tomcat\9\lib\mysql-connector-java-8.0.21.jar"
            />
  </target>
</project>
```

C:/pleiades/tomcat/9/conf/Catalina/localhost のフォルダに以下のように “docoTsubuDao.xml” を作る。

Listing 19　docoTsubuDao.xml

```
<?xml version='1.0' encoding='utf-8'?>
<Context path="/docoTsubuDao"
   docBase="C:\Users\user\sukkiri\chap13-mysql\docoTsubuDao" />
```