

# JDBC

---

## Programmazione in Ambienti Distribuiti

V 1.3 © Marco Torchiano 2005

## Uso di JDBC

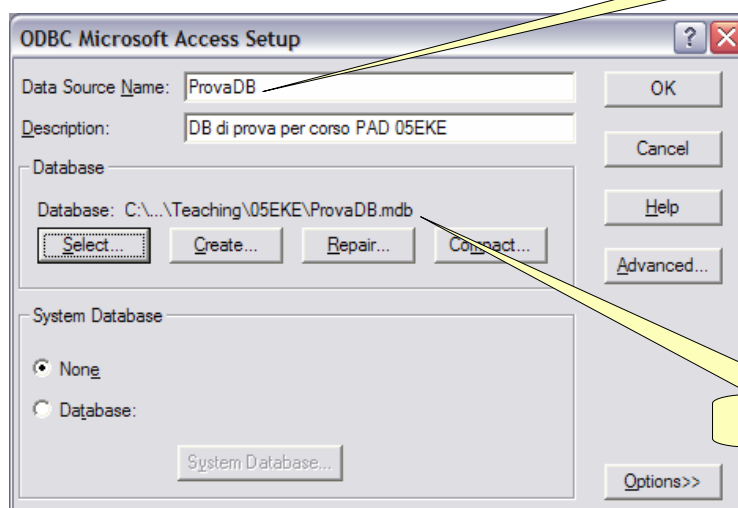
---

- Caricamento della classe driver
    - ◆ `Class.forName()`
  - Apertura della connessione
    - ◆ `DriverManager.getConnection()`
  - Creazione di uno statement
    - ◆ `createStatement()`
  - Esecuzione della query / istruzione SQL
    - ◆ `executeQuery()` / `executeUpdate()`
  - Scansione del `ResultSet`
-

## JDBC - ODBC

- Per usare un database che fornisce un'interfaccia ODBC, tramite JDBC occorre:
- Dichiarare il DB come data source ODBC
  - ♦ Pannello di controllo; Amministrazione; Data Sources (ODBC)
    - /system32/odbcad32.exe
  - ♦ System DSN; Add...
  - ♦ Scegliere il driver (es. MS Access Driver)
  - ♦ Selezionare il DB
- Fare riferimento a quel nome da Java

## ODBC MS Access



## JDBC

---

```
import java.sql.*;
class ProJdbc {
public static void main(String[] pars)
                                throws Exception{
// Carica il driver: bridge Jdbc-Odbc
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
// Ottiene una connessione con il DB
Connection conn = DriverManager.
    getConnection("jdbc:odbc:NOME_ODBC");
// dove "NOME_DNS" è il nome con cui
// il DB è stato dichiarato in ODBC
```

Diverso per  
servlet / JSP

## JDBC

---

```
// Crea uno statement
Statement stmt = conn.createStatement();
// Esegue una query
ResultSet rs = stmt.
    executeQuery("SELECT * FROM Tab1");
// Raccoglie le meta-informazioni sulla query
// (nomi e tipi dei campi)
ResultSetMetaData fields = rs.getMetaData();
int n_col = fields.getColumnCount();
for(int i=0; i<n_col; ++i){

System.out.println(fields.getColumnName(i+1)
    + ":" + fields.getColumnTypeName(i+1));
}
```

## JDBC

---

```
// Scandisce le righe del result set
while(rs.next()){
for(int i=0; i<n_col; ++i){
    String col_name= fields.getColumnName(i+1);
    // getString() converte a String qualsiasi
    // valore; getInt(), getDouble(), etc.
    // richiedono il tipo corretto
    String col_value = rs.getString(col_name);
    System.out.print(col_value+"\t");
}
System.out.print("\n");
}
}}
```

