Visualizzazione ordini in MyShopDB

- Risolviamo gli esercizi della lezione scorsa, scrivendo una pagina **`ordini.jsp`** che visualizzi le intestazioni di ogni ordine (numero d'ordine, data, nome, cognome, indirizzo, pagamento, consegna, spese di spedizione).
- Facciamo inoltre in modo che, cliccando su un apposito link, sia possibile visualizzare il dettaglio dell'ordine (gli articoli ordinati dal cliente).

La query

- La query da utilizzare è:

```
SELECT ID, Data, Nome, Cognome,
Indirizzo, Pagamento, Consegna,
SpeseSpedizione FROM
intestazioni_ordini
```

- Aggiungiamo la clausola finale **`ORDER BY Data DESC`** per ottenere la lista degli ordini a partire dal più recente a quello più vecchio.

L’idea di base

- L’idea di base è di creare un’unica pagina **`ordini.jsp`** che visualizzi (a seconda del valore di un parametro) soltanto le intestazioni oppure anche il dettaglio delle righe dell’ordine selezionato.
- Il parametro sarà passato attraverso il metodo GET di HTTP (nome del parametro: **`id`**).
- Esempio: **`ordini.jsp?id=5`** visualizzerà le intestazioni di tutti gli ordini ed il dettaglio delle righe dell’ordine n. 5.

La query per le righe

- Quindi l'interrogazione corretta per estrarre le righe corrispondenti all'ordine passato come parametro è la seguente:

```
"SELECT P.Prodotto, R.Prezzo,
R.Quantita FROM prodotti AS P,
righe_ordini AS R WHERE
P.ID=R.IDProdotto AND
R.IDOrdine="+idOrdine
```

- Dove **`idOrdine`** è una variabile inizializzata con

```
idOrdine=request.getParameter("id");
```

ordini.jsp (I)

```
...
<%if(session.getValue("myShop.backofficeUser")==null)
        response.sendRedirect("backoffice.jsp");
  String idOrdine=request.getParameter("id");
  try {
        Integer.parseInt(idOrdine);
  } catch (NullPointerException e) {
        idOrdine="0";
  } catch (NumberFormatException e) {
        idOrdine="0";
  }
  Vector ordini=new Vector();
  boolean nessunOrdine=true;
```

Se l'utente non si è autenticato, lo rimandiamo alla pagina di login dell'area riservata (più avanti nei lucidi)

Recupero e controllo del parametro id: contiene il numero d'ordine di cui visualizzare il dettaglio delle righe (se è vuoto, si visualizzano soltanto le intestazioni).

ordini.jsp (II)

```
try {
  Connection c = DriverManager.
   getConnection(stringaConnessione, utenteSQL, passwordSQL);
  Statement s = c.createStatement();
  ResultSet r = s.executeQuery("SELECT ID, Data, Nome, Cognome,
   Indirizzo, Pagamento, Consegna, SpeseSpedizione FROM
   intestazioni_ordini ORDER BY Data DESC");

  while(r.next()) {
    Ordine o=new Ordine(r.getInt(1), r.getDate(2),
      r.getString(3), r.getString(4), r.getString(5),
      r.getString(6), r.getString(7), r.getFloat(8));
    Statement s2 = c.createStatement();
    ResultSet r2=s2.executeQuery("SELECT P.ID, P.Prodotto,
P.Descrizione, R.Prezzo, R.Quantita FROM prodotti AS P,
righe_ordini AS R WHERE P.ID=R.IDProdotto AND
R.IDOrdine="+o.numero);
```

Estrazione degli ordini (intestazioni)

Estrazione delle righe dell'ordine corrente

ordini.jsp (III)

```
 while(r2.next()) {
   Prodotto p=new Prodotto(r2.getInt(1), r2.getString(2),
                           r2.getString(3), r2.getFloat(4));
   o.AggiungiRiga(p, r2.getInt(5)); // Aggiunta della riga
 }

 r2.close(); s2.close();
 ordini.add(o);  // Aggiunta dell'ordine al Vector ordini
}

r.close();
nessunOrdine=ordini.size()==0;
s.close();
c.close();
...
```

ordini.jsp (IV)

```
 for(int i=0; i<ordini.size(); i++) {
   String stile=(i+1)%2==0 ? "riga2_tabella" : "riga1_tabella";
   Ordine o=(Ordine)ordini.elementAt(i);
   boolean testId=Integer.parseInt(idOrdine)==o.numero;
%>
    <tr>
     <td class='<%= stile%>'><%= o.numero%></td>
     <td class='<%= stile%>'><%= o.data%></td>
     ...
     <td class='<%= stile%>'>
      <a href="ordini.jsp?id=<%= !testId ? o.numero : 0%>">
        <%= !testId ? "dettagli" : "nascondi dettagli"%>
      </a>
     </td>
    </tr>
<%
 // codice per le righe
 }
```

Visualizzazione dei vari campi dell'intestazione dell'ordine

Link che richiama la pagina stessa passando come parametro (modalità GET di HTTP) il numero d'ordine di cui visualizzare il dettaglio delle righe.

