時計在庫検索システム-EE8,PostgreSQL

office・M
2024年9月14日 18:29

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。今回はログイン処理でユーザの認証を行った後に時計店の在庫状況を検索することができるシステムを実装します。
とりわけ、DBを操作するクラスファイルを実装する方法を学びます（DAOを作成します）。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。

具体的には時計店の在庫検索システムをWebアプリケーションのMVCモデルで作成します。

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。

今回は、これまでの開発の練習を踏まえ、データベースの接続や抽出処理を行うBeanとともにServletやJSPのプログラム全てを自由に実装します。

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。

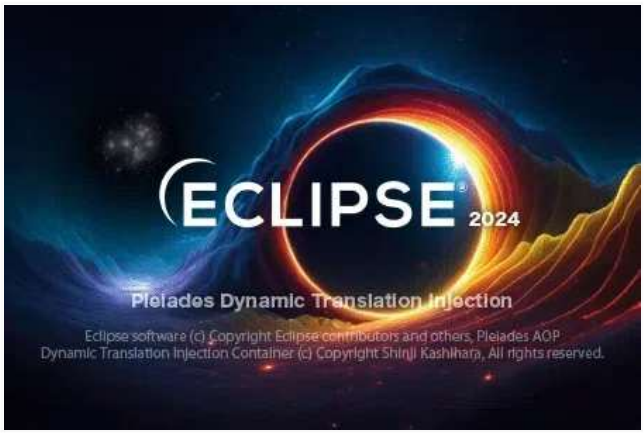

図1．開発環境 Eclipse2024

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。

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

DB設計

論理設計

ユーザテーブル（T-USER）と在庫テーブル（T-ZAIKO）の２つを用意します。ユーザテーブルにはデータベースを検索するユーザのIDとパスワードの情報を格納しログイン処理に用います。ユーザのIDは電話番号としパスワードはハッシュ化して保存します。在庫テーブルには商品在庫情報とともに保管場所の情報も格納します。

図2．ユーザーテーブル（T_USER）レコード形式

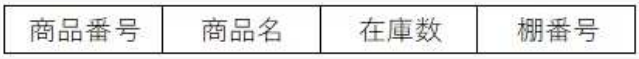

図3．在庫テーブル（T_ZAIKO）レコード形式

物理設計

接続するPostgreSQLサーバのDB名は「**mrp**」とします。

（DB構造,スキーマ）

```
create table T_USER(
  TEL varchar(20) not null,
  PASSWORD varchar(64) not null,
  primary key(TEL)
);
create table T_ZAIKO(
  ITEM_ID varchar(20) not null,
  ITEM_NAME varchar(20) not null,
  ZAIKO_NUM int,
  LOCATION varchar(20),
  PRIMARY KEY (ITEM_ID)
);
```

A5：SQL Mk-2などのSQLクライアントソフトを使って上記のスキーマをDBに構築してください。その後、以下のようなテスト用データを入力しておきます。

（テスト用データ）

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

図4．ユーザテーブル（T_USER）テスト用データ

item_id	item_name	zaiko_num	location
A-001	オメガA1	10	2-1棚
A-002	オメガA2	20	2-2棚
A-003	オメガA3	10	2-1棚
A-004	オメガA4	20	2-2棚
A-005	オメガA5	10	2-1棚
A-006	オメガA6	20	2-2棚
A-007	オメガA7	10	2-1棚
B-001	ゼニスB1	30	3-1棚
B-002	ゼニスB2	40	3-2棚
B-003	ゼニスB3	30	3-2棚
B-004	ゼニスB4	40	3-3棚
C-001	テクノスC1	50	4-1棚
C-002	テクノスC2	60	4-2棚
C-003	テクノスC3	50	4-1棚
C-004	テクノスC4	60	4-2棚
D-001	セイコーD1	70	5-1棚
D-002	セイコーD2	80	5-2棚
D-003	セイコーD3	90	5-3棚

