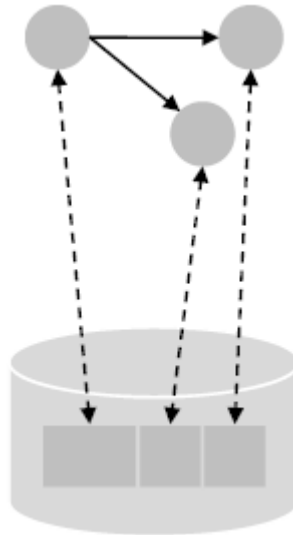


(liberamente adattato da www.db4o.com)

db4o (database for objects)

db4o (www.db4o.com) è un *database* open source che consente agli sviluppatori Java e .NET di rendere *persistenti* gli oggetti (cioè salvarne lo stato su un supporto esterno) di un applicativo con una "sola riga di codice", non importa quanto complessa sia la struttura degli oggetti, perchè li memorizza esattamente nel modo in cui sono definiti dall'applicazione, eliminando la necessità di definire e mantenere un modello dati rigido e separato (come accade in un *database relazionale*). Il modello a oggetti infatti è creato e aggiornato "on demand" da db4o stesso durante la transazione.



Primi passi

L'utilizzo di db4o è piuttosto semplice: si scarica una singola libreria disponibile all'indirizzo http://developer.db4o.com/files/default.aspx#db4o_Database_Engine (una dll o un file jar, nel nostro caso il file jar *db4o-6.4.14.8131-java5.jar*, contenuto nel file *db4-6.4-java.zip*), si crea un file di database vuoto e si memorizza qualsiasi oggetto mediante chiamate a metodi.

Se intendiamo realizzare un'applicazione console, possiamo mettere il file jar *db4o-6.4.14.8131-java5.jar* in un package e importare il package o, se dobbiamo realizzare un'applicazione web, è sufficiente copiare il file jar all'interno della cartella *lib*, sottocartella di *WEB-INF*. Cominciamo col creare una classe di supporto di prova che modella un *pilota* di Formula 1 (F1). Questa classe come si può notare non contiene nessuna istruzione db4o. Gli attributi sono semplicemente il nome del pilota e il suo punteggio.

```
public class Pilot {
    private String name;
    private int points;

    public Pilot(String name,int points) {
        this.name=name;
        this.points=points;
    }

    public int getPoints() {
        return points;
    }

    public void addPoints(int points) {
        this.points+=points;
    }

    public String getName() {
        return name;
    }

    public String toString() {
        return name+"/"+points;
    }
} // fine classe Pilot
```

Apertura del database

Per accedere o creare il database, chiamiamo il metodo `Db4o.openFile("percorso/nomeDB")` e otteniamo una istanza di tipo `ObjectContainer`.

```
ObjectContainer db = Db4o.openFile("f1.yap");
```

Se il file di database non esiste, ne verrà creato uno nuovo, se il file di database esiste, verrà aperto. Possiamo comunque gestire con `try/catch` tutte le situazioni di errore.

Memorizzazione degli oggetti

Per memorizzare un oggetto, usiamo il metodo `set()` dell'oggetto `db`, come nel seguente esempio:

```
// memorizza il primo pilota
Pilot pilot1=new Pilot("Michael Schumacher",100);
db.set(pilot1);
System.out.println("Stored "+pilot1.toString());
```

OUTPUT:

```
Stored Michael Schumacher/100
```

```
// memorizza il secondo pilota
Pilot pilot2=new Pilot("Rubens Barrichello",99);
db.set(pilot2);
System.out.println("Stored "+pilot2.toString());
```

OUTPUT:

```
Stored Rubens Barrichello/99
```

Recupero degli oggetti

`db4o` supporta tre differenti tipi di query: *Query by Example (QBE)*, *Native Queries (NQ)* e *SODA Query API (SODA)*. Useremo una *QBE* e il metodo statico `listResult()`. I risultati vengono restituiti come oggetti di tipo `ObjectSet`.

```
public static void listResult (ObjectSet result){
    System.out.println(result.size());
    while(result.hasNext()) {
        System.out.println(result.next());
    } // fine while
} // fine metodo listResult
```