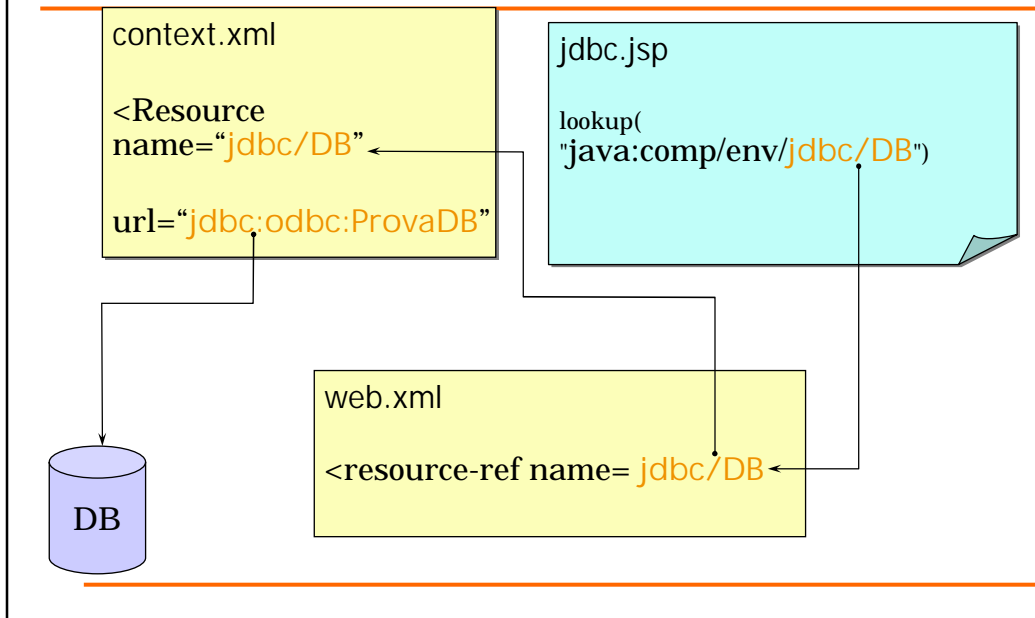
---

## Jdbc in Servlet e JSP

---

- Per utilizzare sorgenti dati Jdbc in applicazioni web occorre
    - ♦ Definire il contesto e dichiarare una risorsa di tipo data source
      - META-INF/context.xml
      - N.B. specifico per Tomcat
    - ♦ Definire un riferimento a tale risorsa
      - WEB-INF/web.xml
    - ♦ Accedere al nome dal servlet
-

## Jdbc in servlet e JSP



## Data Source del Contesto

- All'interno del Context creare una nuova Data Source
- Nome: **jdbc/DB**
- URL: **jdbc:odbc:ProvaBC**
- Driver: **sun.jdbc.odbc.JdbcOdbcDriver**

## context.xml

---

```
<Context path="/Jdbc" doccBase="Jdbc" debug="0">
  <Resource name="j dbc/DB" auth="Contai ner"
            type="j avax. sql . DataSource"/>
  <ResourceParams name="j dbc/DB">
    <parameter>
      <name>dri verCl assName</name>
      <val ue>sun. j dbc. odbc. JdbcOdbcDri ver</val ue>
    </parameter>
    <parameter>
      <name>url </name>
      <val ue>j dbc: odbc: ProvaDB</val ue>
    </parameter>
  </ResourceParams>
</Context>
```

---

## In web.xml

---

- Dichiarare l'uso della data source.
  - Dopo le direttive servlet-mapping:

```
<resource-ref>
  <descri pti on>DB Connecti on</descri pti on>
  <res-ref-name>j dbc/DB</res-ref-name>
  <res-type>j avax. sql . DataSource</res-type>
  <res-auth>Contai ner</res-auth>
</resource-ref>
```
-

# Deployment

file: //D: ... \web\META-INF\context.xml

Deploy

Deploy directory or WAR file located at

Context Path (optional):