図5．在庫テーブル（T_ZAIKO）テスト用データ

外部設計

接続urlは**http://localhost:8080/zaiko/Inventory**とします。よってEclipseの動的Webプロジェクトの名前は**zaiko**となり、サーブレットのurlパターンは**Inventory**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→「zaiko」プロジェクトを作成する
→図６の内容で設定する

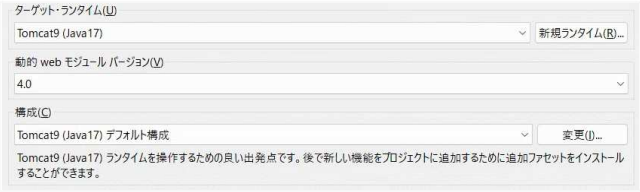

図6．動的Webプロジェクトの設定

※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７のような、ログイン画面が表示されます。電話番号とパスワードで認証し在庫検索を行います。認証がOKの場合のみ図８のような在庫検索画面が表示されます（**認証を経ないと在庫検索画面は表示されません**）。在庫検索はDBから読み込んだ商品番号（商品ID）をもとに**ドロップダウンリストで指定**します。**商品の画像も表示**します。

図7．ログイン画面

図8．在庫検索画面

内部設計

クラスの連携は図９、図10のようになります。ログイン処理用のWebアプリと在庫検索処理用のWebアプリをそれぞれ別々に制作します。両者をセッションでつなぐことで認証済みの場合だけ在庫検索が実行できるような設計にします。

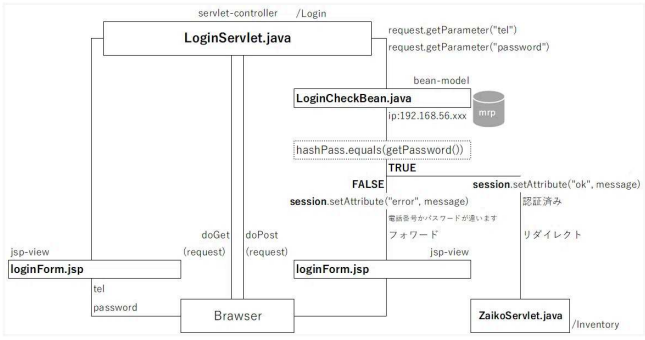

図9．クラス連携図（ログイン処理）

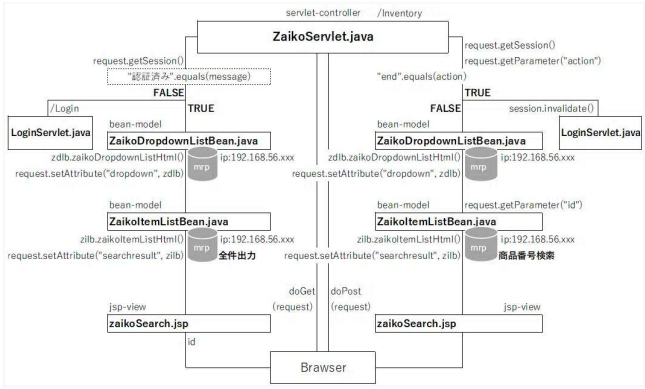

図10．クラス連携図（在庫検索処理）

LoginCheckBean.classの仕様とクラス図

（仕様）

・データベースからTEL番号をキーにハッシュ化されたパスワードを取得します（ハッシュ化されたパスワードは事前にDBに登録されています）。
・コンストラクタの引数でキーとなる電話番号とPostgreSQLサーバのIPとDB名を指定して実体化します。
・コンストラクタの実行により対象となる電話番号のパスワードが取得されます。電話番号が登録されていないときにはパスワードは空となります。
・利用側のクラスはgetPassword()メソッドでパスワードを取得します。

