# Laboratorio di Basi di Dati

Docenti: Alberto Belussi

*Lezione 7*

# JDBC: breve riepilogo (1)

- L'interfaccia JDBC è contenuta nei package java.sql e javax.sql.
- Le classi più utilizzate sono:
  - Connection: collegamento attivo con una base di dati, tramite il quale un programma Java può leggere e scrivere i dati.
    - Un oggetto Connection viene creato tramite una chiamata a DriverManager.getConnection(...)

# JDBC: breve riepilogo (2)

- L'interazione con il DBMS avviene attraverso oggetti generati invocando metodi dell'oggetto Connection, che consentono di inviare delle istruzioni SQL e di ricevere i risultati. Ci sono due classi di oggetti per tale interazione:
  - Statement: utilizzato per eseguire interrogazioni SQL statiche. Un oggetto Statement può essere creato con Connection.createStatement().
  - PreparedStatement: estensione di Statement che consente di precompilare interrogazioni SQL con parametri di input etichettati con il simbolo '?' e attualizzati successivamente con metodi specifici prima dell'esecuzione effettiva. Un oggetto PreparedStatement può essere creato con Connection.prepareStatement(stringaSQL).

# JDBC: breve riepilogo (3)

- ResultSet: risultato composto da un insieme ordinato di righe prodotte da un server SQL.
  - Un ResultSet viene restituito come risultato della chiamata al metodo executeQuery(stringaSQL) di un oggetto Statement o PreparedStatement.
- SQLException: classe base per eccezioni utilizzata dall'API JDBC.

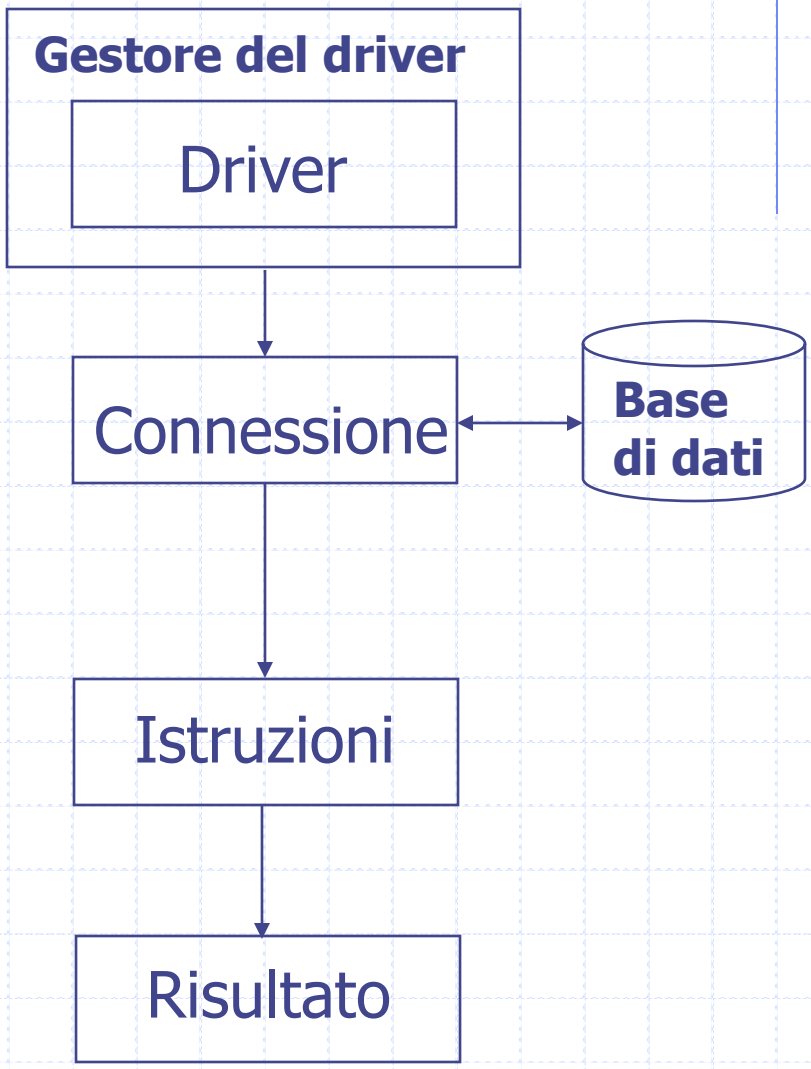
# Interazione con DBMS via JDBC: riepilogo