XML Configuration file URL: ab\WEB-INF\context.xml

WAR or Directory URL: D:\workspace\Progetto\JSP\web

Deploy

file: //D: \workspace\Progetto\web

# JSP (1)

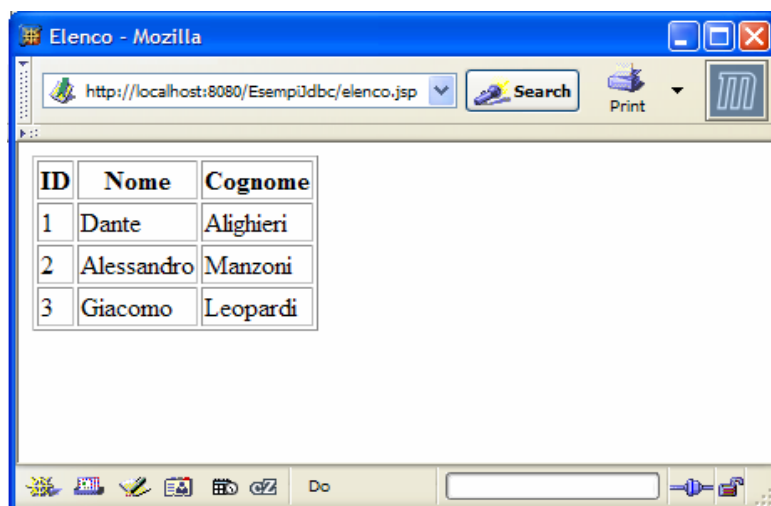
```
<%@ page contentType="text/html" language="java" %>
<%@ page import="javax.naming.*, javax.sql.*, java.sql.*"%>
<html><head><title>DB Test</title></head><body>
<%
    try{
        Context ctx = new InitialContext();
        if(ctx == null) throw new Exception("No Context");
        DataSource ds = (DataSource)
            ctx.lookup("java:comp/env/jdbc/DB");
        if (ds != null) {
            Connection conn = ds.getConnection();
            if(conn != null) {
                // .. Codice effettivo
            }else throw new Exception("No connection!");
        }else throw new Exception("No data source!");
    }catch(Exception e) { e.printStackTrace(); }
%>
</body></html >
```

Diverso per applicazioni

## JSP (2)

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from tab");
%> <table border=1>
  <tr> <%
    ResultSetMetaData fields = rs.getMetaData();
    int n_col = fields.getColumnCount();
    for(int i=0; i<n_col; ++i){
      %> <th><%=fields.getColumnName(i+1)%> <%
    }
    while(rs.next()){
      %><tr><%
        for(int i=0; i<n_col; ++i){
          String col_name = fields.getColumnName(i+1);
          %> <td><%= rs.getString(col_name) %> <%
        }
      }
    %></table> <%
  conn.close();
```

## Chiamata del JSP da browser



## Context Listener

---

- Permette di eseguire attività quando l'applicazione web viene attivata.
  - ♦ Quando il server si attiva

## Esempio

---

```
import javax.servlet.*;
public final class ContextListener
implements ServletContextListener {
    public void contextInitialized(
        ServletContextEvent event) {
        // inzializza risorse etc.
    }
    public void contextDestroyed(
        ServletContextEvent event) {
        // rialscia risorse etc.
    }
}
```

---

## web.xml

---

- Nel descrittore di applicazione bisogna elencare il listener

```
<listener>
```

```
<listener-class>
```

```
ContextListener</listener-class>
```

```
</listener>
```

---

## Esempio

---

- Il context listener definisce un contatore condiviso da tutte le istanze di JSP e servlet
    - ♦ Il contatore è un Integer
  - Un jsp legge il valore ed assegna un identificatore unico alla sessione
  - Un secondo jsp stampa il valore memorizzato nella sessione
-

## Esempio – Context Listener

---

```
import javax.servlet.*;
public final class ContextListener
implements ServletContextListener {
    public void contextInitialized(
        ServletContextEvent event) {
        System.out.println("ID initialized");
        event.getServletContext()
            .setAttribute("ID", new Integer(0));
    }
    public void contextDestroyed(
        ServletContextEvent event) {
        System.out.println("Ctxt destroyed");
    }
}
```

---

## Esempio – GetID.jsp

---

```
<%@ page contentType="text/html" language="java"%>
<html >
<head><title>Prova JSP</title></head>
<body><%
    Integer ID = (Integer)
    session.getServletContext().getAttribute("ID");
    int id = ID.intValue();
    session.setAttribute("ID", ID);

    session.getServletContext().setAttribute("ID", new
    Integer(id+1));
%>You've been assigned id <%=id%>. <br>
Go and see it <a href="PrintID.jsp">here</a>
</body></html >
```

---

## Esempio – PrintID.jsp

---

```
<%@ page contentType="text/html" language="java" %>
<html >
<head><title>Prova JSP</title></head>
<body><%
    Integer ID = (Integer)session.getAttribute("ID");
    if(ID==null){
        %>You don't have any ID!
        First go <a href="GetID.jsp">here</a> to get
        one. <%
    }else{
        int id = ID.intValue();
        %> You've been assigned id <%=id%>.
    <%}%>
</body></html >
```

---