Per recuperare i due piloti, forniamo un "prototipo" vuoto:

```
// ritrova tutti i piloti mediante QBE
// lo 0 non significa i piloti con punteggio 0
Pilot proto=new Pilot(null,0);
ObjectSet result=db.get(proto);
listResult(result);
```

OUTPUT:

```
2
Michael Schumacher/100
Rubens Barrichello/99
```

db4o fornisce anche una scorciatoia per ottenere tutte le istanze di una classe:

```
// recupera la lista dei piloti
ObjectSet result=db.get(Pilot.class);
listResult(result);
```

OUTPUT:

```
2
Michael Schumacher/100
Rubens Barrichello/99
```

oppure per il JDK 5 usando il metodo query:

```
List <pilots> = db.query(Pilot.class);
```

Se vogliamo trovare un pilota mediante il nome:

```
// ritorna il pilota conoscendo il nome
Pilot proto=new Pilot("Michael Schumacher",0);
ObjectSet result=db.get(proto);
listResult(result);
```

OUTPUT:

```
1
Michael Schumacher/100
```

oppure i piloti con un determinato numero di punti:

```
// ritorna l'elenco dei piloti con un determinato punteggio
Pilot proto=new Pilot(null,100);
ObjectSet result=db.get(proto);
listResult(result);
```

OUTPUT:

```
1
Michael Schumacher/100
```

Aggiornamento

Per aggiornare un oggetto, si ha a disposizione lo stesso metodo set() che si usa per salvare; dopo aver modificato l'oggetto basta richiamare ancora il metodo set():

```
// aggiorna il punteggio di un pilota
ObjectSet result=db.get(new Pilot("Michael Schumacher",0));
Pilot found=(Pilot)result.next();
found.addPoints(11);
db.set(found);
System.out.println("Added 11 points for "+found);
retrieveAllPilots(db);
```

dove retrieveAllPilots() è il metodo

```
public static void retrieveAllPilots(ObjectContainer db) {
    ObjectSet result=db.get(Pilot.class);
    listResult(result);
}
```

OUTPUT:

```
Added 11 points for Michael Schumacher/111
2
Michael Schumacher/111
Rubens Barrichello/99
```

Cancellazione

Gli oggetti sono rimossi mediante il metodo delete()

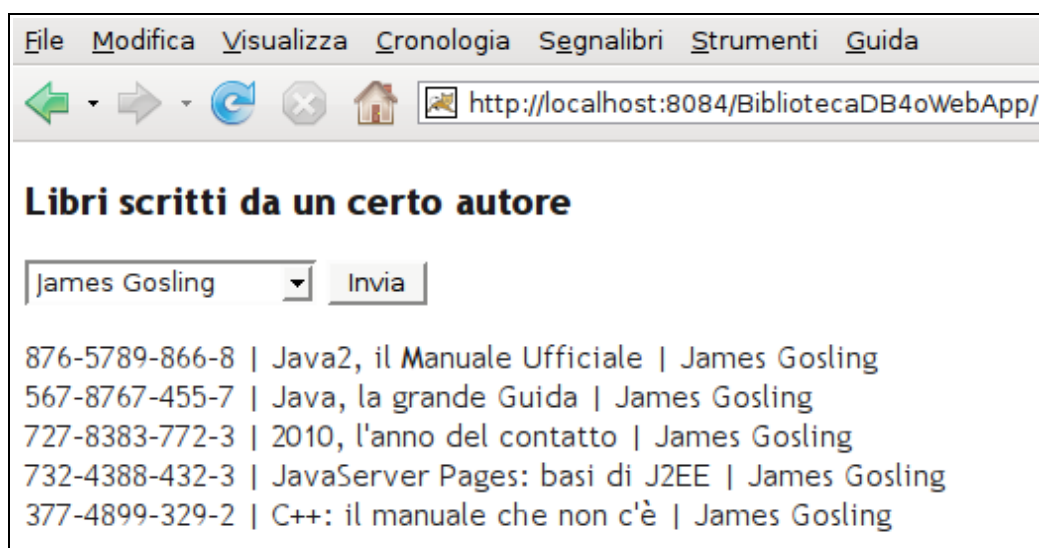
```
// cancella un pilota mediante il nome
ObjectSet result=db.get(new Pilot("Michael Schumacher",0));
Pilot found=(Pilot)result.next();
db.delete(found);
System.out.println("Deleted "+found);
retrieveAllPilots(db);
```