1) Caricamento del driver JDBC:
```
Class.forName("nome-driver");
```

2) Apertura della connessione con la base di dati:
```
DriverManager.getConnection
 (jdbc:tipoDBMS://URL/Database,
user, passwd)
```

3) Esecuzione di una query SQL:
```
stmt = con.createStatement();
rs = stmt.executeQuery
      ("SELECT * FROM Tabella");
```

4) Elaborazione del risultato:
```
while(rs.next()) {
      name = rs.getString("name");
      age = rs.getInt("age"); }
```



# Esecuzione di interrogazioni

- Per definire una query esistono due classi: Statement e PreparedStatement.
- Uno Statement rappresenta una query semplice con tutti i dati specificati all'atto della creazione.
- Un preparedStatement permette di sviluppare uno schema di query (con parametri) che può essere utilizzato più volte con valori differenti.

# Con Statement

```
String nome= “Alberto";
String cognome = “Belussi";

Statement stmt;
ResultSet rs;

sql   = " SELECT * ";
sql += " FROM PERSONA ";
sql += " WHERE nome = ' "+ nome + " ' ";
sql += " AND cognome = ' " + cognome + " ' ";

stmt = connection.createStatement();

rs=stmt.executeQuery(sql);
```

# Con PreparedStatement

```
String nome= “Alberto";
String cognome = “Belussi";

PreparedStatement pstmt;
ResultSet rs;

sql   = " SELECT * ";
sql += " FROM PERSONA ";
sql += " WHERE nome = ? AND cognome = ?";

pstmt = con.prepareStatement(sql);
pstmt.clearParameters()
pstmt.setString(1, nome);
pstmt.setString(2, cognome);

rs=pstmt.executeQuery();
```

# Esempio di Servlet (1)

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Questa classe gestisce le richieste riguardanti le ricerche all'interno della
 * base di dati. Risponde solamente a richieste HTTP di tipo GET. I possibili
 * parametri che vengono considerati e le relative azioni effettuate sono le
 * seguenti:
 * <br>
 * <ol>
 *   <li>nessun parametro: viene visualizzata la lista di tutti i corsi di studio
    esistenti;</li>
 *   <li>parametro 'id': vengono visualizzate le informazioni (compresa la/le facoltà  di
    appartenenza)
 *       del corso di studi con l'id specificato.</li>
 * </ol>
 *
 */
```

# Esempio di Servlet (2)

```
public class ServletCorsoStudi extends HttpServlet {
    /**
     * Questo metodo risponde alle richieste HTTP di tipo GET.
     * Elabora le richieste collegandosi alla base di dati e producendo le pagine
     * HTML di output.
     * @param request Oggetto HttpServletRequest dal quale ottenere
     * informazioni circa la richiesta effettuata.
     * @param response Oggetto HttpServletResponse per l'invio delle risposte.
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {

        //Dati di identificazione dell'utente
        String user = "userlab00";
        String passwd = "";
        /** URL per la connessione alla base di dati è formato dai seguenti componenti:
           * <protocollo>://<host del server>/<nome base di dati>.
        */
        String url = "jdbc:postgresql://dbserver.scienze.univr.it/did2011";
```

# Esempio di Servlet (3)

```
//Dichiarazione delle query necessarie
//Query per il recupero delle informazioni minimali sui corsi di studio
String css = "SELECT id, Codice, Nome "+
                "FROM corsostudi ORDER BY Nome";
//Query per il recupero di tutte le informazioni riguardanti un corso di studio
String cs = "SELECT Codice,  Abbreviazione, Nome, "+
                    "DurataAnni, Sede, Informativa "+
              "FROM corsostudi WHERE id = ?";
//Query per il recupero delle facoltà che gestiscono un dato corso di studio
String csf = "SELECT DISTINCT f.nome "+
      " FROM facolta f JOIN corsoinfacolta csf ON f.id=csf.id_facolta "+
            " WHERE csf.id_corsostudi = ?";

//Caricamento del driver JDBC per il database
try {
      Class.forName("org.postgresql.Driver");
}
catch (ClassNotFoundException cnfe) {
      out.println("Driver jdbc non trovato: " + cnfe.getMessage());
}
```

# Esempio di Servlet (4)

