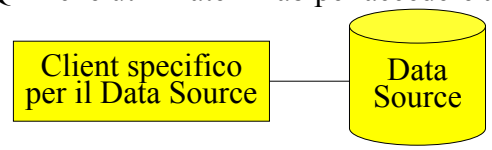
Accesso a Database Relazionali - I

- Produttori di RDBMS offrono **Application Programming Interface (API) per accedere ai dati gestiti dal DBMS**
- Accesso a DB basato su modello client-server ⇒ client accede a DB mediante le API definite dai produttori. Per esempio:
  - client a finestre di Microsoft Access
  - Client SQL+ che utilizzate in lab per accedere a DB Oracle



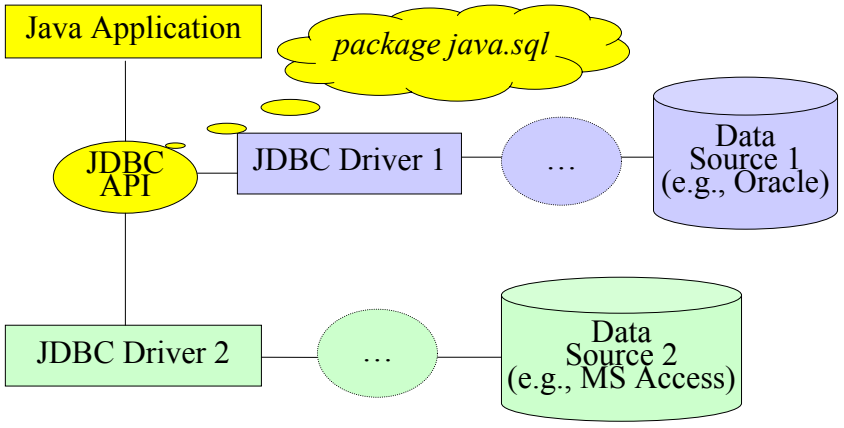
Java Database Connection (JDBC) - I

- **RDBMS diversi hanno API diverse** ⇒ serve strumento per generalizzare l'accesso ai dati
- In questo modo il codice dell'applicazione puo' essere indipendente dal particolare DB utilizzato
- **JDBC** implementa un'**interfaccia standardizzata tra applicazioni Java ed i database relazionali**

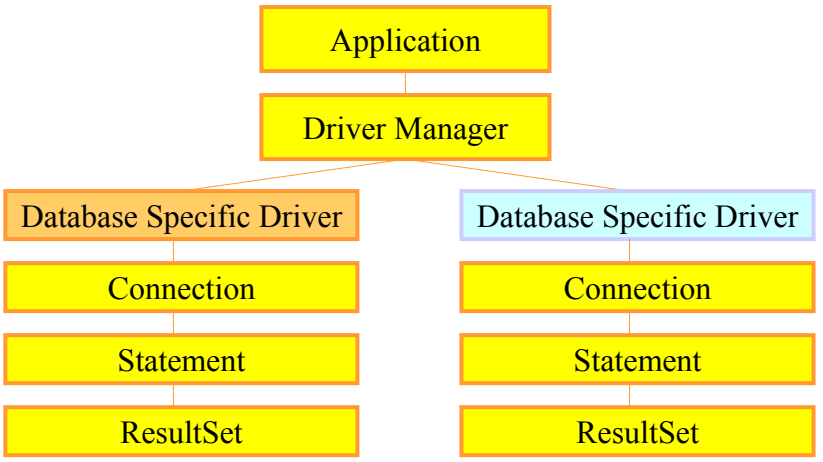
Java Database Connection (JDBC) - II

- permette l'**esecuzione di query SQL** e **analisi dei risultati** delle **query**
- astrae da
  - **dettagli implementativi del database** (MS Access? Oracle? MySql …?) e dei suoi **API**
  - **dettagli di comunicazione** con il database (tipo di connessione, etc.)

JDBC – accesso a DB diversi da applicazione Java



Classi e Interfacce di JDBC



Esempio di tabella

Tabella CUSTOMER: gia' definita e popolata nel DB "sample" di NetBeans

La utilizziamo per vedere come si effettua un'interrogazione a DB

NAME	**CITY**	**…**	**…**	**…**
JumboCom	Fort Lauderdale	…	…	…
Livermore Enterprises	Miami	…	…	...
…	…	…	…	…

Tabella e risultato di query – esempio

CUSTOMER

NAME	CITY	…	…	…
JumboCom	Fort Lauderdale	…	…	…
Livermore Enterprises	Miami	…	…	...
…	…	…	…	…

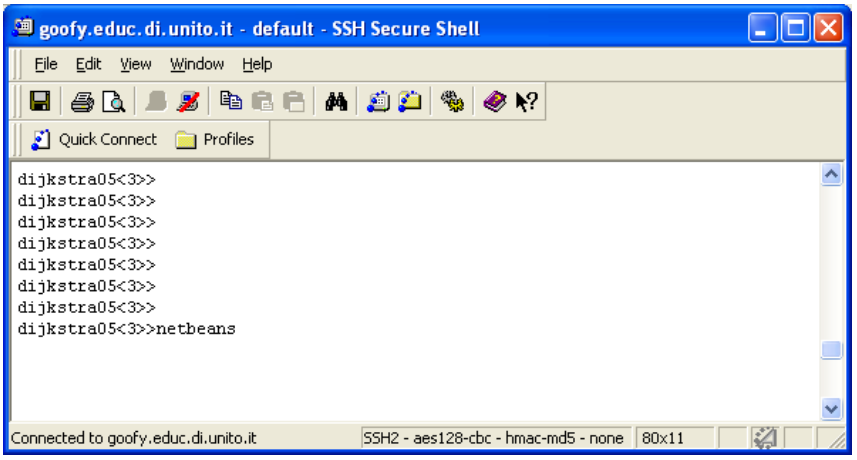
Risultato di query `("SELECT NAME, CITY FROM CUSTOMER")`

NAME	CITY
JumboCom	Fort Lauderdale
Livermore Enterprises	Miami
…	…

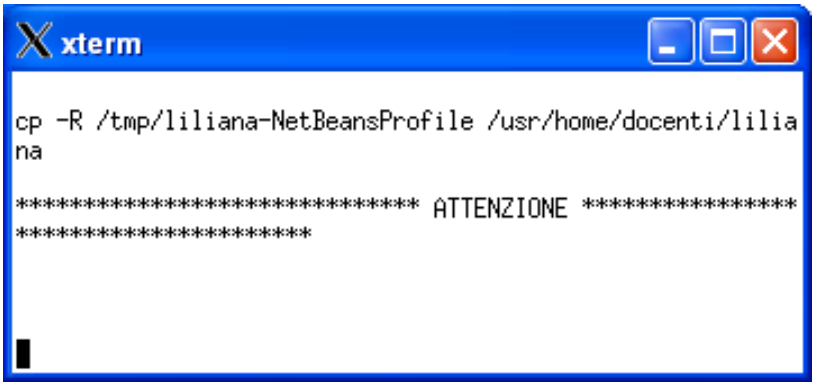
Interrogazione a DB (metodo di una classe java stand-alone)

```
public static String queryDB(String url, String user, String pwd) {
  String out = "";
  try {
    DriverManager.registerDriver(new
                                  org.apache.derby.jdbc.ClientDriver());
    Connection conn = DriverManager.getConnection(url,user,pwd);
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery("SELECT * FROM CUSTOMER");
    while (rs.next()) {
       out = out + "Nome = "+ rs.getString("NAME") + "\n"; }
    rs.close();
    st.close();
    conn.close();
    } catch (SQLException e) {out= e.getMessage();}
  return out;
 }
```

