ハッシュログイン機能付き図書の貸出履歴検索システム-JavaEE8,PostgreSQL

office・M
2024年9月5日 14:48

¥1,000 •••

JavaEE8（JSP・Servlet）の環境でWebアプリケーションの開発方法を学ぶ講座をシリーズで提供しています。今回は部品として作成されたDBを操作するクラスファイル（DataAccessObject）を活用する方法を学びます。チーム開発での役割分担を想定しています。

2024年9月よりECLIPSEのバージョンを最新版（Version: 2024-06 (4.32.0)）に変更しました。

目次

開発概要

JavaEE8アプリケーション・サーバ（Tomcat）とデータベース・サーバ（PostgreSQL）を利用して開発を行うということを前提に実装を進めていきます。開発ツールには統合開発環境のEclipseを用いるこ

とにします。

題材としてハッシュログイン処理付き図書の貸出履歴管理を行うWebアプリをMVCモデルで作成します。ハッシュ値に変換されたパスワードにより認証を行います。

JavaEE8環境ですので使用言語は当然Java（JSP・Servlet）となり、設計技法にはMVCモデルを使います。MVCモデルとはWebアプリケーションの構成をModel（業務ロジック）ーView（表示）ーController（制御）に分割して設計する技法です。３つのモデルに役割分担することで部品化が促され、ひいてはチーム開発に貢献します。

今回、データベースの抽出処理など業務ロジックを実現するBeanは作成しません。他のチームメンバーが作成したという想定で提供されたBeanを活用することにします。

実装していく過程でEclipseの利用方法が不明な場合は、別途メール等でお知らせください。この講座では、作成済みの仕様と設計書から「**JavaEE8環境でのMVCモデルの実装の仕組みを学ぶ**」ことを目的とします。

開発方針

Webアプリケーションの開発環境には図１のようにEclipse2024を用います。

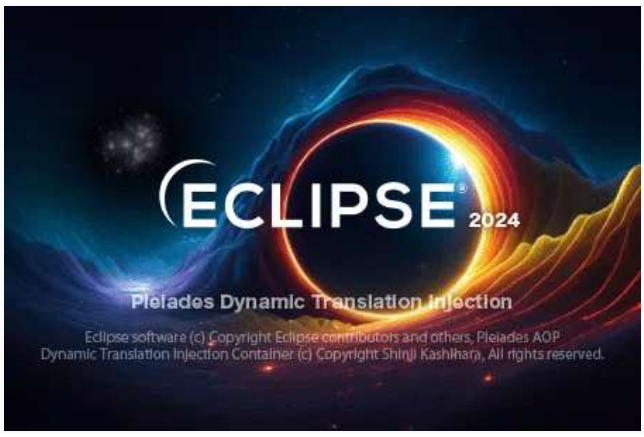

図1. 開発環境 Eclipse2024

Spring等のフレームワークは使用しません。これは、MVCモデルにおけるhttpプロトコルの処理の実装を直に学んでいただきたいためです。

開発に必要な仕様と最小限の設計ドキュメントは提示します。この情報をもとに、まずは、自分で試行錯誤しながら実装してみてください。

DB設計

認証方式

ログイン処理はハッシュ値による認証を行います。ハッシュ値は、文字列からハッシュ値へ変換した場合、そのハッシュ値から元の文字列に戻すことはできないという特徴があります。

ユーザが決めたパスワードをハッシュ値にして事前にDBに保存し、再度ユーザがログインした時に入力されたパスワードからハッシュ値を計算した結果とDBに保存してあるハッシュ値とを比較し、認証します。パスワードをDBにそのまま保存するよりセキュリティが高まります。今回はこの仕組みを実装します。

ハッシュ関数には**SHA-256**を使います。SHA-256では256bitのハッシュ値(**64桁**)が出力されます。様々な場面でよく使用されている関数であり、256bit以上が推奨とされています（MD-5やSHA-128は推奨されません）。

