

DAO & Bean を使ったテスト

1 ・テストアプリ（testDaoBean）の構成

<pre> □/opt/tomcat └─□webapps └─■testDaoBean └─■META-INF └─└─◆context.xml └─■WEB-INF └─■lib └─└─◇mysql-connector-java-8.0.22.jar └─■src └─└─■sample └─└─└─└─◆All.java ---① └─└─└─└─◆ProductList.java ---② └─└─└─└─└─◆ProductList2.java ---③ └─└─■dao └─└─└─└─◆DAO.java ---②③ └─└─└─└─└─◆ProductDAO.java ---③ └─└─└─└─■bean └─└─└─└─└─└─◆Product.java ---③ └─└─■classes └─└─└─└─※ </pre>	<p>/opt/tomcat 配下に、 左図の構成で testDaoBean を配置する</p> <p>□■ ディレクトリ ◇◆ ファイル</p> <p>■◆部を新規で作成</p> <p>jar ファイルは lib 配下に移動させる</p> <p>※classes ディレクトリには java ファイルをコンパイルして 同じ構成で配置していく</p> <p>① サーブレットのみ ② DAO を使う ③ DAO・Bean を使う の3パターンを作っていく</p>
--	---

2 ・テスト用データベース（test_tool）の作成

MySQL にログインし、データベースを作成	
<pre>mysql> CREATE DATABASE test_tool; USE test_tool; Query OK, 1 row affected (0.02 sec)</pre>	CREATE DATABASE test_tool;
<pre>mysql> USE test_tool; Database changed</pre>	USE test_tool;
<pre>mysql> CREATE TABLE product(-> id INT PRIMARY KEY AUTO_INCREMENT, -> name VARCHAR(20),price INT); Query OK, 0 rows affected (0.03 sec)</pre>	CREATE TABLE product(id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(20),price INT);
<pre>mysql> INSERT INTO product(name,price) VALUES -> ('tool1','100'),('tool2','150'), -> ('tool3','200'),('tool4','250'); Query OK, 4 rows affected (0.01 sec) Records: 4 Duplicates: 0 Warnings: 0</pre>	INSERT INTO product(name,price) VALUES ('tool1','100'),('tool2','150'), ('tool3','200'),('tool4','250');

mysql> SELECT * FROM product; +-----+ id name price +-----+ 1 tool1 100 2 tool2 150 3 tool3 200 4 tool4 250 +-----+ 4 rows in set (0.00 sec)	SELECT * FROM product;
mysql> QUIT Bye	QUIT

3 ・ Apache の設定

「 vi /etc/httpd/conf/httpd.conf 」で、末尾に以下のコードを追加

```
ProxyPass /TDB/ ajp://localhost:8009/testDaoBean/
ProxyPassReverse /TDB/ ajp://localhost:8009/testDaoBean/
ProxyPass /TDB/ ajp://localhost:8009/testDaoBean/
ProxyPassReverse /TDB/ ajp://localhost:8009/testDaoBean/
-- INSERT --
```

4 ・ ディレクトリの作成

```
cd /opt/tomcat/webapps;¥
mkdir testDaoBean; cd testDaoBean;¥
mkdir META-INF WEB-INF; cd WEB-INF;¥
mkdir lib src classes; cd src;¥
mkdir sample dao bean; ls ../../; ls ../; ls
```

左記のコマンドで
全てのディレクトリを作成/確認まで

```
[root@localhost ~]# cd /opt/tomcat/webapps;¥
> mkdir testDaoBean; cd testDaoBean;¥
> mkdir META-INF WEB-INF; cd WEB-INF;¥
> mkdir lib src classes; cd src;¥
> mkdir sample dao bean; ls ../../; ls ../; ls
META-INF WEB-INF
classes lib src
bean dao sample
[root@localhost src]#
```

