Tutorial 5 J2EE Java-beans MVC

[Experimental Objective]

1. Learn MVC framework and help understanding Spring and Hibernate framework.
2. Try to understand the inner logic between files.
3. Understand the interaction between jsp and servlet.
4. Learning to Use Intellij idea as Java IDE.
5. Accomplish some new needs based on the given code.

[Environment Setting]

1. Install JDK:

Download address:

https://www.oracle.com/technetwork/java/javase/overview/index.html

2. Download Tomcat:

Download address: http://tomcat.apache.org/download-90.cgi

3. Install mysql (you can install mysqlworkbench if you need) or other database

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

4. Install Intellij idea (Ultimate)

Download address: https://www.jetbrains.com/idea/download/#section=mac

Free student pack: https://www.jetbrains.com/buy/classroom/?fromMenu

You can also use eclipse EE instead, but you need to download and install Tomcat by yourself.

[Create Web Project]

1. File -> new -> Project

2. new -> Tomcat Server

3. next -> typing project name: MVCproject -> finish

The catalog of the project would as follows

4. Create folders in WEB-INF: classes and lib.
5. In File -> Project Structure, you can check your project SDK in Project tab.
   (For Mac User: Right Click Project-> Open Module Settings)

6. In Modules tab, change output path and test output path to classes folder you have created in step 4. The classes files built by IDE will be added into this folder.

7. In Dependencies tab, add lib folder to dependencies. You can put external dependencies like mysql jdbc in this folder.
   Jar or directories -> finding lib folder -> Jar Directory

8. Create a jsp file named login.jsp in web folder. And edit web.xml: add welcome-file-list in order to modified entrance page. Write hello world in <body>.

```
<welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
</welcome-file-list>
```

web.xml:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <welcome-file-list>
        <welcome-file>login.jsp</welcome-file>
    </welcome-file-list>

</web-app>
```

login.jsp:

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" pageEncoding="UTF-8"%>
<html>
  <head>
    <meta charset="utf-8">
    <title>Test</title>
  </head>

  <body>
  Hello World!
  </body>

</html>
```

9. Click Run button in upper right corner. Check whether using tomcat server. Then a page showing hello world shows up.

If you meet the following exception when you run the project, you need to change the port number.

```
08-Oct-2018 22:49:36.872 严重 [main] org.apache.catalina.util.LifecycleBase.handleSubClass
 org.apache.catalina.LifecycleException: Protocol handler initialization failed
    at org.apache.catalina.connector.Connector.initInternal(Connector.java:935)
    at org.apache.catalina.util.LifecycleBase.init(LifecycleBase.java:136)
    at org.apache.catalina.core.StandardService.initInternal(StandardService.java:533)
    at org.apache.catalina.util.LifecycleBase.init(LifecycleBase.java:136)
    at org.apache.catalina.core.StandardServer.initInternal(StandardServer.java:852)
    at org.apache.catalina.util.LifecycleBase.init(LifecycleBase.java:136)
    at org.apache.catalina.startup.Catalina.load(Catalina.java:633)
    at org.apache.catalina.startup.Catalina.load(Catalina.java:656) <4 internal calls>
    at org.apache.catalina.startup.Bootstrap.load(Bootstrap.java:306)
    at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:491)
Caused by: java.net.BindException: Address already in use
    at sun.nio.ch.Net.bind0(Native Method)
```

Click Tomcat 9.0.12 -> Edit configurations and then change the port number.

[Create first servlet]

1. Add login page in login.jsp <body>

```
<body>
please login the system
<br/>
<form action="LoginServlet" method="post">
    username: <input type="text" name="username"

value="<%=request.getAttribute("username") != null ?
request.getAttribute("username") : ""%>"/>
    <%=request.getAttribute("msg_username") != null ?
request.getAttribute("msg_username") : ""%>
    <br/>
    password: <input type="password" name="password"

value="<%=request.getAttribute("password") != null ?
request.getAttribute("password") : ""%>"/>
    <%=request.getAttribute("msg_password") != null ?
request.getAttribute("msg_password") : ""%><br/>
    <input type="submit" value="login"/>
    <br/>
    <%=request.getAttribute("msg") != null ?
request.getAttribute("msg") : ""%>
</form>
</body>
```