Java標準機能であるMessageDigestクラスを使用することでSHA-256のハッシュ値を生成することは可能ですが若干使い辛いため、今回は、外部ライブラリである「**Apache Commons Codec**」の**DigestUtilsクラス**でハッシュ値を求めます。

論理設計

図書の貸出履歴管理のERDモデルを示します。

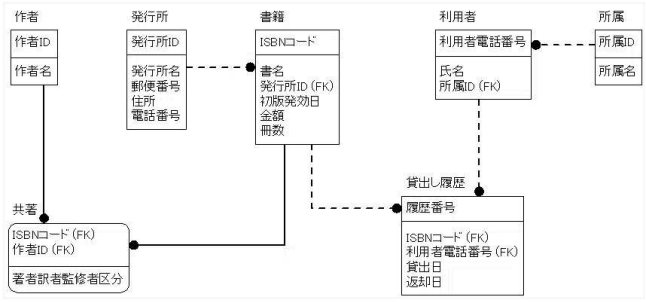

図2．図書の貸出履歴管理ERDモデル

物理設計

接続するPostgreSQLサーバのDB名は「**library**」とします。

（DB構造,スキーマ）

```
CREATE TABLE T_利用者
  (利用者電話番号    CHAR(12)    NOT NULL,
  氏名            VARCHAR(20)    NOT NULL,
  所属            VARCHAR(20),
  PRIMARY KEY (利用者電話番号));
CREATE TABLE T_作者
  (作者id        SERIAL         NOT NULL,
  作者名        VARCHAR(20)    NOT NULL,
  PRIMARY KEY (作者id));
CREATE TABLE T_発行所
  (発行所id        SERIAL    NOT NULL,
  発行所名        VARCHAR(20)    NOT NULL,
  郵便番号        CHAR(8),
  住所            VARCHAR(50),
  電話番号        CHAR(12),
  PRIMARY KEY (発行所id));
CREATE TABLE T_書籍
  (ISBNコード    CHAR(17)    NOT NULL,
  書名            VARCHAR(50)    NOT NULL,
  発行所id        INT    NOT NULL,
  初版発効日    DATE,
  金額            INT,
  冊数            INT,
  PRIMARY KEY (ISBNコード),
  FOREIGN KEY (発行所id)
    REFERENCES T_発行所 ON DELETE RESTRICT ON UPDATE RESTRICT);
CREATE TABLE T_共著
  (ISBNコード    CHAR(17)    NOT NULL,
  作者id        INT            NOT NULL,
  共著区分        VARCHAR(10)    NOT NULL,
  PRIMARY KEY (ISBNコード,作者id),
  FOREIGN KEY (ISBNコード) REFERENCES T_書籍 ON DELETE RESTRICT
                                        ON UPDATE RESTRICT,
  FOREIGN KEY (作者id) REFERENCES T_作者 ON DELETE RESTRICT
                                      ON UPDATE RESTRICT);
CREATE TABLE T_貸出し履歴
  (履歴番号        INT            NOT NULL,
  ISBNコード    CHAR(17)    NOT NULL,
  利用者電話番号    CHAR(12)    NOT NULL,
  貸出日    DATE,
  返却日    DATE,
  貸出冊番  INT,
  PRIMARY KEY (履歴番号),
  FOREIGN KEY (ISBNコード) REFERENCES T_書籍 ON DELETE RESTRICT
                                      ON UPDATE RESTRICT,
  FOREIGN KEY (利用者電話番号) REFERENCES T_利用者
                                      ON DELETE RESTRICT
                                      ON UPDATE RESTRICT);
```

SQL-DDLとSQL-DMLを含んだテキストファイルを添付します。
A5：SQL Mk-2などのSQLクライアントソフトを使ってSQL文を流し込んでください。図書の貸出履歴管理のテーブル群が作成されサンプルデータが入力されます。

（SQL-DDL,SQL-DML）

図書の貸出履歴管理SQL文.txt

7.72 KB

ファイルダウンロードについて

ダウンロード

念のために、作成したすべてのテーブルとビューの中身を確認しておいてください。すべてのinsertデータがDBに格納されているか、しっかりと確認してください！図３のように**V_利用者毎貸出し履歴の出力件数は２１件**あります。