```
//Dichiarazione delle variabili necessarie alla connessione e
//al recupero dei dati
Connection con = null;
PreparedStatement pstmt = null;
Statement stmt = null;
ResultSet rs = null, rsf = null;

PrintWriter out = response.getWriter();
response.setContentType("text/html; charset=ISO-8859-1");

out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01”+
                “Transitional//EN\"");
out.println(" \"http://www.w3.org/TR/REC-html40/loose.dtd\">");

//Recupero del possibile parametro d'ingresso id della servlet
String id = "";
if (request.getParameter("id") != null) {
        // Ottengo se presente il parametro 'id'
                id = request.getParameter("id");
}
```

# Esempio di Servlet (5)

```
try {
// Tentativo di connessione al database
        con = DriverManager.getConnection(url, user, passwd);

        out.println("<html>");
        out.println("<head>");
        // Determino la pagina da visualizzare a seconda della presenza
        // o meno del parametro id
        if (id.equals("")) {// ID NON PRESENTE
                // Recupero e visualizzo tutti i corsi di studi disponibili
                stmt = con.createStatement();
                // Eseguo l'interrogazione desiderata
                rs = stmt.executeQuery(css);
                //Genero la parte statica della pagina HTML risultante
                out.println("<title>Corsi di Studio Esistenti</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h1>Corsi di Studio Esistenti:</h1>");
                out.println("<table>");
                out.println("<tr><th>Codice</th><th>Nome</th></tr>");
```

# Esempio di Servlet (6)

```
            //Genero la parte dinamica della pagina HTML risultante
            //recuperando le informazioni dal ResultSet
            while (rs.next()) {  //Per ogni record del ResultSet
                  out.println("<tr><td><a "+
                  "href=\"ServletCorsoStudi?id="+rs.getInt("id")+"\">"+
                  rs.getString("Codice") + "</a></td><td>" +
                  rs.getString("NomeCorsoStudi") + "</td></tr>");
            }
            out.println("</table>");
      } else {// ID PRESENTE
      // Visualizzo le informazioni di un particolare corso di studi
      // Eseguo l'interrogazione per il recupero delle info su un corso
      // di studi
            pstmt = con.prepareStatement(cs);
            pstmt.clearParameters();
            //Imposto il parametro della query
            pstmt.setInt(1, Integer.parseInt(id));
```

# Esempio di Servlet (7)

```
//Eseguo la query
rs = pstmt.executeQuery();

//Il ResultSet contiene un solo record dato che è stata
//effettuata un'interrogazione su una particolare chiave.
//Quindi non devo scorrere diversi record ma devo
//soltanto posizionarmi sul primo e unico disponibile.
rs.next();

//Carico la query per il recupero della/e
//facoltà del corso di studi nel
//PreparedStatement
pstmt = con.prepareStatement(csf);
pstmt.clearParameters();
//Imposto il parametro della query
pstmt.setInt(1, Integer.parseInt(id));
//Eseguo la query
rsf=pstmt.executeQuery();
```

# Esempio di Servlet (8)

```
//Utilizzando il ResultSet rs visualizzo le info sul corso
out.println("<title>Informazioni su un Corso di Studi”+
        “</title>");
out.println("</head>");
out.println("<body>");
out.println("<h2>Informazioni sul Corso di Studi</h2>");
out.println("<ul><li><b>Codice</b>:
        "+rs.getString("Codice")+"</li>");
out.println("<li><b>Nome</b>:
        "+rs.getString("Nome")+"</li>");
out.println("<li><b>Abbreviazione</b>:
        "+rs.getString("Abbreviazione")+"</li>");
out.println("<li><b>Durata anni</b>:
        "+rs.getInt("Durataanni")+"</li>");
out.println("<li><b>Sede</b>:
        "+rs.getString("Sede")+"</li>");
out.println("<li><b>Facoltà </b>: ");
```

# Esempio di Servlet (9)

```
//Utilizzando il ResultSet rsf visualizzo la/e facoltà di app.
while (rsf.next()) {
        out.println(rsf.getString("Nome")+", ");
}

out.println("</li>");
out.println("<li><b>Informativa</b>: “
            rs.getString("Informativa")+"</li></ul>");

out.println("<a href=\"ServletCorsoStudi\"> “+
            “<font color=\"00AA00\"><< Back</font></a>");
}

