# Laboratorio di Basi di Dati e Multimedia

Laurea in Informatica Multimediale

Docente: Carlo Combi

Email: carlo.combi@univr.it

**Lezione 7**

# Java DataBase Connectivity

- JDBC consente di interfacciare una base di dati ed un programma java
  - fornisce un'interfaccia standard per tutte le basi di dati
- JDBC è un'interfaccia operante a livello delle chiamate
  - un programma può accedere alle funzioni JDBC utilizzando semplici metodi o chiamate a funzioni
- La connessione alla base di dati avviene utilizzando un driver JDBC rappresentato da una classe Java

# Principali classi JDBC (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 può essere creato tramite una chiamata a DriverManager.getConnection()

# Principali classi JDBC (2)

- Statement: oggetto che, tramite una connessione, consente di inviare delle istruzioni SQL e di ricevere i risultati. Esistono due tipi di istruzioni:

  - Statement: utilizzata per eseguire interrogazioni SQL statiche. Un'istruzione Statement può essere creata con Connection.createStatement()

  - PreparedStatement: estensione di Statement che utilizza codice SQL precompilato con parametri di input definiti in modo dinamico. Permette 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)

# Principali classi JDBC (3)

- ResultSet: risultato composto da un insieme ordinato di righe prodotte da un server SQL
  - Un ResultSet può essere restituito dalla chiamata al metodo executeQuery(stringaSQL) di un oggetto Statement o PreparedStatement.
  - Metodo next() per iterare fra le righe di un ResultSet e metodi getxxx() per estrarre il valore delle colonne dove xxx e' il tipo di dati Java
- SQLException: classe base per eccezioni utilizzata dall'API JDBC. Offre metodi che possono fornire il valore SQLState per ogni codice d'errore specifico del database

# Operazioni di base di JDBC

1. Caricamento del driver JDBC

```
Class.forName("nome-driver");
```

4. Apertura connessione con il DB

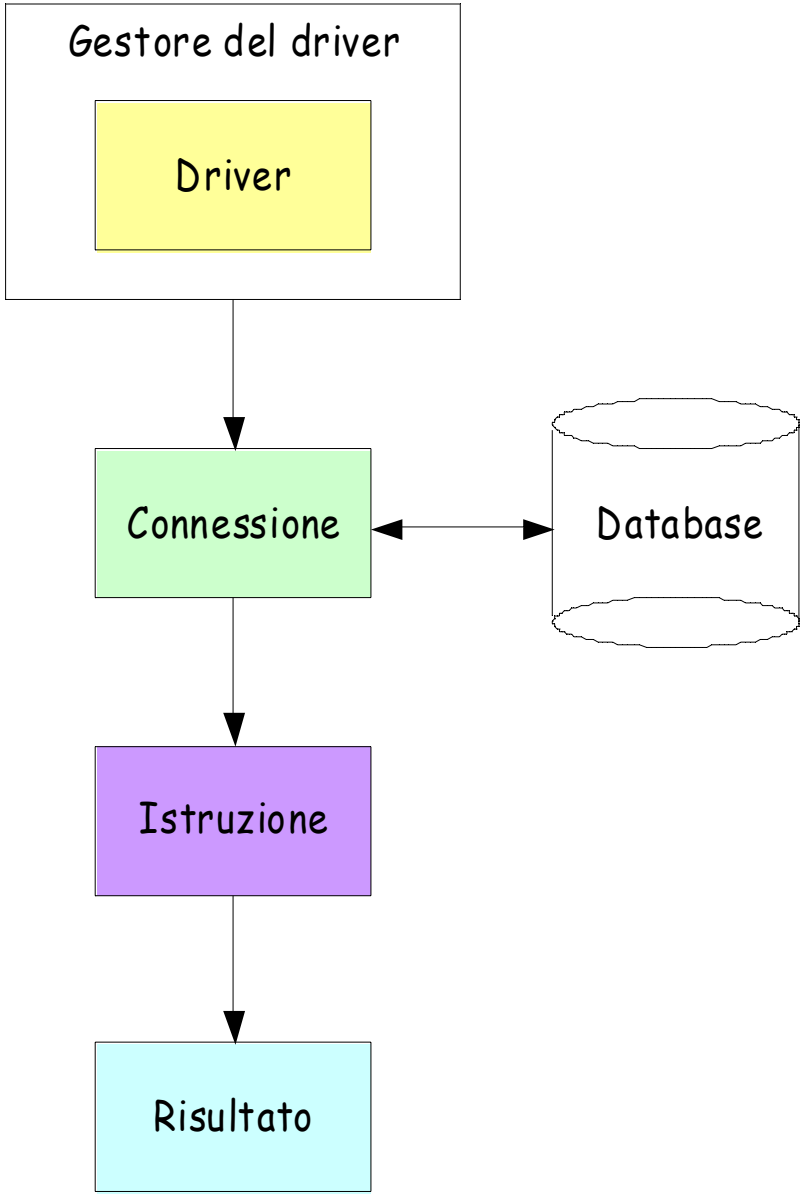
```
DriverManager.getConnection
  (jdbc:tipoDBMS://URL/Database)
```