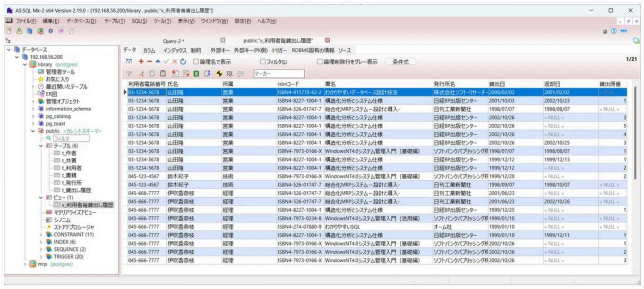

図3．V_利用者毎貸出し履歴の出力

外部設計

接続urlは**http://localhost:8080/booklib/Library**とします。よってEclipseの動的Webプロジェクトの名前は**booklib**となり、サーブレットのurlパターンは**Library**となります。

Eclipseのメニューバーより
ファイル→新規→動的Webプロジェクト→「booklib」プロジェクトを作成する→図４の内容で設定する

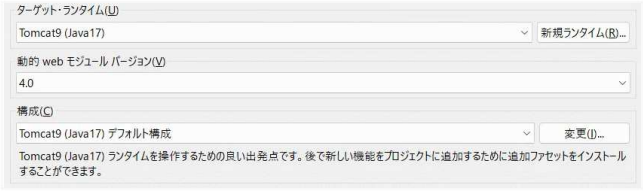

図4．動的Webプロジェクトの設定

※作成済みであればこの処理は必要ありません

---

urlにアクセスすると図５のような、ログイン画面が表示されます。IDは電話番号（ハイフン付き）でパスワードも電話番号（ハイフンなし）となります。正しく認証が行われると図６のように電話番号をキーとした貸出書籍の検索が行われ結果の貸出履歴が表示されます。

図5．ログイン画面

図6．認証後の図書貸出履歴の検索結果画面

内部設計

クラス連携図は図７のようになります。

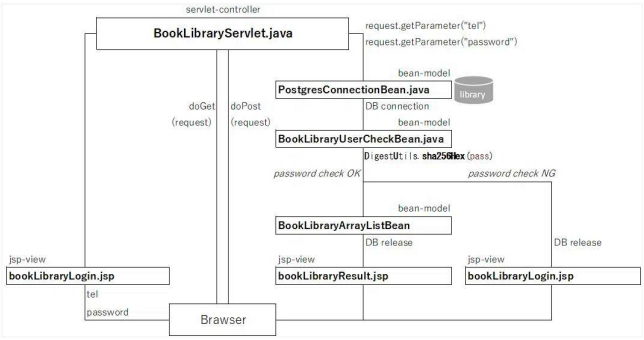

図7．MVCモデル図（クラス連携図）

DBに接続して図書検索を行う業務ロジック（**DataAccessObject**）を実装したBeanは提供されます。使い方とクラス図は以下のようになります。

PostgresConnectionBean.classの使い方とクラス図

（使い方）
コンストラクタの引数でPostgreSQLサーバのIPとDB名を指定して接続コネクションを実体化します。
ゲッターメソッドで接続connectionを取得します。
接続パラメータはデフォルトで以下のように指定されています。当然、PostgreSQL用ドライバの登録が必要です。
・private String driver = "org.postgresql.Driver";
・private String url = "jdbc:postgresql://引数１:5432/引数２";
・private String user = "postgres";
・private String password = "postgres";

