

Vogliamo realizzare una pagina web che ci consenta di filtrare i record della tabella books del database MySQL shop.

	book_id	title	author	price	publisher	category_id
<input type="checkbox"/> <input type="text"/> <input type="text"/>	1	Pro CSS and HTML Design Patterns	Michael Bowers	44.99	Apogeo	1
<input type="checkbox"/> <input type="text"/> <input type="text"/>	2	Pro PayPal E-Commerce	Damon Williams	59.99	Mondadori	1
<input type="checkbox"/> <input type="text"/> <input type="text"/>	3	The Complete Robot	Isaac Asimov	8.95	Urania	2
<input type="checkbox"/> <input type="text"/> <input type="text"/>	4	Foundation	Isaac Asimov	8.95	Urania	2
<input type="checkbox"/> <input type="text"/> <input type="text"/>	5	Area 7	Matthew Reilly	5.99	Apogeo	3
<input type="checkbox"/> <input type="text"/> <input type="text"/>	6	Term Limits	Vince Flynn	6.99	Mondadori	3
<input type="checkbox"/> <input type="text"/> <input type="text"/>	7	2001 - A Space Odyssey	Arthur C. Clarke	20	Urania	2

Fig. 1 – Tabella books, database MySQL shop

Il filtro viene scritto direttamente dall'utente in una textarea e i risultati visualizzati di seguito.

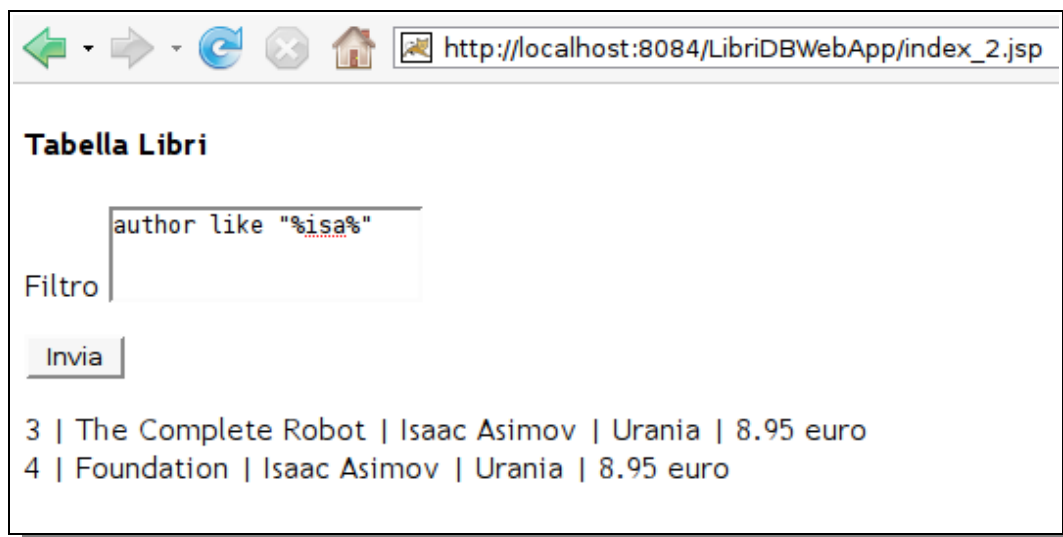


Fig. 2 – Visualizziamo i libri di Isaac Asimov

Potremmo procedere, come abbiamo fatto molte altre volte (per esempio come descritto nell'articolo *"Una semplice web application con database MySQL"*, categoria *Java-Applicazioni web*, file *tabella\_books.pdf*), nella realizzazione di una pagina jsp che, tramite JDBC, interagisca con la tabella books per leggere le informazioni in essa contenute.

Questo approccio destrutturato è però sconsigliabile. Infatti, se optiamo per Java, vogliamo anche utilizzare le sue caratteristiche orientate agli oggetti al fine di realizzare un software quanto più modulare ed espandibile possibile. Con questo approccio riusciamo inoltre a mascherare anche i dettagli di accesso alla base dati e a separare la parte di *presentazione* dalla parte di *modellazione* dei dati (per maggiori informazioni, consigliamo la lettura dell'articolo *"MVC, paradigma Model, View, Controller"*, categoria *Java-Applicazioni web*, file *MCV.pdf*).