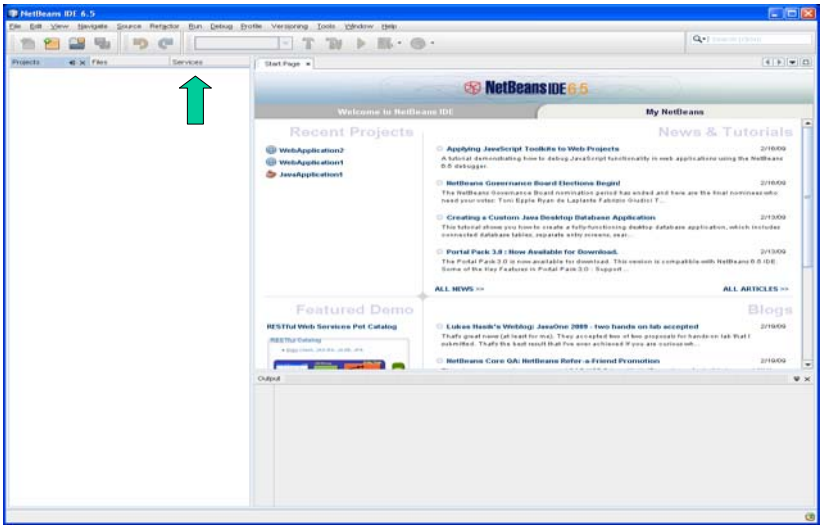
Lanciare NetBeans: aprire un Xterm e digitare netbeans ...



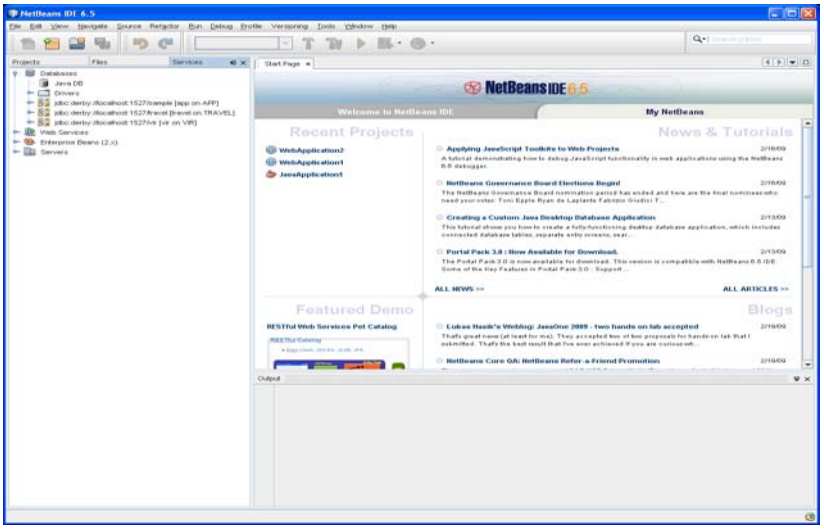
Lanciare NetBeans: ... ignorare il messaggio che viene visualizzato e ATTENDERE SENZA FARE NULLA



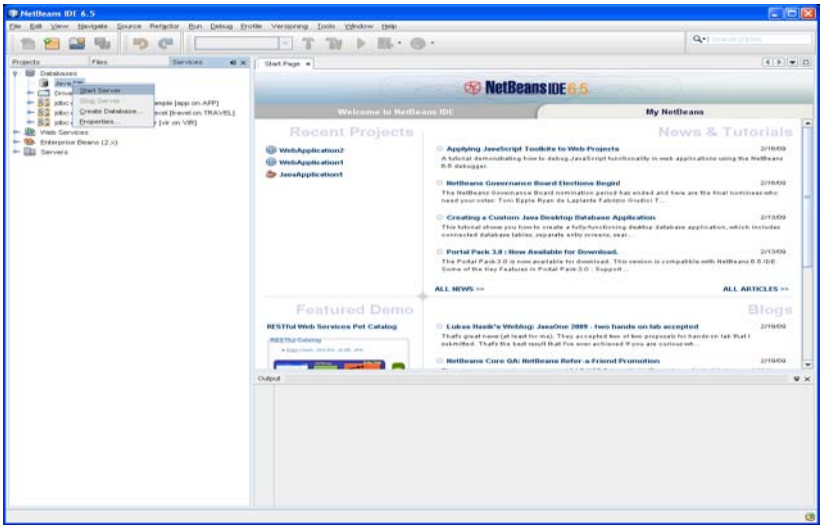
Pagina iniziale – cliccare su tab Services



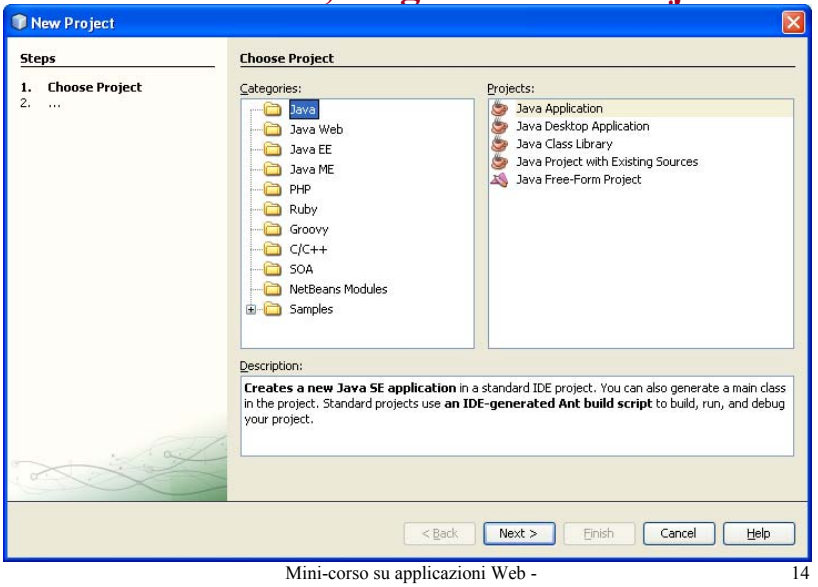
Aprire la cartella Databases



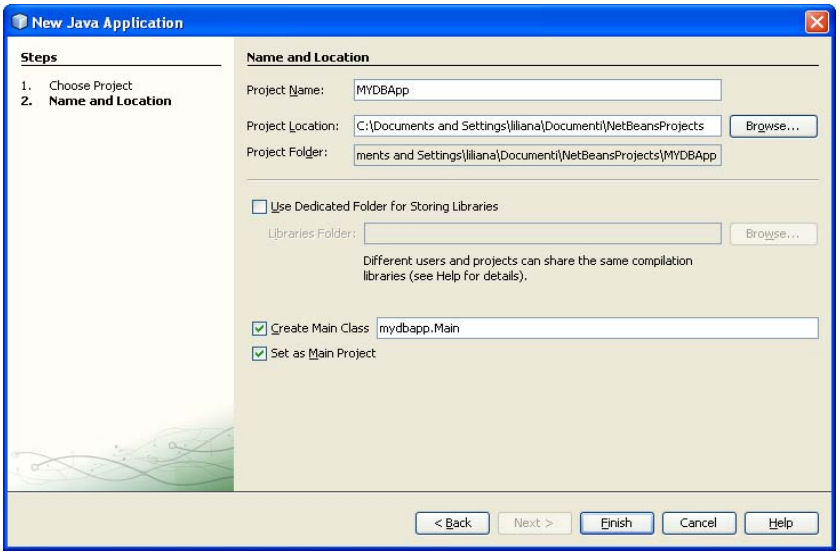
Con il tasto destro del mouse cliccare su Java DB e scegliere Start Server