OUTPUT:

```
Deleted Michael Schumacher/111
1
Rubens Barrichello/99
```

L'esercitazione

Vogliamo progettare una piccola web application che ci consenta di ottenere, scegliendo un autore da una combo box, l'elenco dei libri scritti da quell'autore (è un esempio già trattato in altre esercitazioni), come mostrato nelle figure seguenti:



Per semplicità supponiamo la relazione Libro->Autore sia di tipo molti a uno.

```
/*
 * Libro.java
 *
 * Created on 7 aprile 2008, 20.38
 *
 */

package it.mauriziocozzetto.classiPkg;

/**
 *
 * @author maurizio
 */
public class Libro {

    private String isbn;
    private String titolo;
    private Autore autore;

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

    public Libro(String isbn, String titolo) {
        this.setIsbn(isbn);
        this.setTitolo(titolo);
    }

    public Libro(String isbn, String titolo, Autore autore) {
        this.setIsbn(isbn);
        this.setTitolo(titolo);
        this.setAutore-autore);
    }

    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }

    public String getTitolo() {
        return titolo;
    }

    public void setTitolo(String titolo) {
        this.titolo = titolo;
    }
}
```

```
    public Autore getAutore() {
        return autore;
    }

    public void setAutore(Autore autore) {
        this.autore = autore;
    }
} // fine classe Libro

/*
 * Autore.java
 *
 * Created on 7 aprile 2008, 20.39
 *
 */

package it.mauriziocozzetto.classiPkg;

/**
 *
 * @author maurizio
 */
public class Autore {
    private int idAutore;
    private String cognome;
    private String nome;

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

    // costruttore con 3 argomenti
    public Autore(int idAutore, String cognome, String nome) {
        this.setIdAutore(idAutore);
        this.setCognome(cognome);
        this.setNome(nome);
    }

    // restituisce l'anagrafica di un autore
    public String getAnagrafica() {
        return getNome() + " "+getCognome();
    }

    // seguono setter e getter
    public int getIdAutore() {
        return idAutore;
    }

    public void setIdAutore(int idAutore) {
        this.idAutore = idAutore;
    }
}
```

```
public String getCognome() {
    return cognome;
}

public void setCognome(String cognome) {
    this.cognome = cognome;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public boolean equals(String anagrafica) {
    return this.getAnagrafica().equals(anagrafica);
}

} // fine classe Autore
```