5 ・ context.xml の作成

```
vi /opt/tomcat/webapps/testDaoBean/META-INF/context.xml
```

```
[root@localhost src]# vi /opt/tomcat/webapps/testDaoBean/META-INF/context.xml
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<Context>
    <Resource name = "jdbc/test_tool"
        auth = "Container"
        type = "javax.sql.DataSource"
        driverClassName = "com.mysql.jdbc.Driver"
        url          = "jdbc:mysql://localhost/test_tool"
        username = "root"
        password = "AT8_MySQL">

    </Resource>
</Context>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<Context>
    <Resource name = "jdbc/test_tool"
        auth = "Container"
        type = "javax.sql.DataSource"
        driverClassName = "com.mysql.jdbc.Driver"
        url          = "jdbc:mysql://localhost/test_tool"
        username = "root"
        password = "AT8_MySQL">

    </Resource>
</Context>
-- INSERT --
```

```
ls /opt/tomcat/webapps/testDaoBean/META-INF
```

```
[root@localhost src]# ls /opt/tomcat/webapps/testDaoBean/META-INF
context.xml
```

6 ・ lib ディレクトリに JDBC の jar ファイルを配置

```
cp /opt/tomcat/lib/¥
mysql-connector-java-8.0.22.jar ../lib; ls ../lib
```

コピー元を絶対パスで指定
[¥]長いので改行 「ls」で確認

```
[root@localhost src]# cp /opt/tomcat/lib/¥
> mysql-connector-java-8.0.22.jar ../lib; ls ../lib
mysql-connector-java-8.0.22.jar
```

```
package sample;

import java.io.*;
import java.sql.*;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import javax.sql.*;

/**--- Servlet implementation class All ---**/
@WebServlet("/All")
public class All extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        PrintWriter out = response.getWriter();

        try {
            // コネクション取得
            InitialContext ic = new InitialContext();
            DataSource ds = (DataSource) ic.lookup("java:/comp/env/jdbc/test_tool");
            Connection con = ds.getConnection();

            // SQL 文送信
            PreparedStatement st = con.prepareStatement("select * from product");
            // 実行&結果受け取り
            ResultSet rs = st.executeQuery();

            // データの表示
            while (rs.next()) {
                out.println(
                    rs.getInt("id") + " : " +
                    rs.getString("name") + " : ¥¥" +
                    rs.getInt("price")
                );
            }
            // データベース切断
            st.close();
            con.close();

        } catch (Exception e) {
            // 接続・SQL 文エラー
            e.printStackTrace(out);
        } // try
    }
}
```

8・Servlet をコンパイルして classes ディレクトリへ配置

```
javac -classpath ¥  
/opt/tomcat/lib/servlet-api.jar ¥  
sample/All.java -d ../classes; ¥  
ls ../classes; ls ../classes/sample
```

servlet-api のパスを指定してコンパイル
「-d」で保存先を指定
「ls」で確認

```
[root@localhost src]# javac -classpath ¥  
> /opt/tomcat/lib/servlet-api.jar ¥  
> sample/All.java -d ../classes; ¥  
> ls ../classes; ls ../classes/sample  
sample  
All.class
```

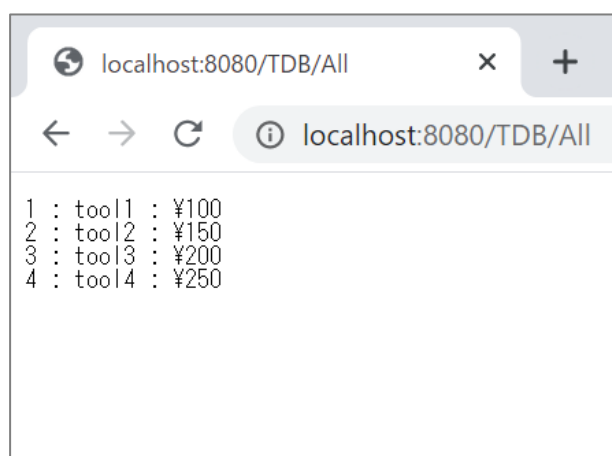
自動的に sample ディレクトリが作成され、その配下に class ファイルが配置されている

9・再起動させて読み込ませる

```
「systemctl restart tomcat httpd」
```

```
[root@localhost src]# systemctl restart tomcat httpd
```

10・ブラウザで①を実行



「http://localhost:8080/TDB/All」

にアクセスし、
データベースのデータが表示されれば OK

11・② DAO を使う

All.java の内容をライブラリ化する

- 1) DAO.java を作成 (コネクション取得)
- 2) ProductList.java を作成 (DAO を所得&処理&表示)