（クラス図）

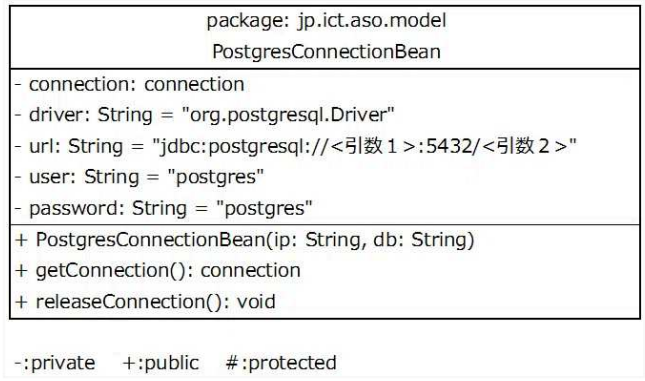

図8. PostgresConnectionBeanクラス図

BookLibraryUserCheckBean.classの使い方とクラス図

（使い方）
Beanは引数なしのコンストラクタで実体化します。
セッターメソッドでPostgreSQLサーバの接続connectionを設定します。userCheck()メソッドを実行することでT_USERテーブルから電話番号(tel)をキーとしてパスワードが登録されているかどうか検索します。
パスワードが登録されていればpasswordフィールドに内容が設定されます。
利用側のクラスはゲッターメソッドでpasswordフィールドの内容を取得します。

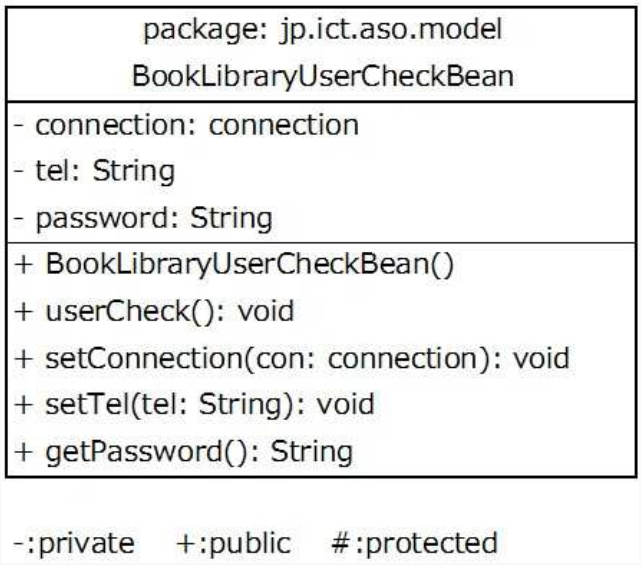

図9．BookLibraryUserCheckBeanクラス図

BookLibraryArrayListBean.classの使い方とクラス図

Beanは引数なしのコンストラクタで実体化します。
セッターメソッドでPostgreSQLサーバの接続connectionを設定します。
bookSearch()メソッドを実行することでV_利用者毎貸出し履歴テーブルから電話番号(tel)をキーとして貸出履歴を検索します。
検索結果はhtml形式でresultSetフィールドに整形表組されます。
利用側のクラスはゲッターメソッドでresultSetフィールドの内容を取得します。

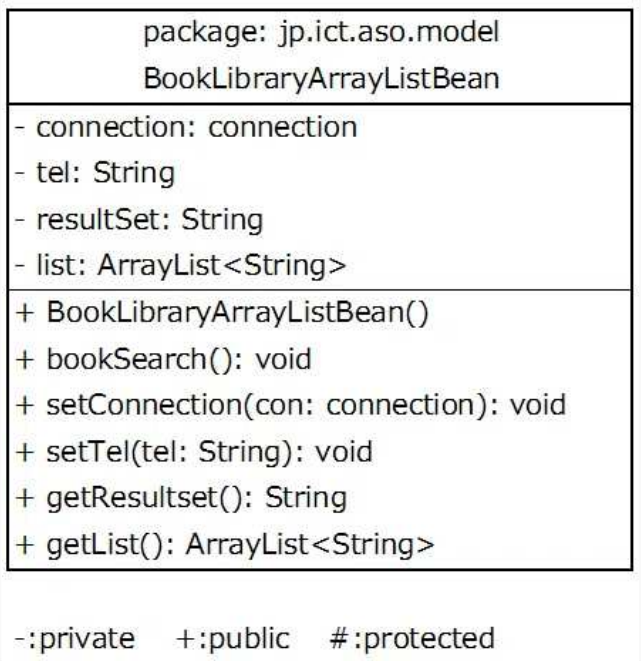

図10．BookLibraryArrayListBeanクラス図

実装準備

ハッシュ算出用ライブラリの登録

「**Apache Commons Codec**」の**DigestUtilsクラス**でハッシュ値を求めます。よって**commons-codec-1.15.jar**ファイルをパッケージエクスプローラ内の**booklib**プロジェクトの**src/main/webapp/WEB-INF/lib**フォルダにドラッグ＆ドロップして登録します。