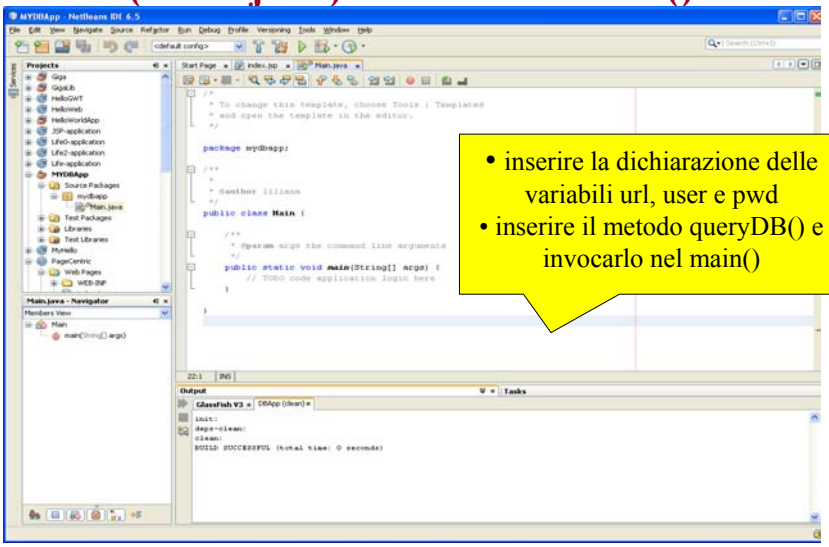
MYDBApp – creo progetto (Java) da menu File, scegliere New Project



MYDBApp – imposto il nome del progetto



MYDBApp – "nasce" la classe principale (Main.java) con metodo main()

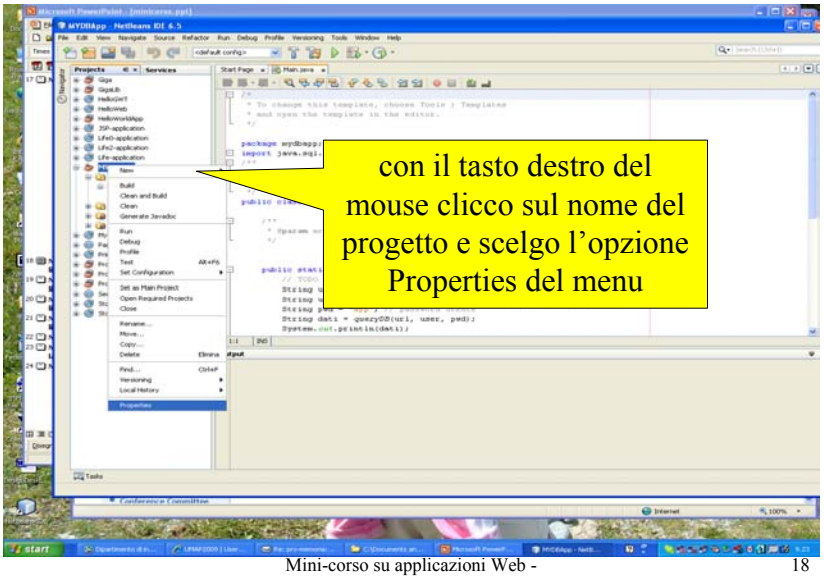


MYDBApp - codice

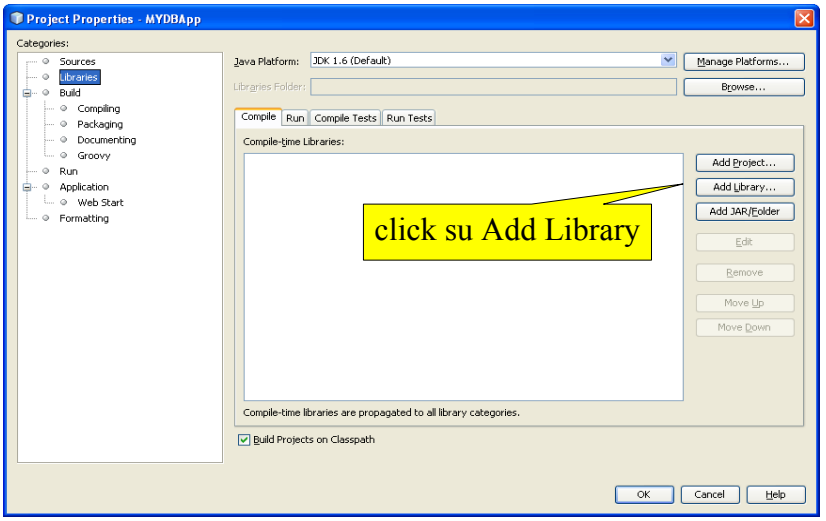
```
package mydbapp;
import java.sql.*;

public class Main {

public static void main(String[] args) {
    String url = "jdbc:derby://localhost:1527/sample"; // url del DB
    String user = "app"; // login utente del DB
    String pwd = "app"; // password utente
    String dati = queryDB(url, user, pwd);
    System.out.println(dati);
 }

  public static String queryDB(String url, String user, String pwd) {
     ... qui inserire il codice del metodo mostrato a pagina 8 ... }
} // fine classe Main
```

NetBeans segnala errore: mancano le librerie del driver JDBC da usare!

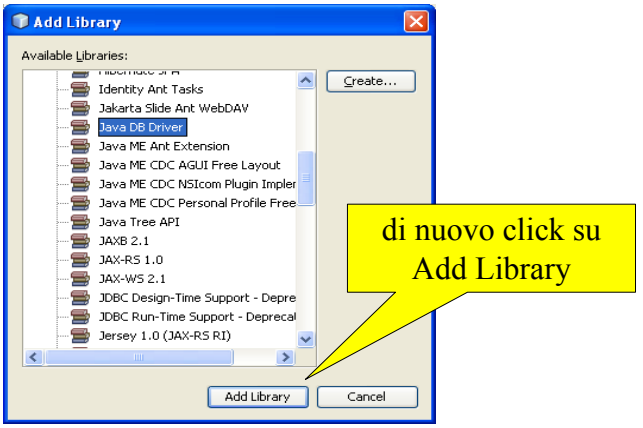
MYDBApp – aggiungo le librerie mancanti



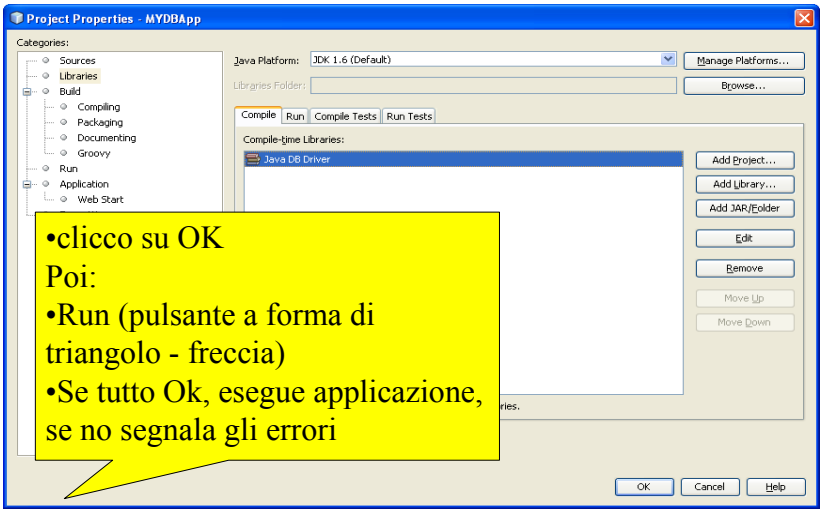
MYDBApp – aggiungo a properties le librerie del Java DB Driver - I