Result:

2. Next we create corresponding servlet file of login.jsp.
   a. Firstly, create a package named servlet in src folder.
   b. Right click servlet package -> new -> servlet, named it LoginServlet.

c. Add following code in web.xml to register servlet in the project so that the form in login.jsp can recognize the corresponding servlet.

```
<servlet>
  <servlet-name>LoginServlet</servlet-name>
  <servlet-class>servlet.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>LoginServlet</servlet-name>
  <url-pattern>/LoginServlet</url-pattern>
</servlet-mapping>
```

The function that we want to achieve in the class LoginServlet is that:

(1) When username and password are all null, print both are null behind those two input boxes.

(2) If one of them is null, print null behind the null one; hold data in another.

"request" and "response" are built-in objects of JSP, and the server (Tomcat) can help analysis those two objects and convert them to a type of HttpServletRequest and HttpServletResponse, which can be used by servlet.

We can use "request" to get basic information of Client by using following methods, such as:

- String getMethod()
- String getRequestURL()
- String getProtocal()
- String getServletPath()
- String getServerPort()
- String getRemoteAddr()

We can also use setArribute() and getArribute in request object. According to our tutorial, the code: `request.setAttribute(var1,var2)` in servlet can set an attribute in request, and its corresponding code: `request.getAttribute(var1)` in jsp can get its value var2.

d. Add code in LoginServlet:

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

    request.setCharacterEncoding("UTF-8");
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    if ((username == null || username.equals("")) && (password == null || password.equals(""))) {
        request.setAttribute("msg_username", "user name shouldn't be none");
        request.setAttribute("msg_password", "password shouldn't be none");
        request.getRequestDispatcher("login.jsp").forward(request, response);
    } else if (password == null || password.equals("")) {
        request.setAttribute("username", username);
        request.setAttribute("msg_password", "password shouldn't be none");
        request.getRequestDispatcher("login.jsp").forward(request, response);
    } else if (username == null || username.equals("")) {
        request.setAttribute("password", password);
        request.setAttribute("msg_username", "user name shouldn't be none");
        request.getRequestDispatcher("login.jsp").forward(request, response);
    } else {
        System.out.println("success");
    }
}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    doGet(request, response);
}
```

The page is shown below:

Success if console prints success after filled username and password.

[Build MVC framework]

1. Create 3 packages in src folder: bean, dao, service.

   (1) bean: model layer in MVC framework, usually being used to define entity class, data fields, manage objects and abstract objects. The data fields in it always can be mapped to the column name in table in database.

   Create a class: UserinfoBean in package bean, and add the getter and setter method for each private fields:

```
public class UserinfoBean {
    private int id;
    private String username;
    private String password;
}
```

   (2) Dao: data access layer, which usually design methods for database operation, and provides interface for accessing database.

   Create an interface: UserinfoDao in package dao:

```
public interface UserinfoDao
```

   Then create a class: UserinfoDaoImpl implementing this interface:

```
public class UserinfoDaoImpl implements UserinfoDao
```

   (3) Service: Business logic layer, which doesn't interact with database directly, it is middle layer between client and database, always encapsulate business logic and provide required data to Servlet or invoke different interface that provided by Dao.

   Create an interface: UserinfoService in package Service:

```
public interface UserinfoService
```

   Then create a class: UserinfoServiceImpl implementing this interface:

```
public class UserinfoServiceImpl implements UserinfoService
```

   (4) Servlet: which uses to interact data with web pages

We need to connect the database to verify username and password. To complete this login logic, now we use MVC framework.

   (1) In LoginServlet.java, just below the statement System.*out*.println("success");, adding following code, and then import UserinfoService and UserinfoServiceImpl.

```
System.out.println("success");
UserinfoService userinfoservice = new UserinfoServiceImpl();
```

```
int result = 0;
result = userinfoservice.login(username, password);
if (result == 1) {
    System.out.println("visiting database successfully");
    //request.getRequestDispatcher("BookServlet").forward(request, response);
} else {
    request.setAttribute("msg", "the username or password is wrong");
    request.getRequestDispatcher("login.jsp").forward(request, response);
}
```

At this moment, login method will report an error, because we have not defined login method on Service layer.

(2) In Service layer

Add method in UserinfoService:

```
int login(String username, String password);
```

Implement login method in UserinfoServiceImpl, and import UserinfoDao, UserinfoDaoImpl.