（クラス図）

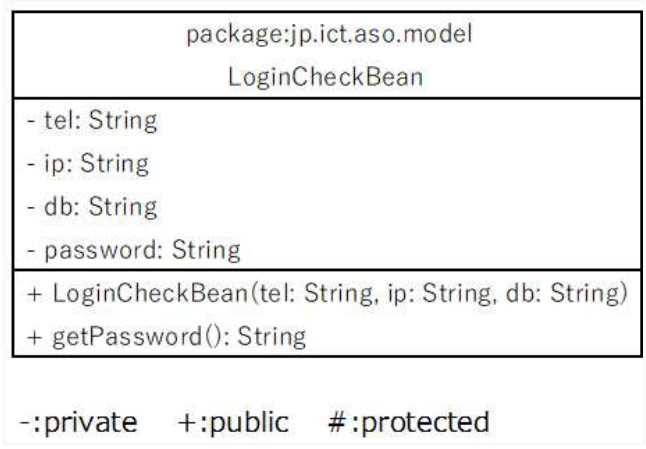

図11．LoginCheckBeanクラス図

ZaikoDropdownListBean.classの仕様とクラス図

（仕様）

・データベースから商品番号のドロップダウンリストを作成します。
・コンストラクタの引数でPostgreSQLサーバのIPとDB名を指定して実体化します。
・zaikoDropdownListHtml()メソッドの実行によりDBから商品番号を取得し、昇順の並べかえを行ってドロップダウンリストのオプション指定の

html文を生成します。
・その際、シリーズごとの出力と全件出力のオプション指定も追加します。
・生成されたhtml文はresultHtmlフィールドに格納されます。
・利用側のクラスはgetResultHtml()メソッドで商品番号リスト（html文）を取得します。

（クラス図）

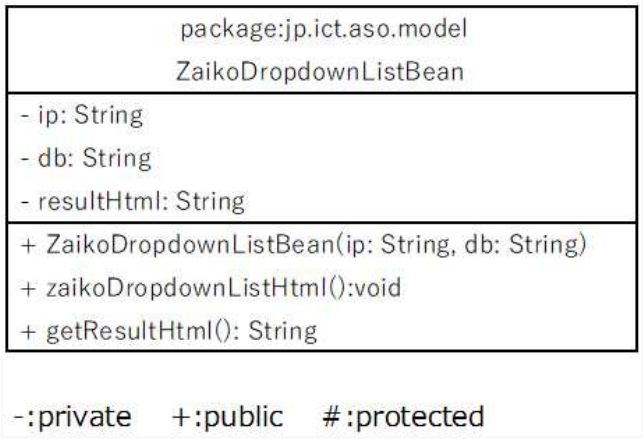

図12．ZaikoDropdownListBeanクラス図

ZaikoItemListBean.classの仕様とクラス図

（仕様）
・データベースから商品番号をキーに商品情報のリストを作成します。
・コンストラクタの引数で商品番号とPostgreSQLサーバのIPとDB名を指定して実体化します。
・zaikoItemListHtml()メソッドの実行によりDBから商品情報を取得し、昇順の並べかえを行ってテーブル整形されたhtml文を生成します。
・その際、シリーズごとの出力や全件出力にも対応させます。
・さらに、時計のイメージ画像の出力にも対応させます。
（時計の画像はimages-watchフォルダ内から読み込みます）
・生成されたhtml文はresultHtmlフィールドに格納されます。
・利用側のクラスはgetResultHtml()メソッドで商品リスト（html文）を取得します。

（クラス図）

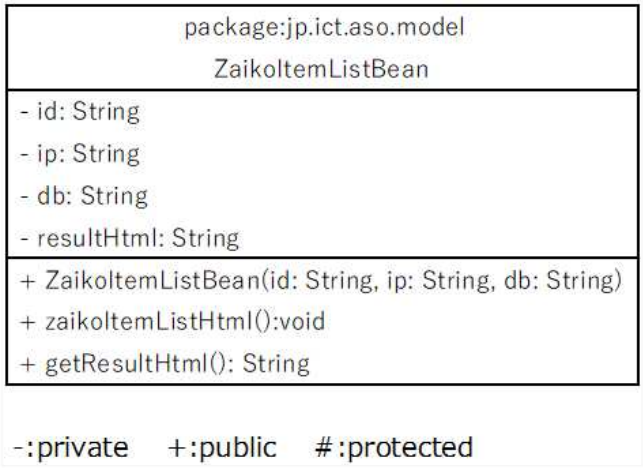

図13．ZaikoItemListBeanクラス図

実装準備

PostgreSQLのjdbcドライバの登録

org.postgresql.driver.jarファイルをパッケージエクスプローラ内の**zaiko**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。