MYDBApp – aggiungo a properties le librerie del Java DB Driver - II



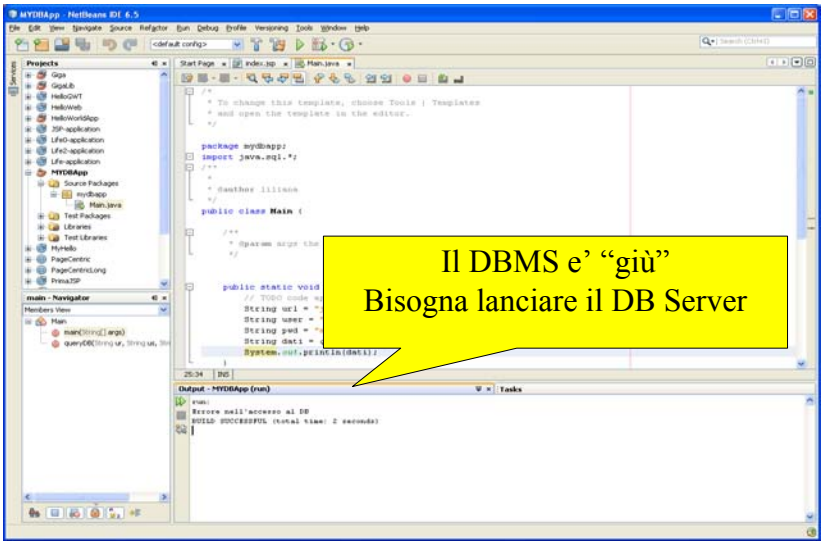
MYDBApp – aggiungo a properties le librerie del Java DB Driver - III



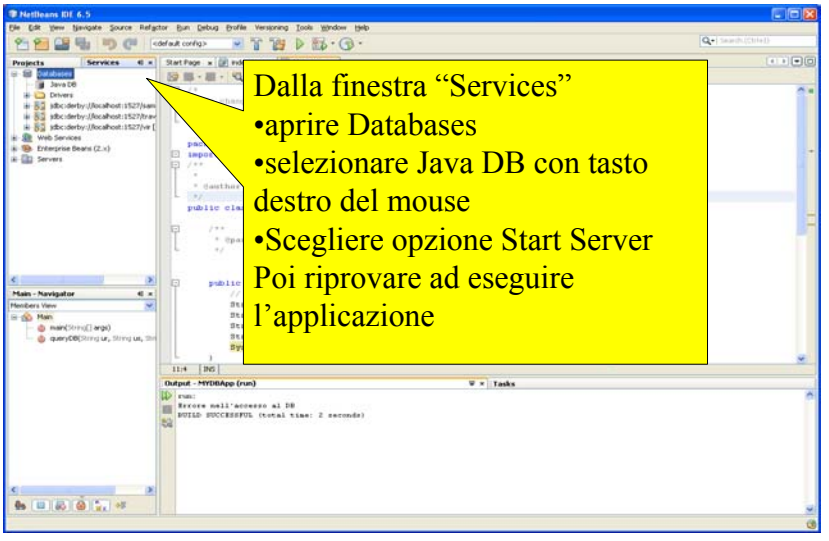
MYDBApp - RUN – output generato a video

```
run:
Nome = JumboCom
Nome = Livermore Enterprises
Nome = Oak Computers
Nome = Nano Apple
Nome = HostProCom
Nome = CentralComp
Nome = Golden Valley Computers
Nome = Top Network Systems
Nome = West Valley Inc.
Nome = Ford Motor Co
Nome = Big Car Parts
Nome = New Media Productions
Nome = Yankee Computer Repair

BUILD SUCCESSFUL (total time: 0 seconds)
```

MYDBApp – SOLO se l'accesso a DB da' eccezione ...



MYDBApp – SOLO se l'accesso a DB da' eccezione ... lancio il DB Server



World Wide Web

- Basata su modello **client-server**
- **Lato Client**
  - WWW appare come enorme collezione di pagine (statiche o dinamiche), accessibili usando un **browser**
  - pagine web legate mediante link
- **Lato Server**
  - Ogni sito Web ha processo server che ascolta su porta TCP per ricevere richieste di connessione e gestirle

Web Server

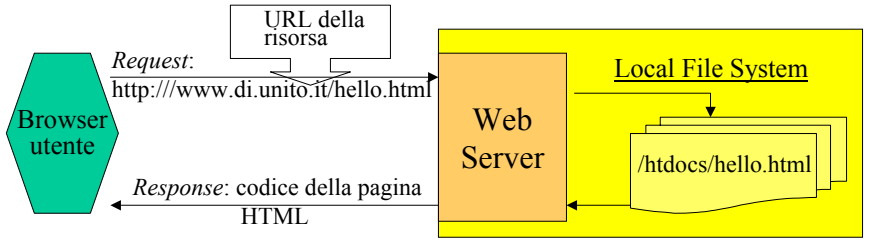
- Programma che
  - gira su una macchina server accessibile da internet
  - resta in ascolto di richieste (da parte di client web) su porta HTTP. E.g., http://www.di.unito.it:8080, http://localhost:4848
  - gestisce le richieste che arrivano (recupera pagina HTML richiesta, esegue programmi, invoca programmi associati a servizi richiesti, etc.)
  - restituisce a client una risposta (risultato della richiesta, messaggio di errore)
- Web server più comuni
  - Apache (disponibile per sistemi operativi diversi)
  - Internet Information Server (IIS di Microsoft, per Windows)

Web Browser

- **Interfaccia utente universale**
  - Invoca Web Server per richiedere pagine statiche o dinamiche
  - Riceve dati da Web Server e li presenta sulla macchina dell'utente
  - Può eseguire script (e.g., javascript) localmente, per effettuare semplici computazioni
  - Può scaricare plug-in da Web Server per eseguire programmi localmente (e.g., animazioni Flash)

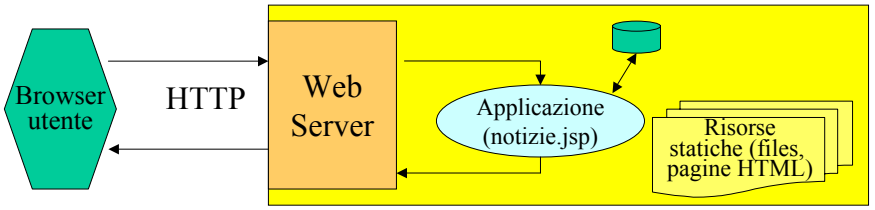
Accesso a pagina HTML statica

- la pagina HTML (risorsa Web) e' memorizzata su file system e recuperata dal Web Server