12・②-1) DAO.java を作成する (コネクション取得)

```
vi dao/DAO.java; ls dao
```

```
package dao;

import java.sql.Connection;
import javax.naming.InitialContext;
import javax.sql.DataSource;

public class DAO {
    static DataSource ds;

    public Connection getConnection() throws Exception {

        // 初期コンテキスト構築
        InitialContext ic = new InitialContext();

        // DB の接続先情報を取得 (context.xml の内容)
        ds = (DataSource) ic.lookup("java:/comp/env/jdbc/test_tool");

        return ds.getConnection();
    }
}
```

```
[root@localhost src]# vi dao/DAO.java; ls dao
DAO.java
```

13・②-2) ProductList.java を作成 (DAO を所得 & 処理 & 表示)

```
vi sample/ProductList.java; ls sample
```

```
package sample;

import dao.DAO;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/List")
public class ProductList extends HttpServlet {

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

        PrintWriter out = response.getWriter();

        try {
            // DB 接続
```

```

        DAO dao = new DAO();
        Connection con = dao.getConnection();

        // SQL 作成
        PreparedStatement st = con.prepareStatement("SELECT * FROM product");

        // SQL 実行
        ResultSet rs = st.executeQuery();

        // データをセット
        out.println("Use DAO");
        while (rs.next()) {
            out.println(
                rs.getInt("id")      + " : " +
                rs.getString("name") + " : ¥¥" +
                rs.getInt("price")
            );
        }
        // 接続解除
        st.close();
        con.close();
    } catch (Exception e) {
        e.printStackTrace(out);
    }
}
}

```

```

[root@localhost src]# vi sample/ProductList.java; ls sample
All.java ProductList.java

```

14・コンパイルして classes ディレクトリへ配置

```

javac -classpath ¥
/opt/tomcat/lib/servlet-api.jar ¥
*/*.java -d ../classes; ls ../classes/*

```

「*」全てを指定、なので
直下の全てのディレクトリの
全ての java ファイルをコンパイルする

```

[root@localhost src]# javac -classpath ¥
> /opt/tomcat/lib/servlet-api.jar ¥
> */*.java -d ../classes; ls ../classes/*
../classes/dao:
DAO.class

../classes/sample:
All.class ProductList.class

```

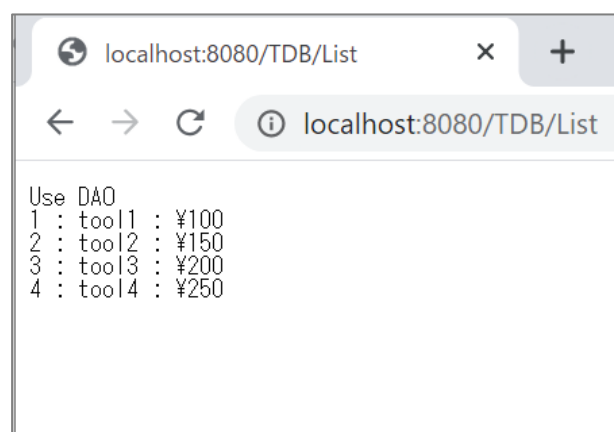
Classes ディレクトリ内に、新たに dao ディレクトリも自動生成されている

15・再起動させて読み込ませる

```
「systemctl restart tomcat」
```

```
[root@localhost src]# systemctl restart tomcat
```

16・ブラウザで②を実行



「http://localhost:8080/TDB/List」

「Use DAO」と一緒に
データベースのデータが表示されれば OK

17・③ DAO と Bean を使う

- 1) DAO. java はそのまま使用 (コネクション取得)
- 2) Product. java を作成する (Bean)
- 3) ProductDAO. java を作成する (DAO 取得 & 処理)
- 4) ProductList2. java を作成する (ProductDAO 取得 & 表示)

