

Database e Web



Applicazioni e DBMS

Un'applicazione (per l'accesso a database) contiene **tre elementi**

- un'interfaccia **utente**
- i **dati**
- una **logica**

Si può usare un'architettura

- a **due strati (2-tier)**
 - uno strato contiene la logica e l'interfaccia
 - uno strato contiene i dati
- a **tre strati (3-tier) o più' (n-tier)**
 - uno strato diverso per ciascun elemento

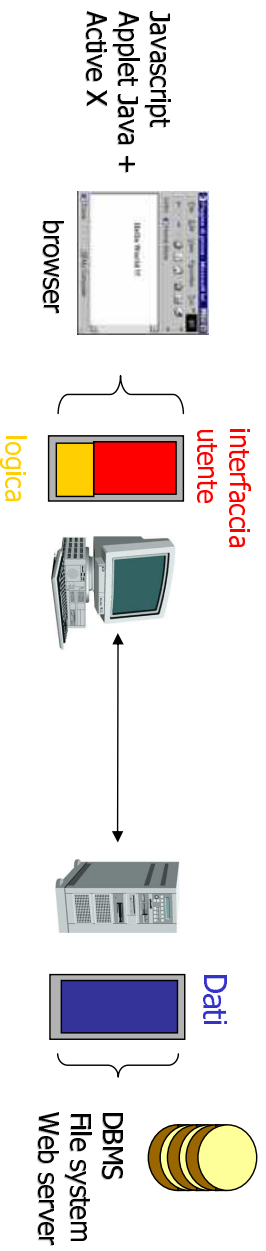
Architettura a due strati

Il client: **interfaccia utente e logica di controllo**

- Un'applicazione sviluppata con
 - un comune linguaggio di programmazione
 - browser + client side scripting (Javascript, Applet Java, Active X)

Il server: **i dati**

- un DBMS e il file system
- eventualmente un Web server (se e' un'applicazione Web)



Maggini, Scarselli

Sistemi per basi di dati

3

Architettura a due strati II

Il client accede ai dati attraverso

- API proprietarie del DBMS
- connettori standard (JDBC, ODBC, ...)

Osservazioni

- puo' anche accadere che parte della logica sia realizzata sul lato server
- l'architettura 2-tier
 - è semplice da realizzare
 - difficile da espandere
 - tipica delle applicazioni tradizionali

Maggini, Scarselli

Sistemi per basi di dati

4

Architettura a tre strati

Presentation tier

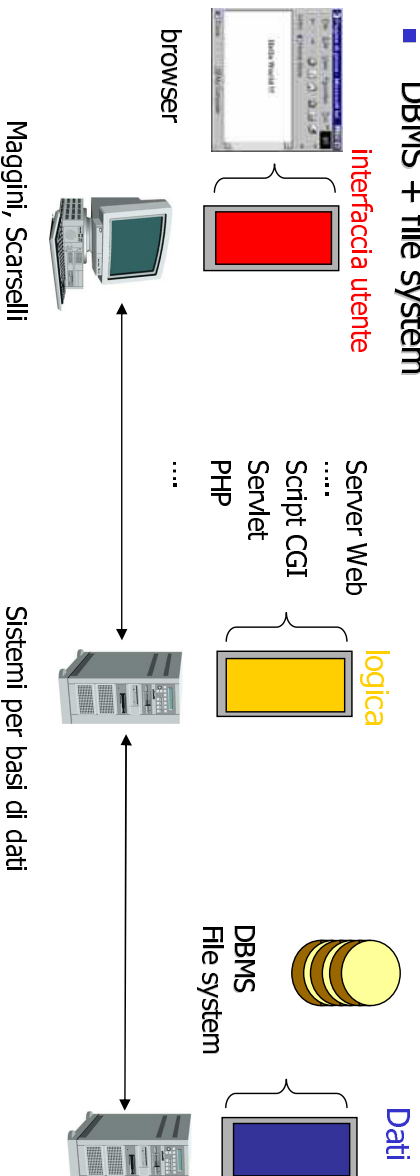
- un browser

Business logic tier (middle tier)

- un server Web, un transaction server, un application server.....
- server side logic (CGI, PHP, servlet, JSP,...)