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

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**zaiko**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14．ライブラリの配置位置

実装手順１（ログイン処理）

１．DAO用JavaBeansクラスを作成します

Eclipseパッケージ・エクスプローラより
zaiko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

・LoginCheckBean.java
　　パッケージ：jp.ict.aso.model
　　クラス名：LoginCheckBean
　　ソースコード：以下のヒントと図11を参考に考えましょう！
※try{～}catch(Exception e){e.printStackTrace();}が必要です！！
**　ヒントの全体を囲みましょう。**

【ヒント】

```
/* ●業務処理セクション　　*/
//[1]JDBCドライバのロード
Class.forName("org.postgresql.Driver");

//「2」JRDBへの接続
String url="jdbc:postgresql://"+ip+":5432/"+db;
```

```
Connection connection =DriverManager.getConnection(url,"postgres","postgres");

//[3]SQL文のコンテナ作成
Statement stmt = connection.createStatement();

//[4]SQL文を実行する
ResultSet rs = stmt.executeQuery
         ("select TEL,PASSWORD from T_USER where TEL = '" + tel + "'");

//[5]RDBの検索結果を取り出す
while( rs.next() ){
      tel = rs.getString("TEL" );
      password = rs.getString("PASSWORD" );
}

//[6]データベースから切断する
stmt.close();
connection.close();
```

２．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
zaiko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

・LoginServlet.java
　パッケージ：jp.ict.aso.controller
　クラス名：LoginServlet
　ソースコード：以下のヒントを参考に考えましょう
※アノテーションは**/Login**

```
【ヒント】
doGet()
  // エラーメッセージ(なし)をセッションに設定
  String message="";
  HttpSession session=request.getSession();
  session.setAttribute("error", message);
  // loginForm.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");
```

```
String hashPass = DigestUtils.sha256Hex(pass);//ハッシュ値を求める
// DB接続パラメータの設定
String ip="192.168.56.200";←←←←？？？？？
String db="mrp";
// DB接続・ユーザ検索・ハッシュパスワード取得
LoginCheckBean lcb = new LoginCheckBean(tel,ip,db);
// リクエストパスワードとDB取得パスワードのチェック
if(hashPass.equals(lcb.getPassword())) {
  // 認証OKなら認証済みメッセージをセッションに設定
  String message="認証済み";
  HttpSession session=request.getSession();
  session.setAttribute("ok", message);
  //zaiko検索サーブレットへリダイレクト（フォワードだとpostされるため）
  response.sendRedirect("Inventory");
 }else {
  // 認証NGなら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);
  // loginForm.jsp(アカウント再入力)へフォワード
 }
```

３．アカウント入力用画面の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
zaiko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

・loginForm.jsp
　保存場所：zaiko/src/main/webapp/**WEB-INF/jsp** ← **注意！**
　ファイル名：loginForm.jsp
　ソースコード：以下のヒントを参考に考えましょう！

```
【ヒント】
<!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=320>
```

```
<FORM METHOD="POST" ACTION="Login">
<TABLE border=1>
  <TR>
    <TD width=200>利用者電話番号：</TD>
    <TD width=200 align="left">
      <INPUT TYPE=TEXT NAME="tel" value="03-1234-5678" SIZE=20></TD>
  </TR>
  <TR>
    <TD width=200>パスワード：</TD>
    <TD width=200 align="left">
      <INPUT TYPE=PASSWORD NAME="password" value="0312345678" SIZE=20></TD>
  </TR>
</TABLE>
<INPUT TYPE=SUBMIT VALUE=ログイン><INPUT TYPE=RESET VALUE=リセット>
</FORM>
<HR WIDTH=320>
<p>
<%=message %>
<% session.removeAttribute("error"); %>
</p>
</BODY>
</html>
```