Accesso a pagina dinamica

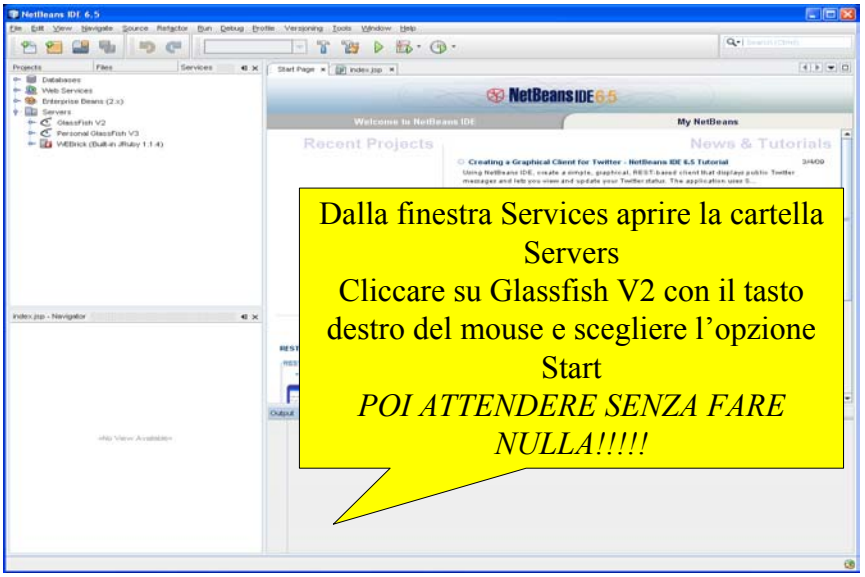
- la pagina dinamica (risorsa Web) viene generata dinamicamente da un'applicazione che gira sul server
- es: http://news.com/notizie.jsp



Esecuzione di Applicazioni Web

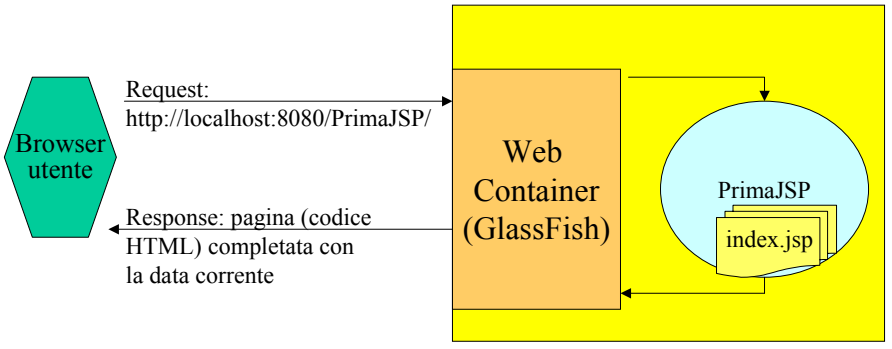
- Per far girare un'applicazione Web basata su java serve:
  - Un **Web server** (che cattura richieste HTTP e invia le risposte)
  - Un **Web Container**: **java runtime** che ospita ed esegue le applicazioni web
  - In questo laboratorio, **GlassFish V2** svolge entrambe le funzioni
  - Glassfish e' interno a NetBeans, gira localmente al vostro computer – **localhost**

NetBeans: lanciare application server Glassfish V2

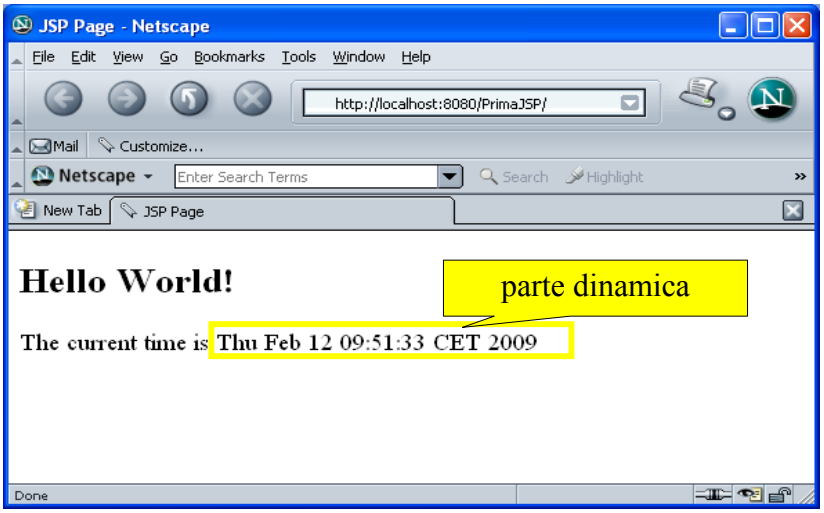


Applicazioni Web basate su pagine dinamiche

- Un'applicazione Web "page centric" e' costituita di una o piu' pagine statiche/dinamiche
- Per es. PrimaJSP e' costituita dalla sola index.jsp



Esecuzione dell'applicazione Web PrimaJSP

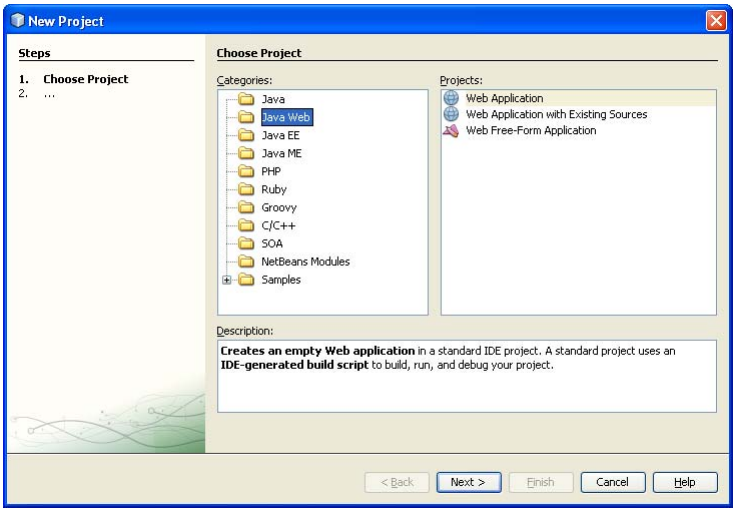


PrimaJSP: index.jsp - codice

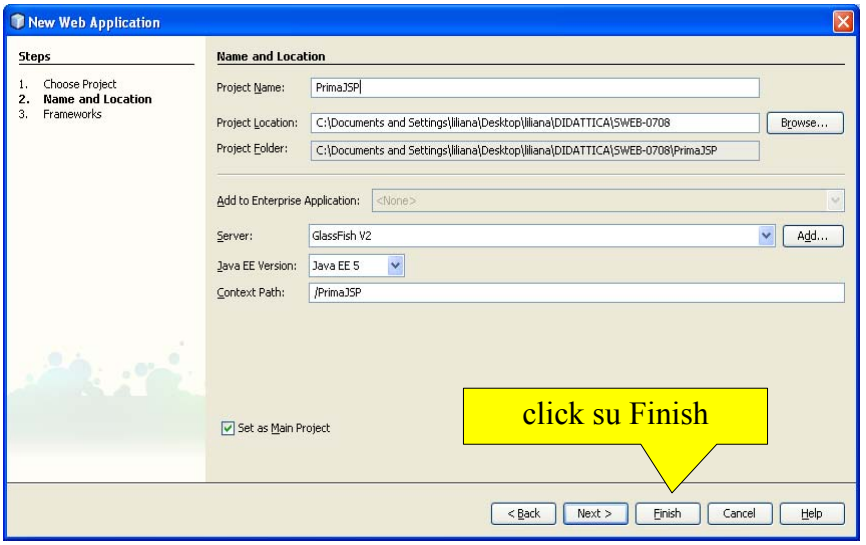
```
<%@page contentType="text/html" pageEncoding="UTF-8"
    import="java.util.Date"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h2>Hello World!</h2>
    The current time is <%= (new Date()).toString() %>
    <br>
  </body>
</html>
```