Data tier

- DBMS + file system



5

Architettura a tre strati II

Osservazioni

- si possono avere architetture **n-tier**
- non è sempre semplice distinguere cosa deve essere messo su un certo livello

Vantaggi

- **separazione** logica di controllo da problematiche di presentazione
 - Si può cambiare la logica di controllo senza cambiare l'interfaccia
 - Si possono riusare gli stessi oggetti di controllo per applicazioni diverse
- la gestione dell'applicazione è **centralizzata**



Architettura a tre strati III

Vantaggi

- **thin client** (client leggeri)
 - semplificata amministrazione dei client (... solo un browser)
 - bassi costi dell'hardware per il client
- carico di lavoro facile da bilanciare

Svantaggi

- architettura più complessa
- si usano molti strumenti per realizzare la logica di controllo

Maggini, Scarselli

Sistemi per basi di dati

7



Web Information Systems

Web Information Systems (WIS)

- sistemi informativi accessibili attraverso un **browser**
- permettono un accesso "universale" all'informazione
- nati per il Web, sono usati anche
 - in **Intranet**
 - per realizzare applicazioni di ogni tipo
- strettamente legati ai DBMS
 - permettono di accedere l'enorme quantità di informazione preesistente nei database
 - man mano che diventano più complessi, parte dell'informazione originariamente memorizzata in pagine HTML viene spostata in database

Maggini, Scarselli

Sistemi per basi di dati

8



Accesso a database tramite Web

Vantaggi

- **indipendenza dalla piattaforma**
 - dei client e parzialmente dei server
 - supporto per l'accesso contemporaneo da piattaforme diverse
 - **visibilita' world wide**
- **interfaccia grafica**
 - **facile da realizzare**
 - **standard**
- **accesso alla rete trasparente**
- **applicazioni facilmente aggiornabili e scalabili**

Maggini, Scarselli

Sistemi per basi di dati

9



Accesso a database tramite Web II

Svantaggi

- **basso livello di**
 - **affidabilita'**
 - **sicurezza**
- **alto costo di sviluppo di un'applicazione**
- **l'HTML e HTTP sono poveri di funzionalita'**
 - es. connessioni statless, difficulta' nell'interagire con l'utente, ...
- **Lentezza**
 - Internet è lenta
 - Gli interpreti dei linguaggi usati sono lenti
- **strumenti di sviluppo immaturi**
- **scalare un sito non è banale, se un solo server Web non è sufficiente**

Maggini, Scarselli

Sistemi per basi di dati

10

Funzionamento di un server Web

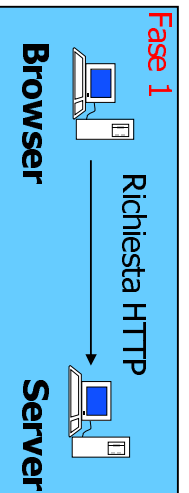
Quando si clicca sul link "http://www.ing.unisi.it/eventi.html"

Prima fase: il browser

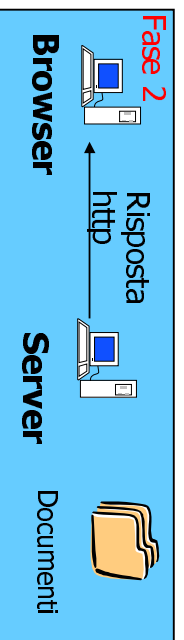
- prepara una richiesta secondo il protocollo HTTP
- apre una connessione con il server `www.ing.unisi.it` ed invia la richiesta per avere il documento `/eventi.html`

Seconda fase: il server

- prepara una risposta nel formato HTTP che contiene
 - il documento HTML in `/eventi.html` se esiste
 - oppure contiene un messaggio di errore
- invia la risposta e chiude la connessione



Maggini, Scarselli



Sistemi per basi di dati

11

Funzionamento di un server Web II

Terza fase: il browser

- legge la risposta estraendo il documento inviato dal server
- mostra a video il documento secondo regole che dipendono dallo specifico browser

Osservazioni

- il server Web è solo un "passa carte":
 - non fa altro che inviare i documenti richiesti
 - non conosce il contenuto dei documenti
- le connessioni sono stateless
 - per mantenere memoria dello stato occorre usare dei cookie
- I documenti HTML sono **statici**
 - occorre estendere HTML e/o i server Web se si vuole connettersi ad un DBMS



Estendere HTML e i server Web

Estensioni dei server Web

- **Common Gateway Interface (CGI), Java Servlets**
 - i documenti (HTML o altro) sono prodotti da (l'output) di programmi
- **Estensioni proprietarie dei Web server (Netscape API, IIS API)**
 - CGI e Servlets possono accedere a funzionalità messe direttamente a disposizione dal server Web
- **linguaggi di scripting** lato server (PHP, ASP, JSP)
 - si inserisce all'interno di HTML degli script che verranno eseguiti dal server Web (o da un'applicazione server) producendo documenti HTML puri