//Termino la pagina HTML
out.println("</body>");
out.println("</html>");
```

# Esempio di Servlet (10)

```
            //Chiudo la connessione
            con.close();
        } catch(Exception e) {
            e.printStackTrace();
        } // end try
    } // end doGet
} // end classe
```

# Java Data Bean (1)

- Una classe Java per essere utilizzata come Java Data Bean deve essere scritta seguendo le seguenti direttive:
  - **deve implementare un costruttore senza argomenti.**

    **Esempio: PersonaBean();**
  - **Per ciascuna proprietà della classe che si vuole rendere visibile, deve essere implementato un metodo con la signature getNomeCampo() dove NomeCampo è il nome del campo (con le iniziali maiuscole).**

    **Esempio: se una bean ha un campo numeroTelefono il metodo dovrà essere getNumeroTelefono().**
  - **Per ciascun campo della classe che si vuole rendere modificabile, deve essere implementato un metodo con la signature setNomeCampo(ClasseCampo v) dove NomeCampo è il nome del campo (con le iniziali maiuscole) e ClasseCampo è il tipo del campo.**

    **Esempio: se un bean ha un campo numeroTelefono di tipo String, il metodo dovrà essere setNumeroTelefono(String value).**
- Inoltre per consuetudine si usa nominare la classe con il suffisso "Bean".

  Esempio: AutoreBean

# Java Data Bean (2)

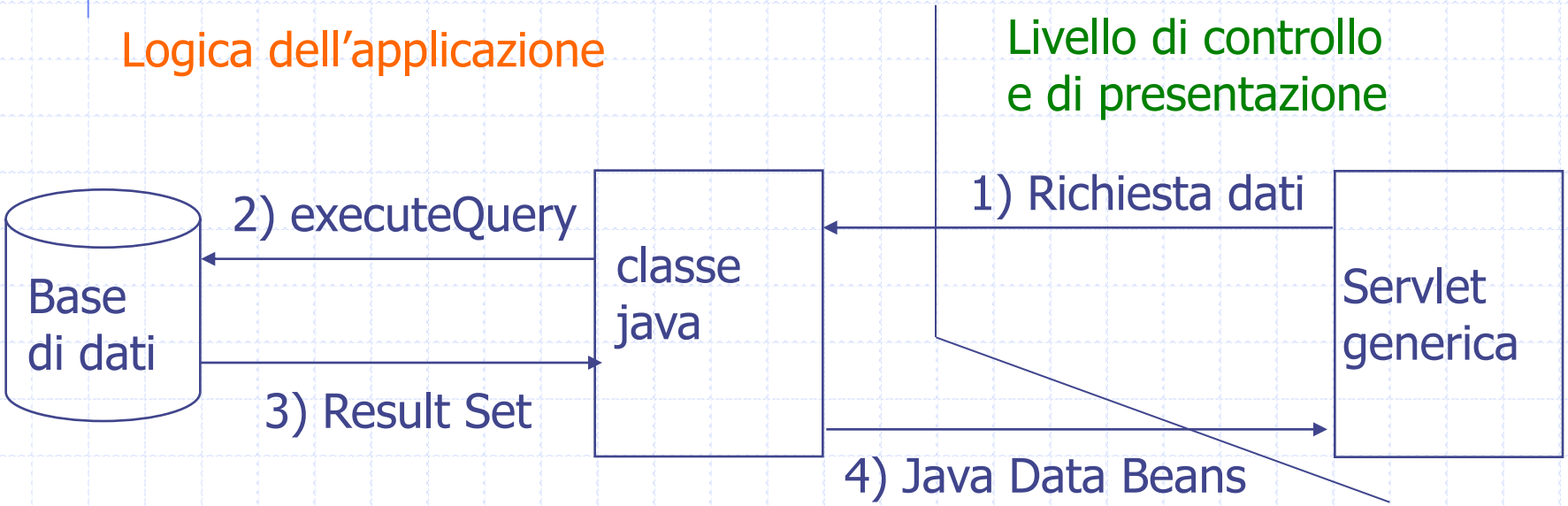
- Per le proprietà della classe di tipo boolean il metodo get va chiamato isNomeCampo() dove NomeCampo è il nome del campo (con le iniziali maiuscole), mentre il metodo set segue la convenzione descritta precedentemente.

# Java Data Bean (3)

- Un Java Data Bean risulta essere il miglior componente per mappare una tupla di un ResultSet in un oggetto Java.
- A tal fine, esso deve contenere:
  1. tanti campi **private** quanti sono gli attributi;
  2. un costruttore di default che assegna i valori di default ai campi;
  3. i metodi **pubblici** di accesso getter e setter per gli attributi che si vogliono esporre.

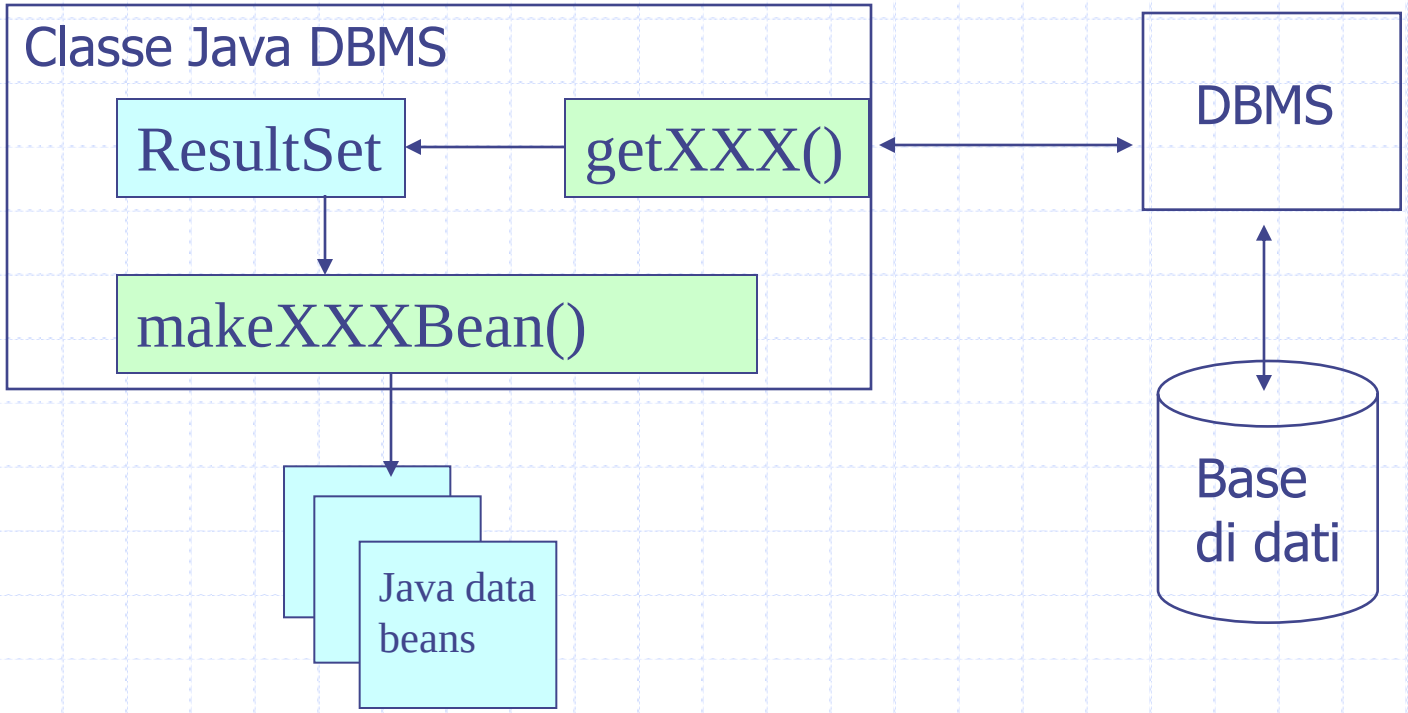