実行確認１（ログイン処理）

サーブレットクラス（LoginServlet.java）を実行します。

１．ログインエラー時の出力を確認します

以下のように利用者電話番号とパスワードの組み合わせを間違えるとエラーメッセージが出力されることを確認します。

図15．ログインエラーの出力例

２．ログイン正常時の状態を確認します

ZaikoServlet.java（/Inventory）を制作していない状態では、当然エラーとなります。エラーメッセージだけ確認します。

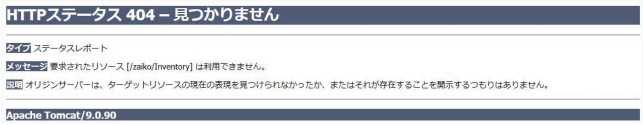

図16．ログイン正常時の出力例

実装手順２（在庫検索処理）

１．DAO用JavaBeansクラスを作成します

Eclipseパッケージ・エクスプローラより
zaiko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

・ZaikoDropdownListBean.java
　パッケージ：jp.ict.aso.model
　クラス名：ZaikoDropdownListBean
　ソースコード：以下のヒントと図12を参考に考えましょう！
※try{～}catch(Exception e){e.printStackTrace();}が必要です！！
**　ヒントの全体を囲みましょう。**

【ヒント】

```
//[1]JDBCドライバのロード
  Class.forName("org.postgresql.Driver");

//「2」RDBへの接続
  String url="jdbc:postgresql://"+ip+":5432/"+db;
  Connection connection =DriverManager.getConnection(url,"postgres","postgres");

//[3]SQL文のコンテナ作成
  Statement stmt = connection.createStatement();

//[4]SQL文を実行する
  ResultSet rs = stmt.executeQuery
    ("select ITEM_ID from T_ZAIKO order by ITEM_ID");

//[5]RDBの検索結果を取り出す
  while (rs.next()) {
    id = rs.getString("ITEM_ID");
    resultHtml=resultHtml+"<option>"+id+"</option>";
  }
```

```
  resultHtml=resultHtml+"<option value='A%'>Aシリーズ</option>";
  resultHtml=resultHtml+"<option value='B%'>Bシリーズ</option>";
  resultHtml=resultHtml+"<option value='C%'>Cシリーズ</option>";
  resultHtml=resultHtml+"<option value='D%'>Dシリーズ</option>";
  resultHtml=resultHtml+"<option value='%'>全て</option>";

//[6]データベースから切断する
  stmt.close();
  connection.close();
```

Eclipseパッケージ・エクスプローラより
zaiko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

・ZaikoItemListBean.java
　パッケージ：jp.ict.aso.model
　クラス名：ZaikoItemListBean
　ソースコード：以下のヒントと図13を参考に考えましょう！
※時計の画像はpng形式を想定しています！！
※try{～}catch(Exception e){e.printStackTrace();}が必要です！！
**　ヒントの全体を囲みましょう。**

【ヒント】

```
//[1]JDBCドライバのロード
  Class.forName("org.postgresql.Driver");

//「2」RDBへの接続
  String url="jdbc:postgresql://"+ip+":5432/"+db;
  Connection connection =DriverManager.getConnection(url,"postgres","postgres");

//[3]SQL文のコンテナ作成
  Statement stmt = connection.createStatement();

//[4]SQL文を実行する
  ResultSet rs = stmt.executeQuery
    ("select ITEM_ID,ITEM_NAME,ZAIKO_NUM,LOCATION from T_ZAIKO where ITEM_ID like '"
    + id + "' order by ITEM_ID");

//[5]RDBの検索結果を取り出す
  resultHtml=resultHtml+"<table border=1><tr>";
  resultHtml=resultHtml+"<tr><th>No.</th><th>商品ID</th><th>商品イメージ</th>"
    + "<th>商品名</th>";
  resultHtml=resultHtml+"<th>在庫数</th><th>保管棚</th></tr>";
  int no=0;
  while( rs.next() ){
```

```
        no++;
        id = rs.getString("ITEM_ID");
        name = rs.getString("ITEM_NAME" );
        zaiko = rs.getInt( "ZAIKO_NUM" );
        location = rs.getString( "LOCATION" );
        resultHtml=resultHtml+"<tr><td>"+no+"</td>";
        resultHtml=resultHtml+"<td>"+id+"</td>";
        resultHtml=resultHtml+"<td><img src='watch-images/"+id+".png' width=100></td>";
        resultHtml=resultHtml+"<td>"+name+"</td>";
        resultHtml=resultHtml+"<td>"+zaiko+"</td>";
        resultHtml=resultHtml+"<td>"+location+"</td></tr>";
      }
      resultHtml=resultHtml+"</table>";

    //[6]データベースから切断する
      stmt.close();connection.close();
```