Decidiamo quindi di realizzare le classi *Libro* e *FiltroLibriDB*. Libro modella un record della tabella books e FiltroLibriDB provvede a filtrare i record restituendo i risultati in una "collection" (arraylist) che possiamo facilmente scorrere.

```
/*
 * Libro.java
 *
 * Created on 25 febbraio 2008, 20.08
 *
 */

package shopPkg;

/**
 *
 * @author maurizio
 */
public class Libro {
    private int book_id;
    private String title;
    private String author;
    private double price;
    private String publisher;

    /** Creates a new instance of Libro */
    public Libro() {
    }

    // costruttore con più argomenti
    public Libro( int book_id, String title, String author, double price, String publisher)
    {
        this.book_id = book_id;
        this.title = title;
        this.author = author;
        this.price = price;
        this.publisher = publisher;
    }

    // seguono getter e setter
    public int getBook_id() {
        return book_id;
    }

    public void setBook_id(int book_id) {
        this.book_id = book_id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}
```

```
public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}

public String getPublisher() {
    return publisher;
}

public void setPublisher(String publisher) {
    this.publisher = publisher;
}

} // fine classe Libro

/*
 * ListaLibriDB.java
 *
 * Created on 25 febbraio 2008, 19.45
 *
 */

package shopPkg;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

/**
 *
 * @author maurizio
 */
public class FiltroLibriDB {
    Connection conn = null; // rappresenta la connessione al DB
    ArrayList<Libro> libri = null; // struttura dati di appoggio
```

```
/**
 * Creates a new instance of ListaLibriDB
 */

public FiltroLibriDB() throws ClassNotFoundException, SQLException {
    connect();
} // fine costruttore

// connessione al database MySQL shop
public void connect() throws ClassNotFoundException, SQLException {
    try {
        Class.forName("com.mysql.jdbc.Driver");
    } catch(ClassNotFoundException e) {
        throw new ClassNotFoundException("Codice di errore #50: Non sono stati
trovati i driver di MySQL");
    }

    try {
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/shop?
user=root&password=coz123");
    } catch(SQLException e) {
        throw new SQLException("Codice di errore #60: E' impossibile stabilire una
connessione al database");
    }
} // fine metodo connect()

// chiusura della connessione
public void closeConn() throws SQLException {
    try {
        // chiudo la connessione
        conn.close();
    } catch(SQLException e) {
        throw new SQLException("Codice di errore #100: Impossibile effettuare le
operazioni");
    }
}

// caricamento dei dati "filtrati" dalla tabella books nell'arraylist libri
public void caricaLibri(String sql) throws SQLException {
    libri = new ArrayList<Libro>(); // struttura dati di appoggio

    try {
        // creo lo statement che rappresenta l'istruzione SQL
        Statement st = conn.createStatement();
        // trovo tutti i record della tabella libri che soddisfano al criterio
        ResultSet rs = st.executeQuery("SELECT * FROM books WHERE "+sql);

        // per ogni record
        while (rs.next()) {
            int book_id = rs.getInt(1);

```

```
        String title = rs.getString(2);
        String author = rs.getString(3);
        double price = rs.getDouble(4);
        String publisher = rs.getString(5);
        // istanzio un oggetto di tipo Libro
        Libro l = new Libro();
        // setto le proprietà dell'oggetto l con i dati prelevati dal database
        l.setBook_id(book_id);
        l.setTitle(title);
        l.setAuthor(author);
        l.setPrice(price);
        l.setPublisher(publisher);
        // aggiungo un libro
        libri.add(l);
    }
    // chiudo il ResultSet e lo statement
    rs.close();
    st.close();

    } catch(SQLException e) {
        throw new SQLException("Codice di errore #101: "+e.getMessage());
    }
} // fine metodo caricaLibri

// restituisco l'elenco dei libri caricati
public ArrayList<Libro> getElencoLibri() {
    return libri;
}

// restituisce il numero dei libri
public int numLibri() {
    return libri.size();
}

// restituisce il libro di posto i
public Libro getLibro(int i) {
    return libri.get(i);
}
} // fine classe FiltroLibriDB
```