18・③-2) Product.java を作成 (Bean)

```
vi bean/Product. java; ls bean
```

```
package bean;
```

```
public class Product implements java.io.Serializable {  
    private int id;  
    private String name;  
    private int price;  
  
    public int getId() {  
        return id;  
    }  
}
```



```

    }
    public String getName() {
        return name;
    }
    public int getPrice() {
        return price;
    }
    public void setId(int id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setPrice(int price) {
        this.price = price;
    }
}

```

```

[root@localhost src]# vi bean/Product.java; ls bean
Product.java

```

19・③-3) ProductDAO.java を作成する (DAO 取得 & 処理)

```
vi dao/ProductDAO.java; ls dao
```

```

package dao;

import java.sql.*;
import java.util.*;
import bean.Product;

public class ProductDAO extends DAO {
    public List<Product> listAll() throws Exception {

        List<Product> list = new ArrayList<>();

        // DB 接続
        Connection con = getConnection();

        // SQL 作成
        PreparedStatement st = con.prepareStatement("SELECT * FROM product");

        // SQL 実行
        ResultSet rs = st.executeQuery();

        // データをセット
        while (rs.next()) {
            Product p = new Product();

```

```

        p.setId(rs.getInt("id"));
        p.setName(rs.getString("name"));
        p.setPrice(rs.getInt("price"));

        list.add(p);
    }

    // 接続解除
    st.close();
    con.close();

    return list;
}
}

```

```

[root@localhost src]# vi dao/ProductDAO.java; ls dao
DAO.java ProductDAO.java

```

20・③-4) ProductList2.java を作成する (ProductDAO 取得 & 表示)

```

vi sample/ProductList2.java; ls sample

```

```

package sample;

import java.io.*;
import java.sql.*;
import java.util.List;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import javax.sql.*;
import bean.Product;
import dao.ProductDAO;

@WebServlet("/List2")
public class ProductList2 extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        PrintWriter out = response.getWriter();

        try {
            ProductDAO dao = new ProductDAO();
            List<Product> list = dao.listAll();

            out.println("Use DAO & Bean");
            for (Product p : list) {
                out.println(

```

```

        p.getId()      + " : " +
        p.getName()    + " : ¥¥" +
        p.getPrice()

    );
}
} catch (Exception e) {
    e.printStackTrace(out);
}
}
}

```

```
[root@localhost src]# vi sample/ProductList2.java; ls sample
All.java ProductList.java ProductList2.java
```

21・コンパイルして classes ディレクトリへ配置

```

javac -classpath ¥
/opt/tomcat/lib/servlet-api.jar ¥
*/*.java -d ../classes; ls ../classes/*

```

前述と同じコード

複数ファイルを一つ一つ指定しなくても済む

```

[root@localhost src]# javac -classpath ¥
> /opt/tomcat/lib/servlet-api.jar ¥
> */*.java -d ../classes; ls ../classes/*
../classes/bean:
Product.class

../classes/dao:
DAO.class ProductDAO.class

../classes/sample:
All.class ProductList.class ProductList2.class

```

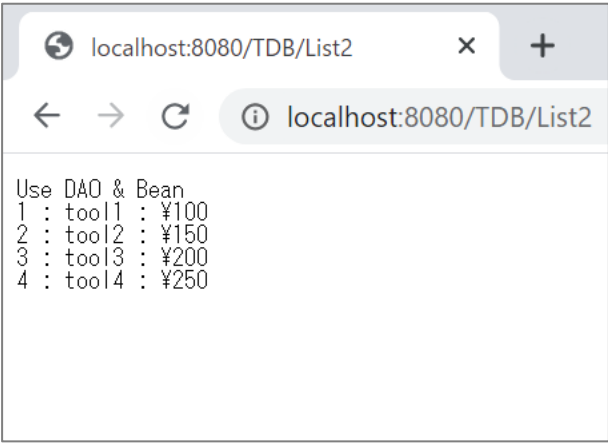
新たに bean ディレクトリも自動生成され、3つの java ファイルが作成されている

22・再起動させて読み込ませる

```
「systemctl restart tomcat」
```

```
[root@localhost src]# systemctl restart tomcat
```

23・ブラウザで③を実行

	<p>「http://localhost:8080/TDB/List2」</p> <p>「Use DAO & Bean」と一緒に データベースのデータが表示されれば OK</p>
---	---

Creation Date 2020 12