ordini.jsp (V): codice per le righe

Visualizzazione delle righe dell'ordine
In una sotto-tabella della tabella delle intestazioni

```
<%if(testId) {%>
...
<%...
  for(int j=0; j<o.NumeroRighe(); j++) {
    String stile2=(j+1)%2==0 ? "riga2_sotto_tabella" :
                               "riga1_sotto_tabella";
    ProdottoSelezionato ps=o.TrovaRiga(j);%>
    <tr>
      <td class='<%= stile2%>'><%= ps.nome%></td>
      <td class='<%= stile2%>'><%= fmt.format(ps.prezzo)%></td>
      <td class='<%= stile2%>'><%= ps.quantitaSelezionata%></td>
      <td class='<%= stile2%>'>
        <%= fmt.format(ps.prezzo*ps.quantitaSelezionata)%></td>
    <tr>
 <%}%>
```

Proteggiamo l'accesso a `ordini.jsp`

- La pagina degli ordini non dovrebbe essere pubblica (per ovvi motivi di sicurezza e privacy), ma accessibile soltanto all'amministratore di MyShopDB.
- Dobbiamo quindi proteggerla realizzando una piccola area riservata con un meccanismo di autenticazione.
- Sfruttiamo quindi la tabella **`utenti_backoffice`** che contiene un utente per ogni record (composto dai campi **`Username`** e **`Password`**).

Controllo dell'avvenuta autenticazione

- Sfruttiamo una variante della classe Utente, già utilizzata a suo tempo con le servlet, per memorizzare nella sessione degli utenti che sono riusciti ad autenticarsi un oggetto di tale tipo.
- In questo modo potremo distinguere fra accessi legittimi o meno in base alla presenza di tale oggetto nella sessione (ed agire di conseguenza con il metodo **`sendRedirect()`** dell'oggetto predefinito **`response`**).

backoffice.jsp (I)

```
...
<%if(session.getValue("myShop.backofficeUser")!=null)
        response.sendRedirect("ordini.jsp");

  //recupero del nome utente dal form
  String username=request.getParameter("user");
  if(username==null)
        username="";
  //recupero della password dal form
  String password=request.getParameter("pwd");
  if(password==null)
        password="";

  // flag che rappresenta lo stato dell'utente
  // default: non autenticato
  boolean autenticato=false;
```

Se l'utente si è già autenticato, lo inviamo alla pagina degli ordini.

backoffice.jsp (II)

```
 try {
   Connection c = DriverManager.getConnection(stringaConnessione, utenteSQL,
   passwordSQL);
   Statement s = c.createStatement();
   ResultSet r = s.executeQuery("SELECT COUNT(*) FROM utenti_backoffice WHERE
                                 Username='"+CodificaApici(username)+"' AND
                                 Password='"+CodificaApici(password)+"'");
   if(r.next()) {
     autenticato=r.getInt(1)==1;
   }
   ...
} catch (SQLException e) { response.sendRedirect("errore.jsp");}

if(autenticato) {
  Utente u=new Utente(username, password, "Amministratore");
  session.putValue("myShop.backofficeUser", u);
  response.sendRedirect("ordini.jsp");
}
%>
```

backoffice.jsp (III)

```
<%
  if(!autenticato && username.length()>0) {
%>
   <div class="errore"> </div>
   <div class="errore">Nome utente e/o password non sono
     corretti!<br><br><a href="javascript:window.history.go(-1)">
     Riprova.</a></div>
   <div class="errore"> </div>
<%
  }
  else {
    // form di autenticazione (vedi lucido seguente)
  }
%>
```

Messaggio d'errore: l'utente ha sbagliato nome utente e/o password

backoffice.jsp (IV): form di autenticazione

```
<form method="post" action="backoffice.jsp">
   <table>
     <tr>
       <td class="intestazione_form">Nome utente: </td>
       <td><input name="user" type="text" size="10"></td>
     </tr>
     <tr>
       <td class="intestazione_form">Password: </td>
       <td><input name="pwd" type="password" size="10"></td>
     </tr>
     <tr>
       <td colspan="2"><input type="submit" Value="Login &gt;&gt;">
       </td>
    </tr>
  </table>
</form>
```

I file JAR e JSP

- La classe **`Utente`** è stata inserita nel file **`myShop.jar`**.

```
package MyShop;
public class Utente {
      String username;
      String password;
      String descrizione;
      public Utente(String u, String p, String d) {
              username = u; password = p; descrizione = d;
      }
}
```

- L'archivio JAR va quindi copiato nella directory **`/home/<nome-utente>/servlets/WEB-INF/lib`** e importato con **`<%@ page import="myShop.*" %>`**

Esercizio

- Modificare la pagina **`ordini.jsp`** aggiungendo la possibilità di visualizzare contemporaneamente le righe di tutti gli ordini.
- Aggiungere inoltre la possibilità di cancellare un ordine (intestazione e righe).

Riferimenti (I)

- David Flanagan, Jim Farley, William Crawford
  Java Enterprise in a Nutshell (2nd Edition)
  O'Reilly 2002
  - Capitoli 1, 2, 5, 6
  - SQL Reference Manual: cap. 12
  - **`java.sql`** Reference Manual: cap. 23
  - **`javax.servlet`**, **`javax.servlet.http`**, **`javax.servlet.jsp`** Reference Manual: cap. 36-38.

Riferimenti (II)

- Wrox team

  Professional Java Server Programming – J2EE 1.3 Edition

  Wrox 2001

  – Capitoli: 1 (introduzione a J2EE), 7-11
  – Approfondimenti su JDBC: cap. 3 (fino a “Mapping SQL Types to Java”, pag. 125)
  – Approfondimenti sulla “manutenzione” di applicazioni web in JSP: cap. 14