*File index\_2.jsp*

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page errorPage = "gestioneErrori.jsp" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Accesso al database Shop</title>
    <style type="text/css">
      body,td,th {
        font-family: Trebuchet MS;
        font-size: 16px;
      }
    </style>
  </head>
  <body>
    <h4>Tabella Libri</h4>

    <form name="provaFrm" action="{request.requestURI}" method="post">
      <p>Filtro <textarea name="filtroTxt" value="{param.filtroTxt}"></textarea>
      <p><input type="submit" name="inviaBtn" value="Invia"/></p>
    </form>

    <% // se l'utente ha confermato il filtro cliccando sul tasto Invia
      if (request.getParameter("filtroTxt") != null) {
    %>
      <!-- istanzio un javabean cioè un oggetto di tipo FiltroLibriDB -->
      <!-- equivale al codice -->
      <!-- shopPkg.FiltroLibriDB libri = new shopPkg.FiltroLibriDB(); -->
      <jsp:useBean id="libri" class="shopPkg.FiltroLibriDB"/>

    <%
      // recupero il contenuto del filtro
      String filtro = request.getParameter("filtroTxt");

      // se l'utente non ha scritto nulla nella textarea
      // impostiamo il filtro a vero=1
      if (filtro.equals(""))
        filtro = "1";
```

```
// carichiamo in memoria i libri che soddisfano al criterio
libri.caricaLibri(filtro);
// se ci sono libri
if (libri.numLibri()!=0) {
%>
  <!-- ciclo di lettura e visualizzazione dei dati dei libri usando JSTL -->
  <!-- il metodo getElencoLibri restituisce una "collection" (arraylist) -->
  <!-- di oggetti di tipo Libro che facilmente possiamo scorrere -->
  <!-- nell'EL (Expression Language) non si scrive il get -->

  <c:forEach var="book" items="{libri.elencoLibri}">
    <c:out value="{book.book_id}"/> |
    <c:out value="{book.title}"/> |
    <c:out value="{book.author}"/> |
    <c:out value="{book.publisher}"/> |
    <c:out value="{book.price}"/> euro<br/>
  </c:forEach>

  <!-- oppure possiamo scrivere -->
  <!-- for (int i=0; i<libri.numLibri(); i++)
    out.println(libri.getLibro(i).getBook_id()+" | "
      +libri.getLibro(i).getAuthor()+" | "
      +libri.getLibro(i).getTitle()+" | "
      +libri.getLibro(i).getPublisher()+" | "
      +libri.getLibro(i).getPrice()+" euro<br/>");
  -->

  <!-- oppure ancora -->
  <!-- for (shopPkg.Libro l : libri.getElencoLibri())
    out.println(l.getBook_id()+" | "
      +l.getAuthor()+" | "
      +l.getTitle()+" | "
      +l.getPrice()+" | "
      +l.getPublisher()
      +l.getPrice()+" euro<br/>");
  -->
  <%
  } else {
    // non ci sono libri che soddisfano al criterio indicato
    out.println("Non ci sono libri che soddisfano al criterio di ricerca");
  } // fine if numLibri

  // chiudo la connessione
  libri.closeConn();

  } // fine if request
%>

</body>
</html>
```

*File gestioneErrori.jsp*

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page isErrorPage = "true" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Gestione errori</title>
    <style type="text/css">
      body,td,th {
        font-family: Trebuchet MS;
        font-size: 16px;
      }
    </style>
  </head>
  <body>

    <h3>Gestione errori</h3>

    Siamo spiacenti, si è verificato un errore durante l'esecuzione: <br/>
    <%= exception.getMessage()%>
  </body>
</html>
```