8. Esecuzione di una query SQL

```
stmt = con.createStatement();
stmt.executeQuery
  ("SELECT * FROM Tabella");
```

13. Elaborazione del Risultato

```
while(rs.next(){
  name = rs.getString("name");
  amount = rs.getInt("amt"); }
```



# Connessione (1)

- Caricare il driver JDBC specifico del DBMS. Ad esempio per il database PostgreSQL:

```
try
{
  Class.forName("org.postgresql.Driver");
 }
catch (ClassNotFoundException cnfe)
{
  out("Driver jdbc non trovato: " + cnfe.getMessage());
 }
```

- Solitamente si verifica un'eccezione quando il class loader non riesce a trovare nel path la libreria postgresql.jar

# Connessione (2)

- Preparare l'URI del database, il nome utente e la password per l'autorizzazione al collegamento
  - Ad esempio per collegarsi al database esercitazioni del DBMS presente su sqlserver, si possono preparare tre variabili String:

```
String uri
  ="jdbc:postgresql://sqlserver.sci.univr.it/dblabxx";
String user = "postgres login";
String passwd = "";
```

- L'URI è sempre composto da:
  "*jdbc:tipoDBMS:URLDatabase*"
  dove l'URL del database solitamente ha il formato
  //host/nomeDatabase

# Connessione (3)

- Attivare una connessione con il metodo statico
  DriverManager.getConnection()

  Ad esempio:

```
Connection connection =
DriverManager.getConnection(uri,user,passwd);
```

- A questo punto l'oggetto connection rappresenta la connessione al database. Tutte le operazioni che si possono eseguire sul database sono date dai metodi di questo oggetto

# 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= “Carlo";
String cognome = “Combi";

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= “Carlo";
String cognome = “Combi";

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();
```

# Esempi di Servlet

# 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://sqlserver.sci.univr.it/didattica";
```

# Esempio di Servlet (3)

```
//Dichiarazione delle query necessarie
//Query per il recupero delle informazioni minimali sui corsi di studio
String css = "SELECT id, Codice, NomeCorsoStudi “+
                "FROM corsostudi ORDER BY NomeCorsoStudi";
//Query per il recupero di tutte le informazioni riguardanti un corso di studio
String cs = "SELECT * "+
                "FROM corsostudi WHERE id = ?";
//Query per il recupero delle facoltà di appartenenza di un dato corso di studio
String csf = "SELECT DISTINCT f.nome "+
        " FROM facolta f INNER 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 studio 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 studio
// Eseguo l'interrogazione per il recupero delle info su un corso
// di studio
        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 studio 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 Studio”+
        “</title>");
out.println("</head>");
out.println("<body>");
out.println("<h2>Informazioni sul Corso di Studio</h2>");
out.println("<ul><li><b>Codice</b>:
        "+rs.getString("Codice")+"</li>");
out.println("<li><b>Nome</b>:
        "+rs.getString("NomeCorsoStudi")+"</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
```

# Esempi da scaricare (1)

- Creare nella directory ~/tomcat/src una nuova directory CorsoStudi
- Scaricare nella directory ~/tomcat/src/CorsoStudi il pacchetto Servlet_Without_Bean.tgz dalla pagina web del corso.
- Scompattare il pacchetto: tar xzvf Servlet_Without_Bean.tgz
- Si otterrà la directory Servlet_Without_Bean contenente la Servlet ServletCorsoStudi.java per la visualizzazione dei corsi di studio dell'ateneo e dei dati che descrivono un singolo corso di studi.
- Nella directory WEB-INF di ~/tomcat/webapps/CorsoStudi creare la directory lib
- Dalla directory lib creare il link simbolico nel seguente modo:
  - ln -s /usr/share/java/postgres-jdbc3.jar
- Per far funzionare l'esempio è necessario modificare: la parte relativa alla connessione al database didattica inserendo il proprio utente e la corrispondente password.
- Per compilare i file contenuti nella directory ~/tomcat/src/CorsoStudi/Servlet_Without_Bean posizionarsi nella directory medesima ed eseguire i seguenti comandi:

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

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

9)Per vedere la pagina web prodotta dalla servlet: http://localhost:8080/CorsoStudi/servlet/ServletCorsoStudi

# Servlet e immagini

# Bytea

- In PostgreSQL possiamo utilizzare il dominio bytea per dati multimediali come ad esempio immagini o video.
- Il tipo di dati bytea permette di memorizzare stringhe binarie, cioe' sequenze di byte.
- Le stringhe binarie si distinguono dalle stringhe di caratteri perche' consentono di codificare anche valori che non sono ammessi dalla codifica dei caratteri scelta per il DB. Inoltre le operazioni sono operazioni generiche su byte e non dipendono dalla codifica scelta per i caratteri.