commons-codec-1.15.jar

346 KB

ファイルダウンロードについて

ダウンロード

図11．ハッシュライブラリの配置位置

パスワードのハッシュ値の算出

パスワードハッシュ化プログラムの作成

DBに保存するハッシュ化されたパスワードの値を求めるために、通常のクラスファイルでHash化プログラムを作成します。

Eclipseパッケージ・エクスプローラより
booklibプロジェクトを右クリック→新規→クラス
→以下の内容で作成する

・HashCreate.java
　　パッケージ：jp.ict.aso.appli

名前：HashCreate
ソースコード：

```java
package jp.ict.aso.appli;

import org.apache.commons.codec.digest.DigestUtils;

public class HashCreate {
    public static void main(String[] args) {
        String str1=DigestUtils.sha256Hex("0312345678");
        System.out.println(str1);
        String str2=DigestUtils.sha256Hex("0451234567");
        System.out.println(str2);
        String str3=DigestUtils.sha256Hex("0456667777");
        System.out.println(str3);
    }
}
```

図12．HashCreate.java

HashCreate.javaを実行します。

エディタ画面を右クリック → 実行 → Javaアプリケーション
→ 結果は左下コンソールに出力される

図13．パスワードのハッシュ化の結果

※結果のハッシュ値はコンソールからマウスで選択コピーして、プログラム内にコメントとして書き込んでおいたほうが、あとあと便利です。

```java
package jp.ict.aso.appli;

import org.apache.commons.codec.digest.DigestUtils;

public class HashCreate {
    public static void main(String[] args) {
        String str1=DigestUtils.sha256Hex("0312345678");
        System.out.println(str1);
//      03f19e1dad737c9a2d228fc5fba499253e79816bf4b66f64e067845bb12efeaf

        String str2=DigestUtils.sha256Hex("0451234567");
        System.out.println(str2);
//      42559d4ba2c7e909c94cf550bff167fbdb533d71ea4f50ed70a3208ddc5f0ec6

        String str3=DigestUtils.sha256Hex("0456667777");
        System.out.println(str3);
//      01fe7daff5c7626fc27eae7f9da28e03edb90aa4751063d9f482fea86f2b2c33

    }
}
```

図14．HashCreate.java内にパスワードに対応したハッシュ値を記述

パスワード認証用テーブルに格納

ログイン情報を管理する**T_USERテーブル**を作成します。パスワード格納領域はHash文字列の幅にします。処理の手順は以下のようになります。
・T_USERテーブルを作成（PASSWORDフィールドは64文字となる）
・ハッシュ値を登録（図14で保存したハッシュ値などからINSERT文を生成する）
A5：SQL Mk-2などのSQLクライアントソフトを使って以下のSQL文を流し込みます。

（SQL-DDL）

```
create table T_USER(
TEL varchar(20) not null,
PASSWORD varchar(64) not null,
primary key(TEL)
);
```

(SQL-DML)

```
insert into T_USER values('03-1234-5678',
'03f19e1dad737c9a2d228fc5fba499253e79816bf4b66f64e067845bb12efeaf');
insert into T_USER values('045-123-4567',
'42559d4ba2c7e909c94cf550bff167fbdb533d71ea4f50ed70a3208ddc5f0ec6');
insert into T_USER values('045-666-7777',
'01fe7daff5c7626fc27eae7f9da28e03edb90aa4751063d9f482fea86f2b2c33');
```