```
private UserinfoDao userinfoDao = new UserinfoDaoImpl();
@Override
public int login(String username, String password) {
    int result = 0;
    try{
        result=userinfoDao.login(username,password);
    }catch(Exception e){
        e.printStackTrace();
    }
    return result;
}
```

The login method reports an error, that's because we have not define this method in Dao layer.

(3) In Dao layer：

Add login method in UserinforDao,

```
int login(String username, String password);
```

Implement this method in UserinforDaoImpl, the detailed code will be given after connect to the database.

```
@Override
public int login(String username, String password) throws Exception {
    // TODO Auto-generated method stub
```

```
        return 0;
    }
```

[Connect the database]

Create table: userinfo, within which contains username, password and a primary key: id.

1. Use mysql workbench build the database

   (1) Create table: userinfo in database: mvcproject

   (2) 3 column: id, username, password, then click apply.

   (3) Add data in table: userinfo, then apply.

You can also create table using following command in cmd or terminal window.

```
mysql> create database mvcproject charset utf8;
Query OK, 1 row affected (0.01 sec)
mysql> use mvcproject;
Database changed
mysql> create table userinfo(
    -> id int primary key auto_increment,
    -> username varchar(20) not null,
    -> password varchar(20) not null);
Query OK, 0 rows affected (0.03 sec)
mysql> desc userinfo;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| username | varchar(20) | NO   |     | NULL    |                |
| password | varchar(20) | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
mysql> insert into userinfo(username, password) values('成龙',123);
Query OK, 1 row affected (0.01 sec)

mysql> insert into userinfo(username, password) values('范冰冰',123);
Query OK, 1 row affected (0.00 sec)

mysql> insert into userinfo(username, password) values('Helly',123);
Query OK, 1 row affected (0.00 sec)

mysql> select * from userinfo;
+----+-----------+----------+
| id | username  | password |
+----+-----------+----------+
|  1 | 成龙      | 123      |
```

```
|  2 | 范冰冰     | 123       |
|  3 | Helly      | 123       |
+----+-----------+----------+
3 rows in set (0.01 sec)
```

2. Intellij idea connect the database.
   Add "mysql-connector-java" jar file to lib folder

3. Create package: util in src folder. Create DBUtil.java in package util. this class defines methods connecting to the database and closing the database.

```
package util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;


public class DBUtil {
    public Connection getConnection() throws Exception{
        Connection connection=null;
        String diverClass=" ";
        String url=" ";
        String name=" ";
        String password=" ";
        Class.forName(diverClass);
        connection=DriverManager.getConnection(url,name,password);

        return connection;
    }
    /**
     * 关闭数据库链接
     * @param connection
     * @param preparedStatement
```

```
     * @param resultSet
     * @throws Exception
     */
    public void closeDBResource(Connection connection, PreparedStatement preparedStatement,
ResultSet resultSet) throws Exception{
        if(resultSet!=null){
            try{
                resultSet.close();
            }finally{
                resultSet=null;
            }
        }
        if(preparedStatement!=null){
            try{
                preparedStatement.close();
            }finally{
                preparedStatement=null;
            }
        }
        if(connection!=null){
            try{
                connection.close();
            }finally{
                connection=null;
            }
        }
    }