２．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
zaiko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

・ZaikoServlet.java
　パッケージ：jp.ict.aso.controller
　クラス名：ZaikoServlet
　ソースコード：以下のヒントを参考に考えましょう
※アノテーションは**/Inventory**

```
【ヒント】
doGet()
  //認証済みかhttpセッションの内容で確認
  HttpSession session=request.getSession();
  String message=(String)session.getAttribute("ok");
  if("認証済み".equals(message)) {
    // ●ドロップダウンリストの実体化（IP,DB名の初期値をプロパティに設定）
    ZaikoDropdownListBean zdlb = new ZaikoDropdownListBean("192.168.56.200","mrp");
    // ドロップダウンリストを作成
    zdlb.zaikoDropdownListHtml();
    // リクエストスコープに保存（スコープ名：dropdown）
    request.setAttribute("dropdown", zdlb);
    // ●検索結果リストの実体化（商品番号の初期値、IP、DB名をプロパティに設定）
    ZaikoItemListBean zilb = new ZaikoItemListBean("%","192.168.56.200","mrp");
```

```
        // 在庫リストを作成
        zilb.zaikoItemListHtml();
        // リクエストスコープに保存（スコープ名：searchresult）
        request.setAttribute("searchresult", zilb);
        // zaikoSearch.jsp(入出力用JSP)にフォワード

      }else {
      // 認証されていない場合はアカウント入力に戻す
      // getリクエストで転送するためredirectでも良い
        RequestDispatcher dispatcher =
           request.getRequestDispatcher("Login");
           dispatcher.forward(request, response);
      }
   doPost()
      //セッションとリクエストを取得してログアウトボタンが押されたか確認
      HttpSession session=request.getSession();
      String action=(String)request.getParameter("action");
      if("end".equals(action)) {    //actionがend（ログアウト）のとき
        // セッション破棄
        session.invalidate();
        // ログインページにリダイレクト
        // getリクエストで転送したいためredirectにする
        response.sendRedirect("Login");

      }else {                       //actionがviewのとき
        // ●ドロップダウンリストの実体化（IP,DB名の初期値をプロパティに設定）
        ZaikoDropdownListBean zdlb = new ZaikoDropdownListBean("192.168.56.200","mrp");
        // ドロップダウンリストを作成
        zdlb.zaikoDropdownListHtml();
        // リクエストスコープに保存（スコープ名：dropdown）
        request.setAttribute("dropdown", zdlb);
        // リクエストパラメータ（入力値）の取得（パラメータ名：id）
        request.setCharacterEncoding("UTF-8");
        String id = request.getParameter("id");
        // ●検索結果リストの実体化（商品番号の初期値、IP、DB名をプロパティに設定）
        ZaikoItemListBean zilb = new ZaikoItemListBean(id,"192.168.56.200","mrp");
        // 在庫リストを作成
        zilb.zaikoItemListHtml();
        // リクエストスコープに保存（スコープ名：searchresult）
        request.setAttribute("searchresult", zilb);
        // zaikoSearch.jsp(入出力用JSP)にフォワード
      }
```

３．入力・検索結果リスト出力用画面の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
zaiko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