Creazione di progetto Web

- Da Menu File ⇒ New Project ⇒ Java Web

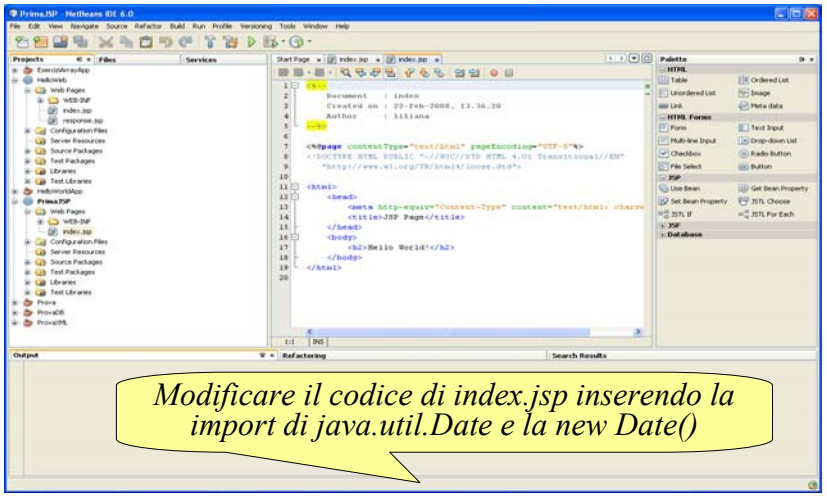


Creazione di progetto Web (PrimaJSP)

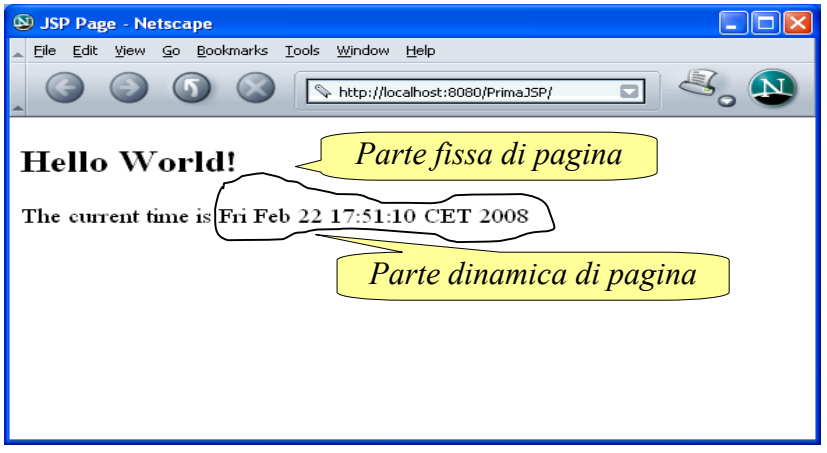


Nasce progetto PrimaJSP con JSP index.jsp

index.jsp e' la pagina principale del progetto – caricata quando si invoca l'applicazione



Esecuzione di index.jsp – TASTO RUN



Scripting elements

- Permettono di inserire codice java in pagina JSP
  - dichiarazioni di **variabili**
  - dichiarazioni di **metodi**
  - **espressioni:** valutazione di variabili, espressioni aritmetiche, invocazione di metodi
  - **scriptlets**: blocchi di codice eseguiti quando client richiede pagina JSP

```
<%!  int anno = 2003;
       public String saluta(String name) {
           return "Ciao, " + name + "!!"; }
%>
<html><head>…</head>
<body>
<p> Ciao <%= saluta("carissimo")%>, siamo nel <%= anno %>
</body>
</html>
```

Dichiarazioni di variabili e metodi

Espressione

Espressione

SecondaJSP: accesso a DB da JSP

SecondaJSP: caratteristiche

- Applicazione SecondaJSP: contiene solo index.jsp, che fa una query al DB "sample"
- usiamo JDBC per accedere a DB da ambiente java
- nella JSP:
  - importare java.sql
  - scrivere il metodo che interroga il DB e restituisce il risultato
  - invocare il metodo nella JSP, come espressione

SecondaJSP - index.jsp – codice - I

```
<%@page contentType="text/html" ...  import="java.sql.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 ...>
<%!
   private String url = "jdbc:derby://localhost:1527/sample";
   private String user = "app";  private String pwd = "app";

   public String queryDB(String ur, String us, String p) {
    String out = "";
    try {
     DriverManager.registerDriver(new org.apache.derby.jdbc.ClientDriver());
     Connection conn = DriverManager.getConnection(ur, us, p);
     Statement st = conn.createStatement();
     ResultSet rs = st.executeQuery("SELECT * FROM CUSTOMER");
     while (rs.next()) {
         out = out + "Nome = "+ rs.getString("NAME") + "<br>"; }
      rs.close(); st.close(); conn.close();
      } catch (SQLException e)  {out= "Errore nell'accesso al DB";}
    return out;
   }
%> ... continua ...
```

variabili

metodo

SecondaJSP - index.jsp – codice - II

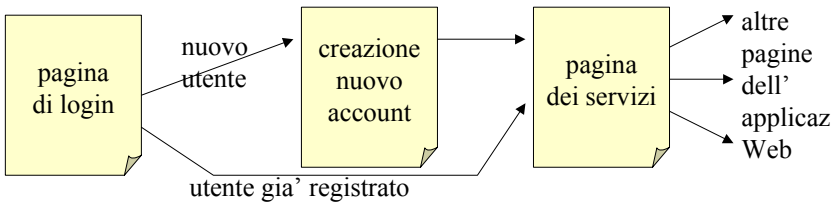
... continua ...

```
<html>
   <head><meta http-equiv="Content-Type" content="text/html;
    charset=UTF-8"> <title>Informazioni</title></head>
   <body>
      <h2>Informazioni sulle aziende:</h2>
      <p>
      <%= queryDB(url, user, pwd)%>
      </p>
   </body>
</html>
```

invocazione del metodo

Applicazioni Web “page centric”

In generale, sono composte da piu’ pagine (statiche e dinamiche) che rappresentano la sequenza di pagine da visualizzare durante l’interazione con l’utente. Ogni pagina include l’invocazione alla pagina successiva. *Es: flusso delle pagine di un’applicazione tipica*

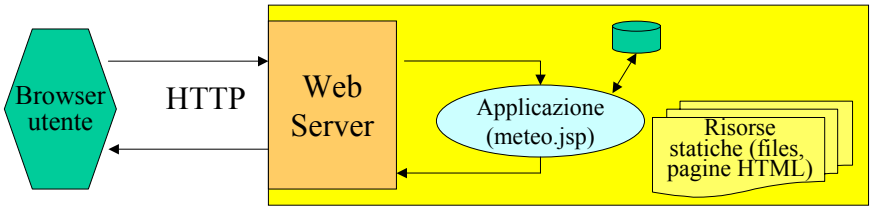


Accesso a pagina dinamica

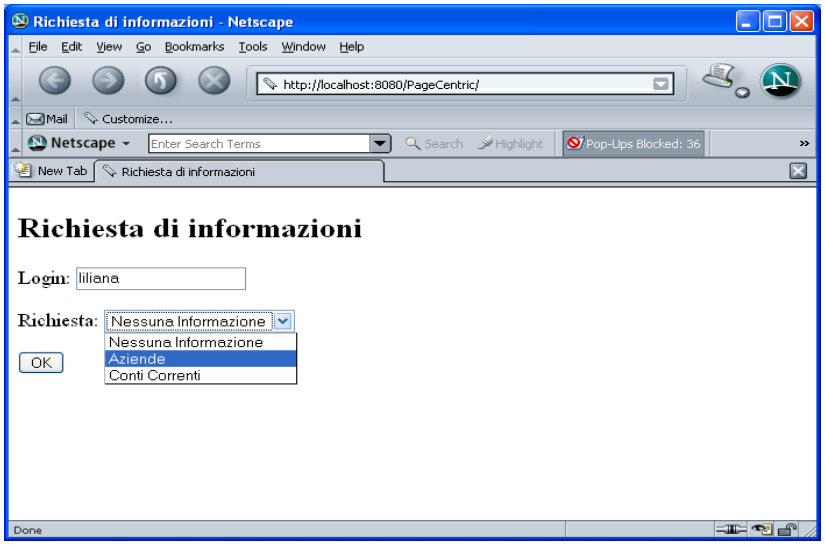
- NB: la richiesta HTTP tipicamente contiene parametri che ne specificano il contenuto. Es:

  http://www.esempio.com/meteo.jsp?regione=Piemonte

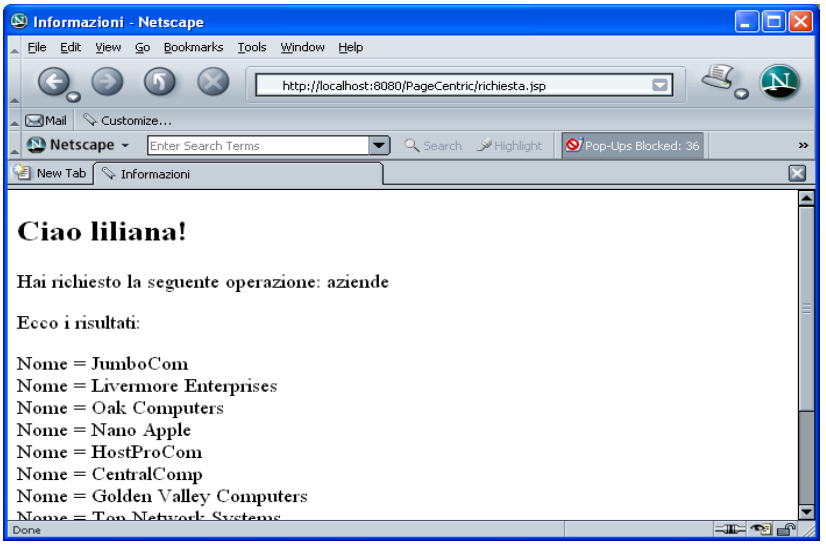
  http://servizi.com/orariTreni.jsp?citta=Torino;stazione=PortaNuova



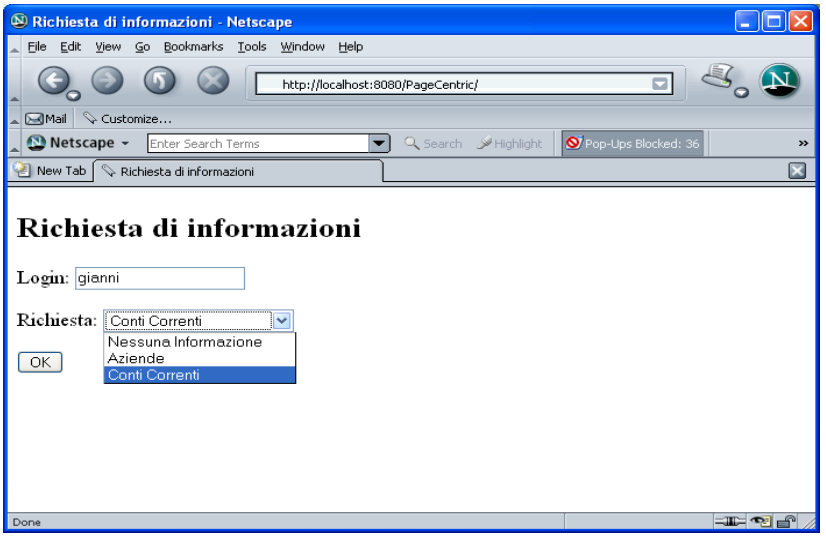
PageCentric – pagina 1 (index)



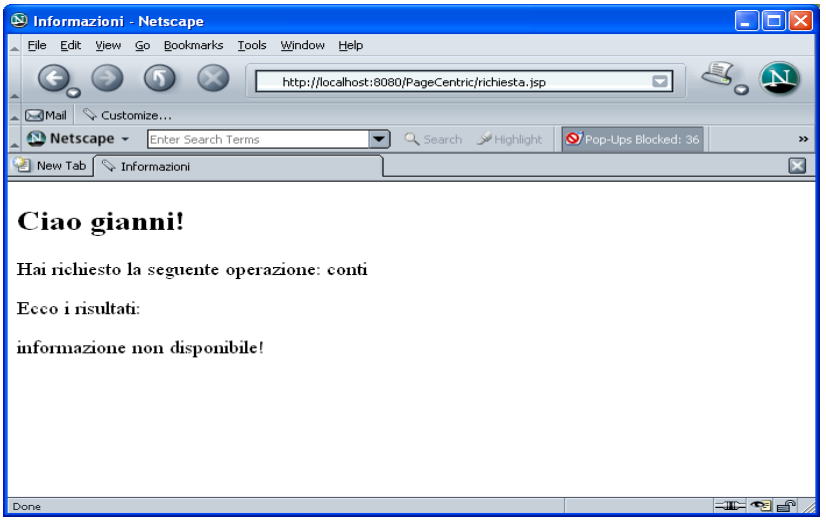
PageCentric – pagina 2 (richiesta.jsp)



PageCentric – pagina 1



PageCentric – pagina 2



PageCentric – index.jsp (statica)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head><title>Richiesta di informazioni</title><meta http-equiv="Content-Type"
    content="text/html; charset=UTF-8"></head>
 <body>
    <h2>Richiesta di informazioni</h2>
    <form action="richiesta.jsp" method="POST">
    <p>Login: <input type="text" name="login" value="ospite"></p>
    <p>Richiesta:
     <select name="tipoInfo">
          <option value="nessuna" selected>Nessuna Informazione</option>
          <option value="aziende">Aziende</option>
          <option value="conti">Conti Correnti</option>
    </select></p>
    <p><input type="submit" name="submit" value="OK"></p>
 </body>
</html>
```

PageCentric – richiesta.jsp – parte 1

```
<%@page contentType="text/html" pageEncoding="UTF-8" import="java.sql.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<%!
   private String url = "jdbc:derby://localhost:1527/sample"; // url del DB
   private String user = "app";  private String pwd = "app"; // login e password utente
   public String queryDB(String ur, String us, String p) {
    String out = "";
    try {
          DriverManager.registerDriver(new org.apache.derby.jdbc.ClientDriver());
          Connection conn = DriverManager.getConnection(ur, us, p);
          Statement st = conn.createStatement();
          ResultSet rs = st.executeQuery("SELECT * FROM CUSTOMER");
          while (rs.next()) {
             out = out + "Nome = "+ rs.getString("NAME") + "<br>"; }
          rs.close(); st.close(); conn.close();
      } catch (SQLException e) {out= "Errore nell'accesso al DB";}
    return out;
   }  ... continua ...
```


PageCentric – richiesta.jsp – parte 2

```
   public String analyzeRequest(String op) {
         String ris = "";
         if (op!=null) {
            if (op.equals("aziende"))
               ris = queryDB(url, user, pwd);
            else if (op.equals("conti"))
                ris = "informazione non disponibile!";
                else ris = "non hai chiesto nessuna informazione";
             }
         else ris = "non e' stata selezionata alcuna operazione";
         return ris;
         }
   %>
   ... continua ...
```

PageCentric – richiesta.jsp – parte 3