}
```

4. Obtain driverClass, url, name, password values.
   Find Database tab at the right side of Intellij idea, click add, choose data source, choose mysql.
   The diver class: com.mysql.jdbc.Driver
   url: jdbc:mysql://localhost:3306/mvcproject

Database
mvcproject@localhost 0 of 9
Ant Build
Maven Projects
Database
Schema
Console
Data Source
DDL Data Source
Data Source from URL
Data Source from Path
Driver and Data Source
Driver
Import from Sources...
MySQL
Amazon Redshift
Azure
ClickHouse
DB2
Derby
Exasol
H2
HSQLDB
MariaDB
Oracle
PostgreSQL
SQL Server
Sqlite
Sybase
Styles

Data Sources and Drivers
Project Data Sources
mvcproject@localhost
Drivers
Amazon Redshift
Azure (Microsoft)
ClickHouse
DB2 (JTOpen)
DB2 (LUW)
Derby (Embedded)
Derby (Remote)
Exasol
H2
HSQLDB (Local)
HSQLDB (Remote)
MariaDB
MySQL
Oracle
PostgreSQL
SQL Server (jTds)
SQL Server (Microsoft)
Sqlite (Xerial)
Sybase (jTds)
Sybase (Native)
Name: MySQL
Comment:
Settings Advanced
Class: com.mysql.jdbc.Driver
Dialect: MySQL
Code Style: Default
Tx: Auto
Auto sync
Send application info
Schema control: Automatic
Driver files
MySQL Connector/J ver. 5.1.46 [latest]
Show License
URL templates
Name
Template
default
jdbc:mysql://{host::localhost}?[:{port::3306}][/{database}?][\?<&,user={user},password={pass...
default
jdbc:mysql://address=\(protocol=tcp\)\(<\)\(,host={host:host_ipv6:localhost},port={port::3306}...
Cancel
Apply
OK

click Test Connection. If connect to the database, it shows successful.

5. Filled the blank in method getConnection.

```
String diverClass="com.mysql.jdbc.Driver";
String url="jdbc:mysql://localhost:3306/mvcproject";
String name="root";
String password="[redacted]";
```

6. Now we implement method login in Dao layer.

Import following package:

```
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.PreparedStatement;

import bean.UserinfoBean;
import com.mysql.jdbc.ResultSetImpl;
import util.DBUtil;
```

Add following code:

```
Connection connection=null;
DBUtil dbutil = new DBUtil();
ResultSet resultSet =null;
PreparedStatement preparedStatement = null;

@Override
public int login(String username, String password) throws Exception {
  // TODO Auto-generated method stub
  int result = 0;
  connection = dbutil.getConnection();
  String sql = "select count(*) from userinfo where username=? and password=?";
  preparedStatement = connection.prepareStatement(sql);
  preparedStatement.setString(1, username);
  preparedStatement.setString(2, password);
  resultSet = preparedStatement.executeQuery();
  while (resultSet.next()) {
    result = resultSet.getInt(1);
  }
  dbutil.closeDBResource(connection, preparedStatement, resultSet);
  return result;
}
```

Attention: We wrote "**throws** Exception" defined in dbutil.getConnection() and modified in the corresponding interface and handle this exception in service layer.

```
@Override
public int login(String username, String password) {
    int result = 0;
    try {
        result = userinfoDao.login(username, password);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;
}
```

7. Save files and click run tomcat, the result shows below:

```
[2018-10-09 11:37:50,111] Artifact MVCproject:war exploded: Artifact is
[2018-10-09 11:37:50,111] Artifact MVCproject:war exploded: Deploy took
09-Oct-2018 11:37:58.920 信息 [ContainerBackgroundProcessor[StandardEngir
09-Oct-2018 11:37:58.981 信息 [ContainerBackgroundProcessor[StandardEngir
success
Tue Oct 09 11:38:00 CST 2018 WARN: Establishing SSL connection without s
visiting database successfully
```

[Implement register function]

1. Create jsp page named register.jsp, and add following codes in <body>.