念のために、作成したテーブルの中身を確認しておいてください。Webページからのコピペの流し込みでは不適切な文字が挿入される場合がありますので注意してください。図15のようにすべてのinsertデータがDBに格納されているか、しっかりと確認してください！

tel	password
03-1234-5678	03f19e1dad737c9a2d228fc5fba499253e79816bf4b66f64e067845bb12efeaf
045-123-4567	42559d4ba2c7e909c94cf550bff167fbdb533d71ea4f50ed70a3208ddc5f0ec6
045-666-7777	01fe7daff5c7626fc27eae7f9da28e03edb90aa4751063d9f482fea86f2b2c33

図15．T_USERテーブルの状態

PostgreSQLのjdbcドライバを配置

org.postgresql.driver.jarファイルをパッケージエクスプローラ内の**booklib**プロジェクトの**src/main/webapp/WEB-INF/lib**フォルダにドラッグ＆ドロップして登録します。

図16. PostgreSQLドライバの配置位置

org.postgresql.driver.jar

325 KB

ファイルダウンロードについて

ダウンロード

DAO用JavaBeansクラスの使用準備

(1)クラスファイルの準備をします
以下Windows環境を想定しています。
事前に「**提供BeanLibrary**」フォルダを作成しておきます。
（例　c:￥提供BeanLibrary）

提供BeanLibraryフォルダ内に以下の**３つのクラスファイル**を保存しておきます。
（**PostgresConnectionBean.clas、BookLibraryUserCheckBean.class、BookLibraryArrayListBean.class**）

その際**パッケージの階層**に従ってください。
（例　c:￥提供BeanLibrary￥jp￥ict￥aso￥model￥〇〇〇.class）

(2)DAO用の３つのクラスファイルをEclipseのビルド・パスに追加します

Eclipseパッケージ・エクスプローラより
booklibプロジェクトを右クリック→ビルド・パス→ビルド・パスの構成→「ライブラリ」タブ→「クラスパス」クリック→外部クラス・フォルダーの追加→「提供BeanLibrary」を指定します→最後に「適用して閉じる」をクリック

※**Eclipse2024の場合** → booklibプロジェクトを右クリック→プロパティ
　→Javaのビルド・パスで設定します。
※図17のように一度設定されていれば再度設定する必要はありません。