Ora con la classe *CreaElenchiDB4o* creiamo la base di dati a oggetti (package *it.mauriziocozzetto.db*, file *biblioteca.dat*) inserendo dei dati di esempio:

```
/*
 * CreaElenchiDB4o.java
 *
 * Created on 12 aprile 2008, 17.45
 *
 */

package it.mauriziocozzetto.classiPkg;
import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import java.util.List;

/**
 *
 * @author maurizio
 */
public class CreaElenchiDB4o {
    // reference al database
    static ObjectContainer db;

    /**
     * Creates a new instance of CreaElenchiDB4o
     */
    public CreaElenchiDB4o() {
        apri();

        creaElenchiAutori();
        creaElenchiLibri();
    }
}
```

```
List<Autore> elencoAutori = trovaAutori();
stampaAutori(elencoAutori);

List<Libro> elencoLibri = trovaLibri();
stampaLibri(elencoLibri);

db.close();
} // fine metodo costruttore

public static void main(String[] args) {
    CreaElenchiDB4o elenchiDB4o = new CreaElenchiDB4o();
}

public static void stampaAutori(List<Autore> autori) {
    for (Autore a : autori)
        System.out.println(a.getAnagrafica());
}

public static void stampaLibri(List<Libro> libri) {
    for (Libro l : libri)
        System.out.println(l.getTitolo());
}

public static void chiudi() {
    db.close();
}

public static void apri() {
    db = Db4o.openFile("src/java/it/mauriziocozzetto/db/biblioteca.dat");
}

public static List<Autore> trovaAutori() {
    List<Autore> autori = db.query(Autore.class);
    return autori;
}

public static List<Libro> trovaLibri() {
    List<Libro> libri = db.query(Libro.class);
    return libri;
}

public static void salvaAutore(Autore a) {
    db.set(a);
}

public static void salvaLibro(Libro l) {
    db.set(l);
}

public static void creaElencoAutori() {
    salvaAutore(new Autore(1,"Cozzetto","Maurizio"));
    salvaAutore(new Autore(2,"Cozzetto","Martina"));
    salvaAutore(new Autore(3,"Gardner","Martin"));
    salvaAutore(new Autore(4,"Pighizzini","Giovanni"));
} // fine metodo creaElencoAutori
```

```
public static void creaElencoLibri() {
    Autore a = new Autore(5,"Ferrari","Fabrizio");
    salvaAutore(a);

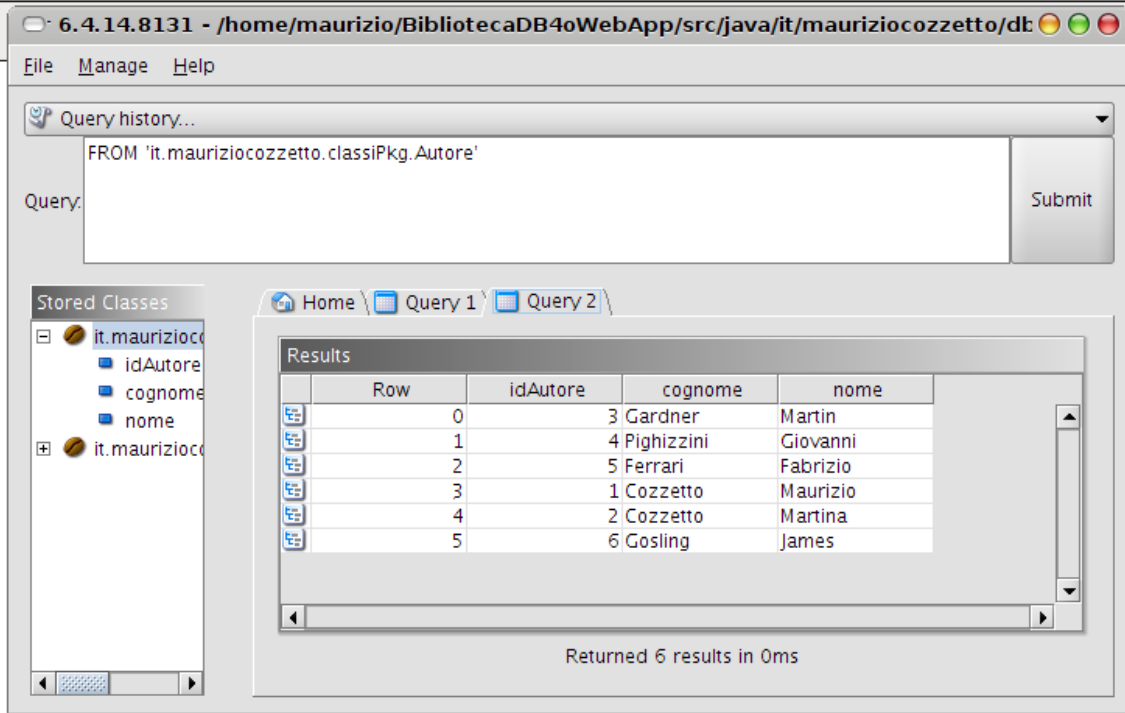
    salvaLibro(new Libro("123-2345-221-3","Java: dalle basi alla programmazione
avanzata",a));
    salvaLibro(new Libro("321-2314-211-4","Java J2EE",a));
    salvaLibro(new Libro("278-8483-288-2","2001, a space odyssey",a));
    salvaLibro(new Libro("829-903-6261-3","JavaServer Pages e Servlet",a));
    salvaLibro(new Libro("785-4567-897-8","C#: la Bibbia",a));

    Autore b = new Autore(6,"Gosling","James");
    salvaAutore(b);

    salvaLibro(new Libro("876-5789-866-8","Java2, il Manuale Ufficiale",b));
    salvaLibro(new Libro("567-8767-455-7","Java, la grande Guida",b));
    salvaLibro(new Libro("727-8383-772-3","2010, l'anno del contatto",b));
    salvaLibro(new Libro("732-4388-432-3","JavaServer Pages: basi di J2EE",b));
    salvaLibro(new Libro("377-4899-329-2","C++: il manuale che non c'è",b));
} // fine metodo creaElencoLibri

} // fine classe CreaElenchiDB4o
```