# Basi di dati e Bean

- I Java Data Bean sono dei componenti fondamentali nella strutturazione di una applicazione web che utilizzi una connessione ad una base di dati.
- I Java Data Bean, insieme ad una o più opportune classi Java, permettono di separare la logica dell'applicazione dalla parte di controllo e di presentazione delle informazioni.



# Classe DBMS

Quando una applicazione web interagisce con una base di dati per recuperare le informazioni di interesse, tutte le query sono gestite all'interno di un'unica classe Java (DBMS.java) che, utilizzando un insieme opportuno di componenti Java Data Beans, realizza un'interfaccia tra le servlet e la base di dati.



# Esempi da scaricare (1.1)

1. Creare nella directory ~/tomcat/src una nuova directory CorsoStudi.
2. Scaricare nella directory ~/tomcat/src/CorsoStudi il pacchetto ServletSenzaBean.zip dalla pagina web del corso.
3. Scompattare il pacchetto: unzip ServletSenzaBean.zip.
4. Si otterrà il file ServletCorsoStudi.java per la visualizzazione dei corsi di studio dell'ateneo e dei dati che descrivono un singolo corso di studi.
5. Creare il contesto ~/tomcat/webapps/CorsoStudi e nel suo interno le directory WEB-INF, e le directory classes e lib. Nella directory WEB-INF creare (copiandolo e incollandolo dal context ROOT) il file web.xml e context.xml, in cui DEVE essere dichiarata la servlet ServletCorsoStudi.
6. Dalla directory lib creare il link simbolico nel seguente modo:

   ln -s /usr/share/java/postgresql-jdbc3.jar

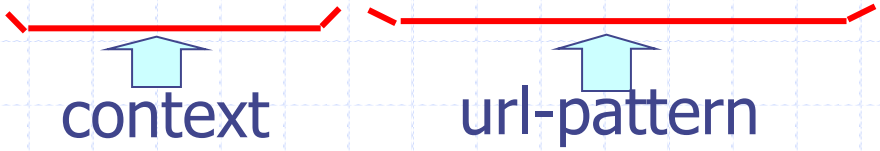
# Esempi da scaricare (1.2)

1. Per far funzionare l'esempio è necessario modificare: la parte relativa alla connessione al database did2011 inserendo il proprio utente e la corrispondente password.
2. Per compilare i files contenuti nella directory ~/tomcat/src/CorsoStudi posizionarsi nella directory medesima ed eseguire i seguenti comandi:

   **javac -d ../../webapps/CorsoStudi/WEB-INF/classes ServletCorsoStudi.java**

   (i file compilati vengono salvati nella corretta directory del context CorsoStudi).