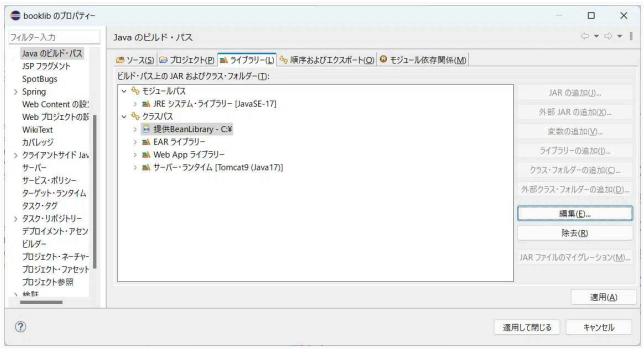

図17．Javaのビルドパス追加画面

実装手順

１．リクエストコントロール用Servletクラスを作成します

Eclipseパッケージ・エクスプローラより
booklibプロジェクトを右クリック→新規→その他→Web→サーブレット
→以下の内容で作成する

・BookLibraryServlet.java
　　パッケージ：jp.ict.aso.controller
　　クラス名：BookLibraryServlet
　　ソースコード：考えましょう！　※アノテーションは**/Library**

```
【ヒント】ソースコードの構成
doGet
  // セッションを実体化してエラーメッセージ(なし)を設定
  String message="";
  HttpSession session=request.getSession();
  session.setAttribute("error", message);
  // bookLibraryLogin.jspにフォワード

doPost
  // リクエストスコープの取得
  request.setCharacterEncoding("UTF-8");
```

```
    String tel = request.getParameter("tel");
    String pass = request.getParameter("password");
    // 入力されたパスワードのハッシュ値を求める
    String hashPass = DigestUtils.sha256Hex(pass);
    // DB接続パラメータの設定
    String ip="192.168.56.200"; ← 自分のIPに変更
    String db="library";
    // DB接続用コネクション作成
    PostgresConnectionBean pcb = new PostgresConnectionBean(ip,db);
    Connection con=pcb.getConnection();

    // ユーザチェックBeanの実体化（DB接続）
    BookLibraryUserCheckBean blu = new BookLibraryUserCheckBean();
    blu.setConnection(con);
    blu.setTel(tel);
    blu.userCheck();

    // ユーザ(パスワード)チェックOK
    if(hashPass.equals(blu.getPassword())) {
      // DB検索（電話番号）
      BookLibraryArrayListBean bla = new BookLibraryArrayListBean();
      bla.setConnection(con);
      bla.setTel(tel);
      bla.bookSearch();
      // DB接続用コネクション開放
      pcb.releaseConnection(con);
      // リクエストスコープに検索結果を保存
      request.setAttribute("resultset", bla);
      // bookLibraryResult.jspにフォワードして検索結果出力

    // ユーザ(パスワード)チェックNG
    }else {
      // DB接続用コネクション開放
      pcb.releaseConnection(con);
      // エラーメッセージをセッションに設定
      String message="電話番号またはパスワードが違います";
      HttpSession session=request.getSession();
      session.setAttribute("error", message);
      // bookLibraryLogin.jspにフォワードしてアカウント再入力
  }
```

２．ログイン画面のJSPファイルを作成します

Eclipseパッケージ・エクスプローラより
booklibプロジェクトを右クリック→新規→その他→Web→JSPファイル
→以下の内容で作成する

・bookLibraryLogin.jsp
　保存場所：booklib/src/main/webapp/**WEB-INF/jsp ← 注意！**
　ファイル名：bookLibraryLogin.jsp
　ソースコード：考えましょう！

【ヒント】ソースコードの構成

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>ログイン</title>
</head>
  <% String message=(String)session.getAttribute("error"); %>
<BODY>
<CENTER>
<P><FONT FACE="Verdana" SIZE=6><B>Login</B></FONT></P>
<HR WIDTH=300>
<FORM METHOD="POST" ACTION="Library">
<TABLE BORDER=0>
<TR>
<TD>利用者電話番号：</TD>
<TD><INPUT TYPE=TEXT NAME="tel" SIZE=20></TD>
</TR>
<TR>
<TD>パスワード：</TD>
<TD><INPUT TYPE=PASSWORD NAME="password" SIZE=20></TD>
</TR>
</TABLE>
<INPUT TYPE=SUBMIT VALUE=ログイン><INPUT TYPE=RESET VALUE=リセット>
</FORM>
<HR WIDTH=300>
  <%=message %>
  <% session.removeAttribute("error"); %>
</CENTER>
</BODY>
</html>
```

３．DB検索結果画面のJSPファイルを作成します

Eclipseパッケージ・エクスプローラより
booklibプロジェクトを右クリック→新規→その他→Web→JSPファイル
→以下の内容で作成する

・bookLibraryResult.jsp
　保存場所：booklib/src/main/webapp/**WEB-INF/jsp ← 注意！**
　ファイル名：bookLibraryResult.jsp
　ソースコード：考えましょう！**リクエストスコープで転送されたbeanの**
　　インスタンスを取得するのを忘れないように！

【ヒント】ソースコードの構成