```
Please register a user
</br>
<form action="RegisterServlet" method="post">
  username: <input type="text" name="username"
            value="<%=request.getAttribute("username") != null ?
request.getAttribute("username") : ""%>" />
  <%=request.getAttribute("msg_username") != null ? request.getAttribute("msg_username") :
""%>
  <br /> password:<input type="password" name="password"
            value="<%=request.getAttribute("password") != null ?
request.getAttribute("password") : ""%>" />
  <%=request.getAttribute("msg_password") != null ? request.getAttribute("msg_password") :
""%><br />
  confirm password:<input type="password" name="con_password"
            value="<%=request.getAttribute("con_password") != null ?
request.getAttribute("con_password") : ""%>" />
  <%=request.getAttribute("msg_con_password") != null ?
request.getAttribute("msg_con_password") : ""%><br />
  <input type="submit" value="Register" />
</form>
<a href="login.jsp">Return back to login page</a>
<%=request.getAttribute("msg") != null ? request.getAttribute("msg") : ""%>
```

There are three input types, which are handled by a form, including username, password and confirm password in this page.
The username, password and confirm password will be posted to servlet marked by action after click submit button.

2. Create servlet
Create servlet: RegisterServlet.java in package: servlet. (Do not forget registering this servlet in web.xml, method given in section: **[Create first servlet]**)
What we needed to accomplish is as following:
   a. If username and password are both null, print information after two input box.
   b. If username is null only, print that username should not be null and clear

password.

c. If password is null only, print that password should not be null but hold username.

d. If username and password are both filled, but confirm password is null, print that confirm password should not be null and hold username and password.

e. If three inputs are all filled, but with different values between password and confirm password, print that two passwords are different and clear confirm password.

Add code:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    doGet(request, response);
}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String username="";
    String password="";
    String conPassword="";
    username=request.getParameter("username");
    password=request.getParameter("password");
    conPassword=request.getParameter("con_password");
    if((username==null||username.equals(""))&&(password==null||password.equals(""))){
        request.setAttribute("msg_username", "user name shouldn't be none");
        request.setAttribute("msg_password", "password shouldn't be none");
        request.getRequestDispatcher("register.jsp").forward(request, response);

    }else if(username==null||username.equals("")){
        request.setAttribute("password", password);
        request.setAttribute("msg_username", "user name shouldn't be none");
        request.getRequestDispatcher("register.jsp").forward(request, response);
    }else if(password==null||password.equals("")){
        request.setAttribute("username", username);
        request.setAttribute("msg_password", "password shouldn't be none");
        request.getRequestDispatcher("register.jsp").forward(request, response);
    }else{
        if(conPassword==""||conPassword.equals("")){
            request.setAttribute("username", username);
            request.setAttribute("password", password);
            request.setAttribute("msg_con_password", "Please confirm password");
            request.getRequestDispatcher("register.jsp").forward(request, response);
        }else{
```

```
            if(!conPassword.equals(password)){
                request.setAttribute("password", password);
                request.setAttribute("username", username);
                request.setAttribute("msg_con_password", "Two password is not same");
                request.setAttribute("con_password", "");
                request.getRequestDispatcher("register.jsp").forward(request, response);
            }else{
                System.out.println("Success");
            }
        }

    }

}
```

then run, we will get a Success print showing success.

3. Encapsulate object:

   Next we start to encapsulate the objects. Create a userinfoBean object, and add information to userinfoBean, passing it to under layer.

   Add following code after System.out.println("Success");

```
UserinfoService userinfoService=new UserinfoServiceImpl();
UserinfoBean userinfoBean=new UserinfoBean();
userinfoBean.setUsername(username);
userinfoBean.setPassword(password);

int result=userinfoService.registerUserinfo(userinfoBean);
System.out.println("In servlet"+result);
if(result==1){
    request.setAttribute("msg", "Register is success, please login to the system");
    request.setAttribute("username", username);
    request.getRequestDispatcher("login.jsp").forward(request, response);
```

```
}else{
    request.setAttribute("msg", "Register is failed");
    request.getRequestDispatcher("register.jsp").forward(request, response);
}
```

4. Service layer

   It will report an error on method registerUserinfo, for the reason that we haven't defined this method in Service layer.

   Create method registerUserinfo in service interface and corresponded implementation in userinfoServiceImpl:

```
@Override
public int registerUserinfo(UserinfoBean userinfoBean) {
    int result=0;
    try{
        result=userinfoDao.registerUserinfo(userinfoBean);
    }catch(Exception e){
        e.printStackTrace();
    }
    return result;
}
```

5. Dao layer

   The same error of the method registerUserinfo(). We need to create this method on Dao layer.

   There is a problem, on the step of inserting data in database, that, id is a primary key which cannot be duplicated or null and how can we determine the new user's id?

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	D
id	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
username	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
password	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

   In mysql, we can set the auto increment mechanism.