# Esempio: Servlet per il caricamento di immagini (1)

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

/**
 * ServletQuery. Esempio di Servlet che si connette alla base di dati ed
 * inserisce un'immagine nella base di dati.

public class ServletStore extends HttpServlet {

   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
      throws IOException, ServletException {

     Connection con = null;
```

Continua ….

# Esempio: Servlet per il caricamento di immagini (2)

```
String url = "jdbc:postgresql://sqlserver.sci.univr.it/dblabxxx";
String user = “userlabxxx";
String passwd = "";

File f = new File("idefix.jpg");
 FileInputStream fis = new FileInputStream(f);

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

Continua ….

# Esempio: Servlet per il caricamento di immagini (3)

```
try {
    /**
     * Connessione alla base di dati
     */

    con = DriverManager.getConnection(url, user, passwd);
    String str = "INSERT INTO prova VALUES (1, ?)";

    PreparedStatement pst = con.prepareStatement(str);
    pst.clearParameters();
    pst.setBinaryStream(1, (InputStream)fis, (int)f.length());
    pst.execute();
    con.close();
```

Continua ….

# Esempio: Servlet per il caricamento di immagini (4)

```
        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\">");


         out.println("<html>");
         out.println("<head>");
         out.println("<title>Caricamento immagine</title>");
         out.println("</head>");
         out.println("<body>");
         out.println("<h1>Immagine</h1>");

      } catch(Exception e) {
         e.printStackTrace();
      }
    }
}
```

# Esempio: Servlet per la visualizzazione di immagini (1)

```
import java.io.*;
import java.util.*;
import java.sql.*;

import javax.servlet.*;
import javax.servlet.http.*;

import com.oreilly.servlet.MultipartRequest;
import com.sun.image.codec.jpeg.*;
import java.awt.image.*;

public class ServletShow extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    String sql;
    Connection con = null;
    PreparedStatement pstmt;
    ResultSet rs;
```

Continua ….

# Esempio: Servlet per la visualizzazione di immagini (2)

```
String url = "jdbc:postgresql://sqlserver.sci.univr.it/dblabxxx";
String user = “userlabxxx";
String passwd = "";

/**
 * Caricamento del driver JDBC per il database
 */

try {
    Class.forName("org.postgresql.Driver");
} catch (ClassNotFoundException cnfe) {
    log("Driver jdbc non trovato: " + cnfe.getMessage());
}
```

Continua ….

# Esempio: Servlet per la visualizzazione di immagini (3)

```
try {
   sql =  "SELECT * " +
            "FROM prova " +
            "WHERE codice = 1";
    /**
     * Connessione alla base di dati
     */

     con = DriverManager.getConnection(url, user, passwd);
   pstmt = con.prepareStatement(sql);

     /**
     * Esecuzione dell'interrogazione
     */

     rs=pstmt.executeQuery();

   response.setContentType("image/jpeg");
```

Continua ….

# Esempio: Servlet per la visualizzazione di immagini (4)

```
            while (rs.next()) {
            InputStream is = rs.getBinaryStream("img");
            JPEGImageDecoder decoder = JPEGCodec.createJPEGDecoder(is);
            BufferedImage bi = decoder.decodeAsBufferedImage();

            ServletOutputStream sos = response.getOutputStream();
            JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(sos);
            encoder.encode(bi);
        }
        con.close();
    }catch (SQLException sqle) {
        sqle.printStackTrace();
    }

  }
}
```

# Esempi da scaricare

- Scaricare nella directory ~/tomcat/webapps/ il pacchetto Immagini.tgz dalla pagina web del corso

- Scompattare il pacchetto: tar xzvf Immagini.tgz

- Si otterrà la directory classes contenente:
  - **la Servlet ServletStore per il caricamento delle immagini nella base di dati e la Servlet ServletShow per la visualizzazione di immagini provenienti dalla base di dati**

- Per poter compilare le classi è necessario includere nel classpath anche la libreria cos.jar che si trova nella directory lib/ del pacchetto (in cui poi è comunque necessario il link simbolico a postgres-jdbc3 come già visto)