```
<!DOCTYPE html>
<html>
<head>
<title>図書検索</title>
</head>
<body>
<CENTER>
<P><FONT FACE='Verdana' SIZE=6><B>図書貸出し検索</B></FONT></P>
  <%=bal.getResultset() %><br>
<HR WIDTH=300>
<P><A HREF="Library">ログイン画面へ</A></P>
<HR WIDTH=300>
</CENTER>
</body>
</html>
```

実行確認

サーブレットクラス（BookLibraryServlet.java）を実行します。しかし、、、

１．InternalServerErrorとなります

実行用Beanのデプロイ（配置）が必要です。この設定を行わないと図18のような実行時エラーになります。

Eclipseパッケージ・エクスプローラより
BookLibraryServlet.javaを右クリック→実行→サーバーで実行
→図18のように**エラー**となる

図18．実行時エラー画面

２．実行用Beanを配置します

提供された３つのクラスファイルをEclipseの実行時のクラスパスに追加します。

Eclipse**プロジェクト・エクスプローラ**より
（**パッケージ・エクスプローラではありません！**）
booklibプロジェクトを展開→buildを展開→classesを展開→jpを展開→ictを展開→asoを展開→modelがなければ作成

図19のようにmodelフォルダの位置へ提供された３つのクラスファイルをドラッグ＆ドロップすることで実行時クラスパスにBeanが追加されます。
※Eclipseを終了させると、再度追加が必要な場合があります。

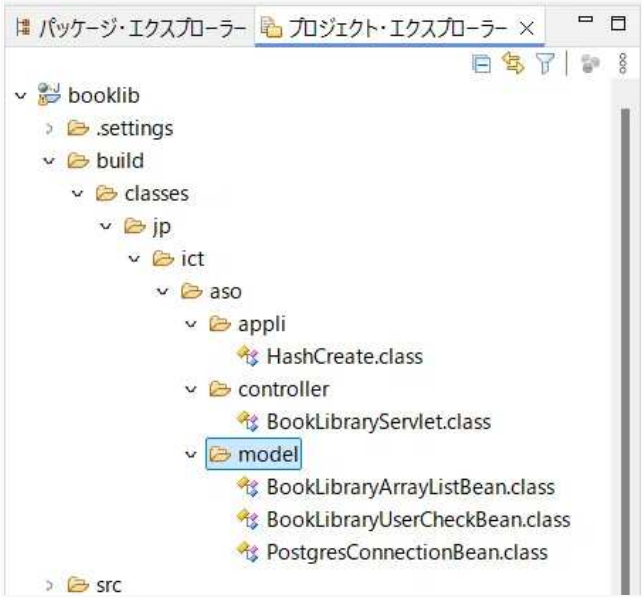

図19．実行時クラスパスの配置

３．再度実行します

Tomcatサーバを再起動したのち、再度サーブレットクラス（BookLibraryServlet.java）を実行します。

Eclipseパッケージ・エクスプローラより
BookLibraryServlet.javaを右クリック→実行→サーバーで実行
→図20、図21のように実行される

図20．ログイン画面

利用者電話番号	氏名	所属	書名	貸出日	返却日
03-1234-5678	山田隆	営業	構造化分析とシステム仕様	1999-12-12	1999-12-13
03-1234-5678	山田隆	営業	構造化分析とシステム仕様	1999-12-12	
03-1234-5678	山田隆	営業	構造化分析とシステム仕様	2001-10-03	2002-10-23
03-1234-5678	山田隆	営業	構造化分析とシステム仕様	2002-10-26	2002-10-25
03-1234-5678	山田隆	営業	構造化分析とシステム仕様	2002-10-26	
03-1234-5678	山田隆	営業	構造化分析とシステム仕様	2002-10-26	
03-1234-5678	山田隆	営業	構造化分析とシステム仕様	2002-10-26	
03-1234-5678	山田隆	営業	WindowsNT4.0システム管理入門（基礎編）	1998-07-07	1998-08-07
03-1234-5678	山田隆	営業	総合化MRPシステム－設計と導入-	1998-07-07	1998-08-07
03-1234-5678	山田隆	営業	わかりやすいデータベース設計技法	2000-02-02	2001-02-02

図21．検索実行結果画面

ソースコード例と提供Bean

以下に各プログラムのソースコードの例（本文内では「考えましょう！」になっている部分）を示しますので、実装の参考にしてください。

また、提供用のBeanとして **DataAccessObject** 関連の３つのクラスを次のセクションに置きますので、ダウンロードして使用してください。

１．PostgresConnectionBean.class

PostgresConnectionBean.class