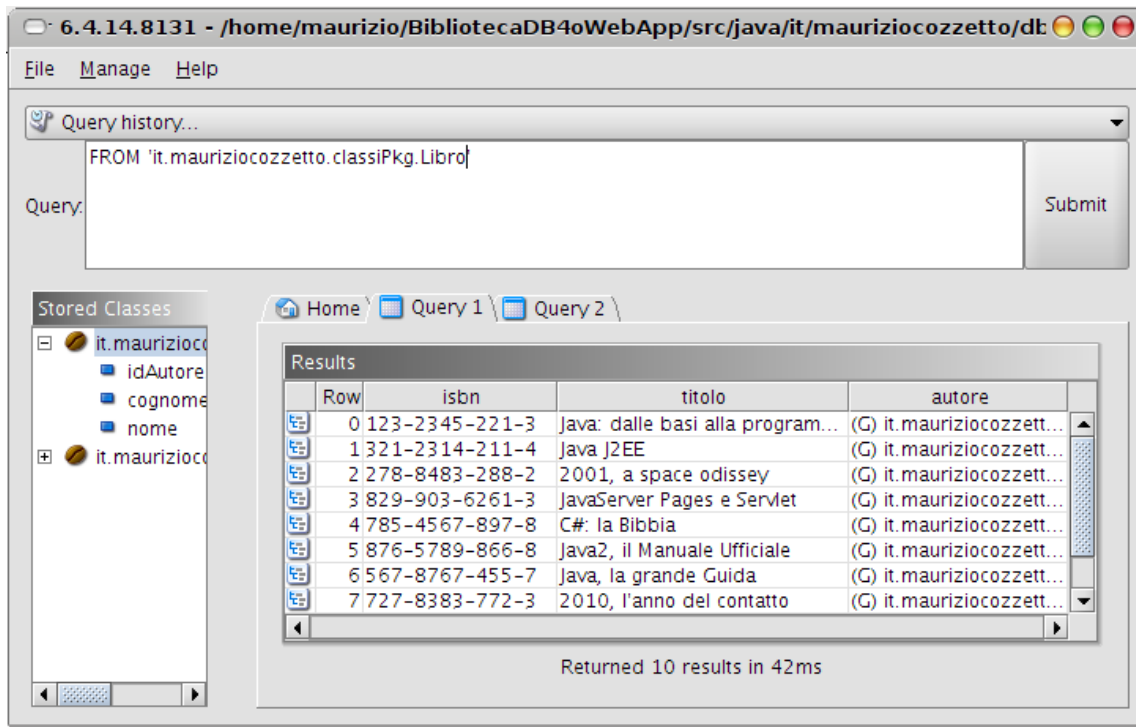
Possiamo osservare il contenuto della nostra base di dati a oggetti usando l'*Object Manager* disponibile all'indirizzo http://developer.db4o.com/files/folders/objectmanager_61/default.aspx



The screenshot shows the db4o Object Manager interface. The title bar indicates the path: 6.4.14.8131 - /home/maurizio/BibliotecaDB4oWebApp/src/java/it/mauriziocozzetto/db. The interface includes a menu bar (File, Manage, Help), a query history dropdown, and a query input field containing the SQL query: FROM 'it.mauriziocozzetto.classiPkg.Autore'. A 'Submit' button is located to the right of the query field. Below the query field, there is a 'Stored Classes' panel on the left showing a tree view of classes: it.maurizioco (expanded) with sub-classes idAutore, cognome, and nome; and it.maurizioco (collapsed). The main area displays a 'Results' table with the following data:

Row	idAutore	cognome	nome
0	3	Gardner	Martin
1	4	Pighizzini	Giovanni
2	5	Ferrari	Fabrizio
3	1	Cozzetto	Maurizio
4	2	Cozzetto	Martina
5	6	Gosling	James

At the bottom of the results panel, it states: Returned 6 results in 0ms.