1. Per vedere la pagina web prodotta dalla servlet: http://localhost:8080/CorsoStudi/servlet/CorsoStudi



# Esempi da scaricare (2)

1. Scaricare nella directory ~/tomcat/src/CorsoStudi il pacchetto ServletConBean.zip dalla pagina web del corso.
2. Scompattare il pacchetto: unzip ServletConBean.tgz
3. Si otterrà il file ServletCorsoStudiB.java e le classi JAVA CorsoStudiBean.java e DBMS.java per la visualizzazione dei corsi di studio dell'ateneo e dei dati che descrivono un singolo corso di studi. I files DBMS.java e CorsoStudiBean.java si trovano nella sottocartella did (che costituisce anche un package java)
4. La nuova servlet ServletCorsoStudiB va dichiarata nel file web.xml
5. Per far funzionare questo secondo esempio è necessario modificare la parte relativa alla connessione al database didattica inserendo il proprio utente e la corrispondente password nella classe DBMS.java.
6. Per compilare i file contenuti nella directory ~/tomcat/src/CorsoStudi posizionarsi nella directory medesima ed eseguire i seguenti comandi:

   **javac -d ../../webapps/CorsoStudi/WEB-INF/classes *.java**

   (i file compilati vengono salvati nella corretta directory del context CorsoStudi).
7. Errori nel file di log:

   tomcat/logs/catalina.out.

# Consegna Esercitazione 7

Inviare via email al docente due file di nome:

**"ES7-servlet-<matricola>.java"**

**"ES7-dbms-<matricola>.java"**

Contenenti rispettivamente: la servlet ServletCorsoStudiB.java e la classe DBMS.java entrambe estese come descritto al punto 2) dell'esercizio proposto.

Il messaggio dovrà aderire al seguente formato:

- Oggetto: <Matricola> - Esercitazione 7
- Contenuto: <Matricola> - <Cognome> - <Nome>
- Allegato: file di nome ES7-servlet-<Matricola>.java, ES7-dbms-<Matricola>.java

Il messaggio email va spedito entro le 24.00 del giorno 14 maggio 2012.

# Riferimenti


- Marty Hall. “CORE. Servlets and JavaServer Pages”. Sun Microsystems Press.
- Phil Hanna. “JSP. La guida Completa.” McGraw-Hill.