   In class UserinfoDaoImpl, add following code:

```
public int registerUserinfo(UserinfoBean userinfoBean) throws Exception {
    int result = 0;
    connection = dbutil.getConnection();
    String sql = "insert into userinfo (username, password) values (?,?)";
    preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setString(1, userinfoBean.getUsername());
    preparedStatement.setString(2, userinfoBean.getPassword());
    result = preparedStatement.executeUpdate();
    dbutil.closeDBResource(connection, preparedStatement, resultSet);
```

```
    return result;
}
```

After success, page jump like following:

mysql workbench:

id	username	password
1	成龙	123
2	范冰冰	123
3	Hellv	123
4	Lili	123
NULL	NULL	NULL

[List books information]

After login in, our page should show the list of books information. Now we introduce how to implement this function on J2EE MVC framework in this example.

1. In the same schema, we create a table called book, with attributes: id (primary key), bookname, author, price and date_added.

Name: book

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
id	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
bookname	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
author	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
price	DOUBLE	☐	☑	☐	☐	☐	☐	☐	☐	
date_added	VARCHAR(10)	☐	☑	☐	☐	☐	☐	☐	☐	

To create a table in terminal window, you can use following query.

```
create table book(
    id int primary key auto_increment,
    bookname varchar(30) not null,
    author varchar(20) not null,
```

```
price double not null,
date_added varchar(10) not null);
```

Then adding some data in table book:

id	bookname	author	price	date_added
1	C++	Emy	79.5	2019-11-1
2	J2EE	Mary	150	2019-09-10
3	SQL	Stephane	500	2019-08-09
4	Java	Lili	99	2019-11-09

To insert a row in terminal window, you can use following query.

```
insert into book (bookname,author,price,date_added) values ('C++','Emy',79.5,'2019-11-1');
insert into book (bookname,author,price,date_added) values ('J2EE','Mary',150,'2019-09-10');
insert into book (bookname,author,price,date_added) values ('SQL','Stephane',500,'2019-08-09');
insert into book (bookname,author,price,date_added) values ('Java','Lili',99,'2019-11-09');
```

……

2. Build mvc framework for Book

   Previously, we build mvc framework for userinfo. Now we need to do same similar operation to book. First we create some java files.

   (1) Bean layer: BookBean and its attribute with getter and setter:

```
private int id;
private String bookName;
private String author;
private double price;
private String addingDate;
```

   (2) Dao layer: BookDao and BookDaoImpl (implements BookDao)

   (3) Service layer: BookService and BookServiceImpl (implements BookService)

   (4) Servlet layer: BookServlet (do not forget registering in web.xml)

3. Build front end jsp

   (1) Create BookList.jsp, add following code in <body>:

```
<%
  List<BookBean> bookBeanList = (List<BookBean>) request.getAttribute("bookList");
%>
<table>
  <caption id="a"> <h2>BookList</h2> </caption>
```

```
<tr>
    <th>BookID</th>
    <th>BookName</th>
    <th>Author</th>
    <th>Price</th>
    <th>Adding Date</th>
    <th>Operation</th>
</tr>

<%
    if (bookBeanList != null && bookBeanList.size() > 0) {
        for (int i=0;i<bookBeanList.size();i++) {
%>
<tr>
    <td><%=i+1%></td>
    <td><%=bookBeanList.get(i).getBookName()%></td>
    <td><%=bookBeanList.get(i).getAuthor()%></td>
    <td><%=bookBeanList.get(i).getPrice() %></td>
    <td><%=bookBeanList.get(i).getAddingDate() %></td>
    <td><a href="#">delete</a></td>
</tr>
<%
    }
} else {
%>
<tr>
    <td colspan="6">can not get the book infomation</td>
</tr>
<%
    }
%>
</table>
```

`<a href="#">`. this is the servlet link that added in our following work about delete function. So it is not completed now.