A questo punto progettiamo la classe DAO (data access object). Essa conterrà tutte le interrogazioni alla base dati e fornirà i ResultSet (ma sarebbe meglio dire gli ObjectSet) richiesti.

```

/*
 * DAO.java
 *
 * Created on 7 aprile 2008, 21.20
 */

package it.mauriziocozzetto.classiPkg;
import java.io.File;
import java.net.URISyntaxException;
import java.util.List;
import com.db4o.Db4o;
import com.db4o.query.Predicate;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import it.mauriziocozzetto.classiPkg.Autore;
import it.mauriziocozzetto.classiPkg.Libro;

/**
 * @author maurizio
 */
public class DAO {
    // reference al database
    private ObjectContainer db;

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

```

```
// restituisce i titoli dei libri (come valori stringa)
public String[] getElencoTitoli() {
    List<Libro> libri = getLibri();
    int numLibri = libri.size();

    String s[] = new String[numLibri];
    int i = 0;
    for (Libro l : libri) {
        s[i] = l.getTitolo();
        i++;
    }
    return s;
}

// restituisce la lista dei libri
public List<Libro> getLibri() {
    return db.query(Libro.class);
}

// restituisce la collection dei libri scritti da un determinato autore
public List<Libro> getElencoLibriAutore(final String anagrafica) {

    // è una NQ, Native Query, query "nativa"
    // consulta il sito www.db4o.com
    // restituisce l'elenco dei libri di un certo autore
    List <Libro> elencoLibri = db.query(new Predicate<Libro>() {
        public boolean match(Libro libro) {
            // se l'anagrafica dell'autore del libro coincide con il parametro allora true
            return libro.getAutore().equals(anagrafica);
        }
    });

    return elencoLibri;
} // fine metodo getElencoLibriAutore()

// ritorna la collection degli autori
public List<Autore> getAutori() {
    return db.query(Autore.class);
}

// restituisce le anagrafiche degli autori
public String[] getElencoAutori() {
    List<Autore> autori = getAutori();
    int numAutori = autori.size();

    String s[] = new String[numAutori];

    int i = 0;
    for (Autore a : autori) {
        s[i] = a.getAnagrafica();
        i++;
    }
    return s;
} // fine metodo getElencoAutori
```

```
public void close() {
    db.close();
}

public void open() {
    File file=null;
    try {
        file = new
File(getClass().getResource("/it/mauriziocozzetto/db/biblioteca.dat").toURI());
    } catch (URISyntaxException ex) {
        ex.printStackTrace();
        return;
    }
    String path = file.getPath();
    db = Db4o.openFile(path);
} // fine metodo open

} // fine classe ElencoLibri
```

L'applicativo vero e proprio è fatto dalla sola pagina web *index.jsp*:

File index.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.List"%>
<%@page import="it.mauriziocozzetto.classiPkg.*"%>

<!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>Libri scritti da un certo autore</title>
    <style type="text/css">
      body,td,th {
        font-family: Trebuchet MS;
        font-size: 16px;
      }
    </style>
  </head>
  <body>

    <h3>Libri scritti da un certo autore</h3>

    <jsp:useBean id="dao" class="it.mauriziocozzetto.classiPkg.DAO"/>

    <%
      // apro la base dati a oggetti
      dao.open();

      String[] anagraficheAutori = dao.getElencoAutori();
    %>
```

```
<form name="provaFrm" action="{request.requestURI}" method="post">
  <p><select name="mnuAutore">
    <%
      for (int i = 0; i<anagraficheAutori.length; i++) {
    %>
      <option><%=anagraficheAutori[i]%></option>
    <%
      } // fine for
    %>
  </select>
  <input type="submit" name="inviaBtn" value="Invia"/></p>
</form>

<%
  // se l'utente ha premuto sul tasto Invia
  if (request.getParameter("inviaBtn")!=null) {

    // mi faccio dare l'anagrafica dell'autore
    String anagraficaAutore = request.getParameter("mnuAutore");

    // trovo la lista dei libri scritti da quell'autore
    List<Libro> listaLibri = dao.getElencoLibriAutore(anagraficaAutore);

    // se non ci sono libri
    if (listaLibri.size()==0)
      out.println("Non ci sono libri scritti da "+anagraficaAutore);
    else {
      // visualizziamo l'elenco dei libri di quell'autore
      for (Libro l : listaLibri)
        out.println(l.getIsbn()+" | "+l.getTitolo()+" | "+l.getAutore().getAnagrafica()
+"<br/>");
      } // fine if

    } // fine if request

    // chiusura della base dati
    dao.close();

  %>

</body>
</html>
```