```
<html>
  <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>Informazioni</title></head>
  <body>
     <% String op = request.getParameter("tipoInfo"); %>

     <h2>Ciao <%=request.getParameter("login")%>!</h2>
     <p>Hai richiesto la seguente operazione: <%= op %> </p>
     <p>Ecco i risultati:
     </p>
     <p>
     <%= analyzeRequest(op) %>
     </p>
  </body>
</html>
```

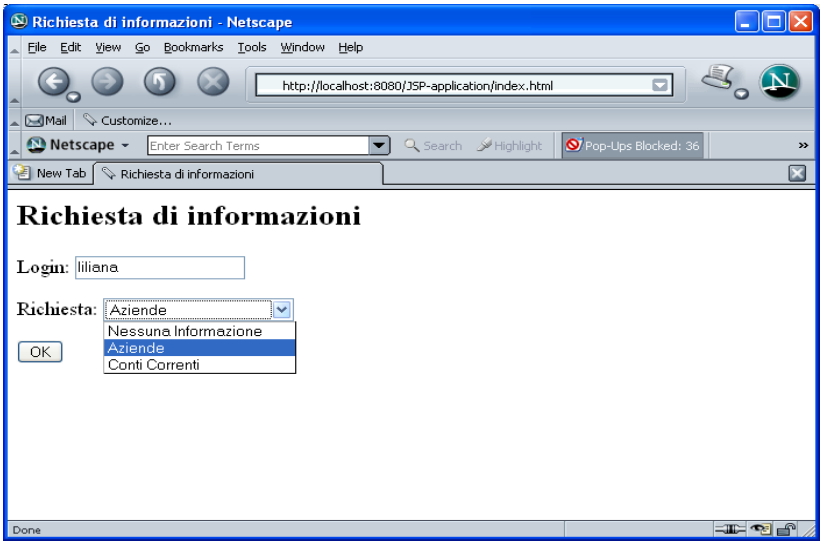

Problemi

- poca leggibilita' del codice (il codice java e' inserito direttamente nella JSP)
- come mantenere i dati dell'utente (per es., il suo login) nel caso l'applicazione abbia molte pagine? Bisogna gestire la sessione utente...

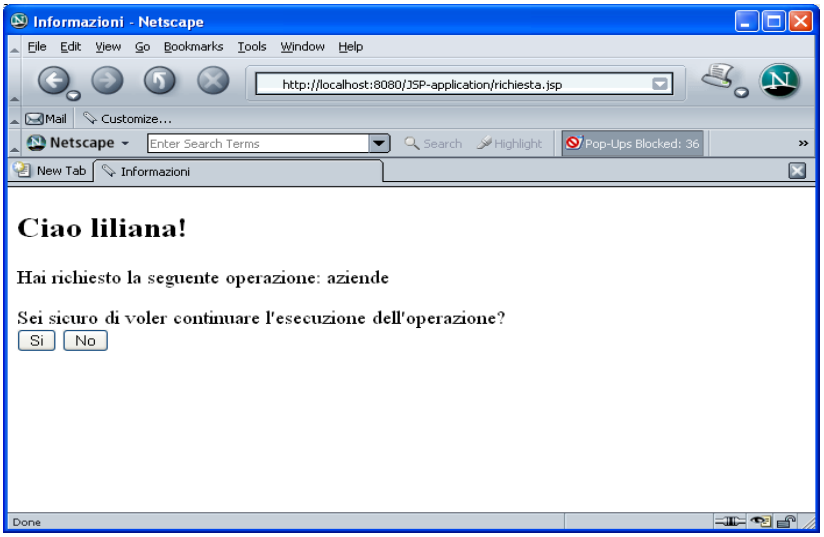
JavaBeans

- Java Beans: classi java che rispettano uno standard di definizione dei metodi e delle loro variabili di istanza (o di stato, dette *properties*)
  - per ogni variabile xxx di istanza (property)
    - public Type getXxx(), per leggere valore della property
    - public void setXxx(), per assegnare un valore alla property
  - si possono poi definire altri metodi getYyy() non associati a variabili di istanza: tali metodi sono utili per eseguire codice applicativo

Esempio: JSP-application – index.jsp



Esempio: JSP-application – richiesta.jsp



Esempio: JSP-application – gestioneRichiesta.jsp

JSP-application – richiesta.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 .../loose.dtd">
<jsp:useBean id="opb" scope="session" class="myBeans.OPBean" />
<jsp:setProperty name="opb" property="*" />
<html>
  <head><meta ..."> <title>Informazioni</title></head>
  <body>
    <form action="gestioneRichiesta.jsp" method="POST">
    <h2>Ciao <jsp:getProperty name="opb" property="login"/>!</h2>
    <p>Hai richiesto la seguente operazione:
          <jsp:getProperty name="opb" property="tipoInfo"/>
    </p>
    <p>Sei sicuro di voler continuare l'esecuzione dell'operazione? <br>
      <input type="submit" name="confirm" value="Si">
      <input type="submit" name="confirm" value="No">
    </p>
  </body>
</html>
```


JSP-application – gestioneRichiesta.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<jsp:useBean id="opb" scope="session" class="myBeans.OPBean" />
<html>
  <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Informazioni</title></head>
  <body>
    <h2>Sei collegato come <jsp:getProperty name="opb" property="login"/></h2>
    <% String info = opb.getTipoInfo(); %>
    <% if (request.getParameter("confirm").equals("Si")) { %>
      <jsp:getProperty name="opb" property="risultato"/>
    <% } else { %>
    Operazione annullata.
    <% } %>
  </body>
</html>
```

JSP-application – OPBean.java

```
package myBeans;
import java.sql.*;

public class OPBean {

   private String login;
   private String tipoInfo;

   private static final String url = "jdbc:derby://localhost:1527/sample";
   private static final String user = "app"; // login utente da usare per connettersi al DB
   private static final String pwd = "app"; // password utente


   public OPBean() {
      login = "sconosciuto";
      tipoInfo = "sconosciuto";
   }
   ... continua ...
```


JSP-application – OPBean.java

```
public void setLogin(String l) {
      login = l;
   }
   public String getLogin() {
      return login;
   }
public void setTipoInfo(String ti) {
      tipoInfo = ti;
   }
   public String getTipoInfo() {
      return tipoInfo;
   }
   ... continua ...
```

JSP-application – OPBean.java

```
public String getRisultato() {
      String out = "";
      if (tipoInfo.equals("nessuna"))
         out = "Non e' stata richiesta alcuna informazione";
      else if (tipoInfo.equals(“conti"))
         out = "Informazioni sui conti correnti non disponibili";
      else if (tipoInfo.equals(“aziende"))
            out = getDatiDaDB();
      else out = "Tipo di informazione non disponibile";
      return out;
   }
```

... continua ...

JSP-application – OPBean.java

```
public String getDatiDaDB() {
   String out = "";
   try {
        DriverManager.registerDriver(new org.apache.derby.jdbc.ClientDriver());
        Connection conn = DriverManager.getConnection(url, user, pwd);
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM CUSTOMER");
        while (rs.next()) {
           out = out + "Nome = "+ rs.getString("NAME") + "<br>"; }
        rs.close();
        st.close();
        conn.close();
     } catch (SQLException e)  {out= "Errore nell'accesso al DB";}

   return out;
   }
}// end OPBean
```

FINE!!

Grazie per l'attenzione

Liliana Ardissono