・zaikoSearch.jsp

保存場所：zaiko/src/main/webapp/**WEB-INF/jsp ← 注意！**

ファイル名：zaikoSearch.jsp

ソースコード：以下のヒントを参考に考えましょう！

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】

```
<%
String message=(String)session.getAttribute("ok");
%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>商品在庫取得</title>
 </head>
<body><center>
  <h1>在庫検索システム</h1>
    <p><font face="Verdana" size="5"><b>Inventory Search</b></font></p>
  <hr width="360">
  <table border="0">
  <tr>
   <form action="Inventory?action=view" method="post">
     <td>商品ID:
     <select name="id">
     <%= zdlb.getResultHtml() %>
     </select>
     </td>
     <td><input type="submit" value="在庫状況取得"></td>
   </form>
   <td><%= message %></td>
   <td>
   <form action="Inventory?action=end"  method="post">
     <input type="submit" name="logout" value="ログアウト">
   </form>
   </td>
  </tr>
  </table>
  <%= zilb.getResultHtml() %>
<!-- フッター -->
    <hr width="360">
    <footer>
        <p>&copy; 2024 在庫検索システム. All Rights Reserved.</p>
        <p>お問い合わせ: info@slowlife.halfmoon.jp</p>
    </footer>
</body>
</html>
```

実行確認２（在庫検索処理）

１．ログインシステムから在庫検索システムが正常に実行されることを確認します

・図17のようにログインシステム（LoginServlet.java）を実行します。正常に認証されたとき在庫検索画面が表示されるか確認します。（/Loginが実行されて認証がOKの場合は/Inventoryへ遷移するはずです。）
・あわせて「在庫状況取得」と「ログアウト」のボタンの動きも確認します。
※時計の画像は後で設定しますのでこの段階では表示されません！！

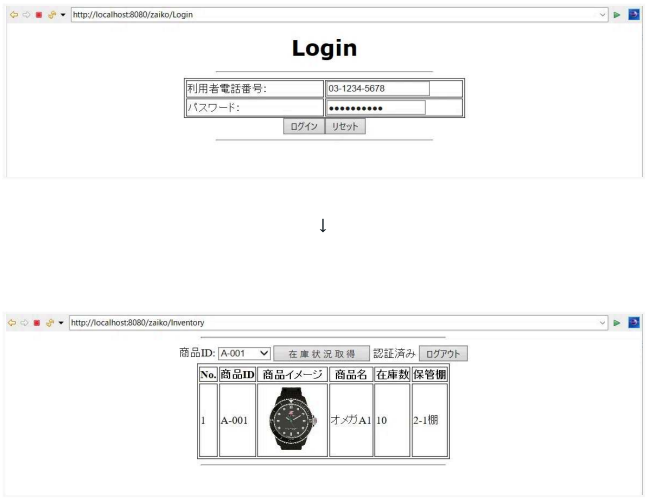

図17．ログインシステムから在庫検索システムへ遷移

２．ログアウト時に在庫検索システムの直接起動ができないことを確認します

・図18のように**ログアウトしたとき**に/Inventoryのurl直接指定でログイン状態になることを確認します（ログインしていない状態で/Inventoryの内容が表示されてはいけません）。

図18．在庫検索システムの直接実行はログインシステムへ遷移

３．スタイルシートや画像及びヘッター・フッターで表示を装飾します

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

・zaiko.css
　保存場所：zaiko/src/main/**webapp/** ← **注意！**
　ファイル名：zaiko.css
　ソースコード：考えましょう！

【参考】フリー画像サイト↓
https://pixabay.com/illustrations/search/

・watgch-images/時計画像やシンボル画像など
　画像を配置するwatch-imagesフォルダの位置は以下の場所です。

図19．画像ファイルの位置

最終確認

装飾の状態、ボタンの動き、２システム（/Loginと/Inventory）間の連携など全て問題なく動くか確認して完成です。

図20．時計在庫検索システム完成

仕様拡張

時計を販売したときに在庫数を減らす、また入荷したとき在庫数を増やす処理を加えることにより在庫管理システムとしての機能をもたせます。

以下の例のように［販売］と［入荷］のボタンを追加するようシステムの仕様を変更します。詳細については別の回で紹介します。

図21．仕様変更後の時計在庫管理システム