At the top of jsp file, add following two import line:

```
<%@ page import="bean.BookBean" %>
<%@ page import="java.util.List" %>
```

4. Servlet layer

   Do not forget to import related package

```
@WebServlet(name = "BookServlet ")
public class BookServlet extends HttpServlet {

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

    BookService bookService=new BookServiceImpl();
    List<BookBean> bookList=bookService.fetchBookList();
    request.setAttribute("bookList", bookList);
    request.getRequestDispatcher("BookList.jsp").forward(request, response);
  }

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

}
```

5. Service layer

   Add following code in BookServiceImpl:

```
BookDao bookDao=new BookDaoImpl();
@Override
public List<BookBean> fetchBookList() {
  List<BookBean> bookList=null;
  try{
    bookList=bookDao.fetchBookList();
  }catch(Exception e){
    e.printStackTrace();
  }
  return bookList;
}
```

6. Dao layer

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import bean.BookBean;
```

```java
import util.DBUtil;

public class BookDaoImpl implements BookDao {
    DBUtil dbutil = new DBUtil();
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    Statement statement = null;
    ResultSetMetaData metaData = null;
    @Override
    public List<BookBean> fetchBookList() throws Exception{
        List<BookBean> bookBeanList=null;
        connection = dbutil.getConnection();
        String sql="select * from book";
        preparedStatement=connection.prepareStatement(sql);
        resultSet=preparedStatement.executeQuery();
        bookBeanList=new ArrayList<BookBean>();
        while(resultSet.next()){
            BookBean bookBean=new BookBean();
            bookBean.setId(resultSet.getInt("id"));
            bookBean.setBookName(resultSet.getString("bookname"));
            bookBean.setAuthor(resultSet.getString("author"));
            bookBean.setPrice(resultSet.getDouble("price"));
            bookBean.setAddingDate(resultSet.getString("date_added"));
            bookBeanList.add(bookBean);
        }
        dbutil.closeDBResource(connection, preparedStatement, resultSet);
        return bookBeanList;
    }
}
```

7. Then modified LoginServlet file, implement page jump function.

```java
} else {
    System.out.println("success");
    UserinfoService userinfoservice = new UserinfoServiceImpl();
    int result = 0;
    result = userinfoservice.login(username, password);
    if (result == 1) {
        System.out.println("visiting database successfully");
        request.getRequestDispatcher( s: "BookServlet").forward(request, response);
    } else {
        request.setAttribute( s: "msg",  o: "the username or password is wrong");
        request.getRequestDispatcher( s: "login.jsp").forward(request, response);
    }
}
```

8. Run our project, the page is shown below:

[delete function in BookList]

1. Modified BookList.jsp file
   (1) `<a href="#">` add a DeleteBookServlet to implement delete function.
   `<td><a href="DeleteBookServlet?id=<%=bookBeanList.get(i).getId()%>">delete</a></td>`
   (2) add an output after <table>, in order to show the result information of operation.
   `<%=request.getAttribute("msg") != null ? request.getAttribute("msg") : ""%>`

   we pass the id as parameter in DeleteBookServlet, which is the primary key in database.

2. Servlet layer
   Create DeleteBookServlet (do not forget registering in web.xml), adding following code:

```
try{
   String idString=request.getParameter("id");
   int id=Integer.parseInt(idString);
   BookService bookService=new BookServiceImpl();
   int result=bookService.deleteBookById(id);
   if(result==1){
      request.setAttribute("msg", "delete successfully");
   }else{
      request.setAttribute("msg", "delete failed");
   }
   request.getRequestDispatcher("BookServlet").forward(request, response);
}catch(Exception e){
   request.setAttribute("msg", "The id of this user is null");
   request.getRequestDispatcher("BookServlet").forward(request, response);
}
```

3. Other layer modification

Using familiar method modifying service layer and dao layer.

Create deleteBookById(int id) method: (in BookDaoImpl.java)

```
@Override
public int deleteBookById(int id) throws Exception {
    int result=0;
    connection=dbutil.getConnection();
    String sql="delete from book where id=?";
    preparedStatement=connection.prepareStatement(sql);
    preparedStatement.setInt(1, id);
    result=preparedStatement.executeUpdate();
    dbutil.closeDBResource(connection, preparedStatement, resultSet);
    return result;
}
```

Result:

Designed by Yueming ZHU (in 2017) and Weiduo LIAO(modified in 2018).