Estensioni di HTML

- **linguaggi di scripting** lato client (JavaScript, VBScript)
 - si inserisce all'interno di HTML degli script che verranno eseguiti dal browser
- **Java applet**
 - vere e proprie applicazioni che possono essere scaricate ed eseguite dal browser

Maggini, Scarselli

Sistemi per basi di dati

13



Common Gateway Interface (CGI)

- **è un protocollo**
 - consente al Web Server di eseguire programmi esterni in grado di produrre pagine dinamicamente
 - definisce un insieme di **variabili di ambiente utili** alla applicazione (es., i parametri inviati dal client)
- **l'applicazione**
 - può essere scritta in un **qualsiasi linguaggio** (C, Java, linguaggi per script)
 - gli eseguibili devono essere inseriti in una directory gestita dal Web Administrator (/cgi-bin)

Maggini, Scarselli

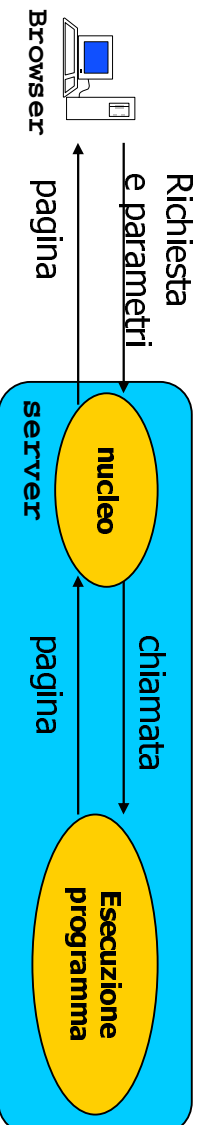
Sistemi per basi di dati

14

Common Gateway Interface II

- Il funzionamento

- il browser invia al server una richiesta in cui si fa riferimento ad un cgi es. www.search.com/search.cgi?Picasso
- il server chiama il programma "search.cgi"
- il parametro "Picasso" è passato attraverso lo standard input (metodo POST nel form) o settando una variabile d'ambiente (metodo GET)
- Il programma [search.cgi](#) produce in uscita un documento HTML e termina
- Il server Web invia il documento al browser



Maggini, Scarselli

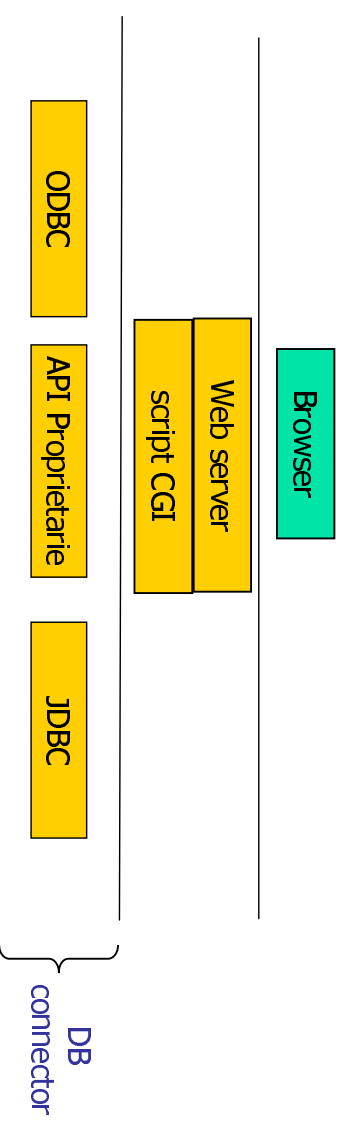
Sistemi per basi di dati

15

Common Gateway Interface III

- connessione ad un DBMS

- attraverso qualsiasi meccanismo disponibile nel linguaggio di programmazione prescelto
- ODBC, JDBC, API proprietarie



Maggini, Scarselli

Sistemi per basi di dati

16



DB connectors

API proprietarie

- moduli che forniscono le funzionalità necessarie a connettersi ed accedere ai dati di **uno specifico DBMS**
- realizzati dai produttori dei DBMS per i principali linguaggi

Connettori standard

- moduli che forniscono le funzionalità per connettersi a **molti DBMS**
- **JDBC (Java DataBase Connectivity)**
 - Package di Java (usabile solo con Java)
 - disponibili driver su tutte le piattaforme per i principali DBMS SQL
- **ODBC (Microsoft Open Database Connectivity)**
 - scritto in C++
 - adatto per client Windows
 - disponibili driver per i principali DBMS SQL

Maggini, Scarselli

Sistemi per basi di dati

17



Un esempio: JDBC

Quattro tipi di **driver JDBC**

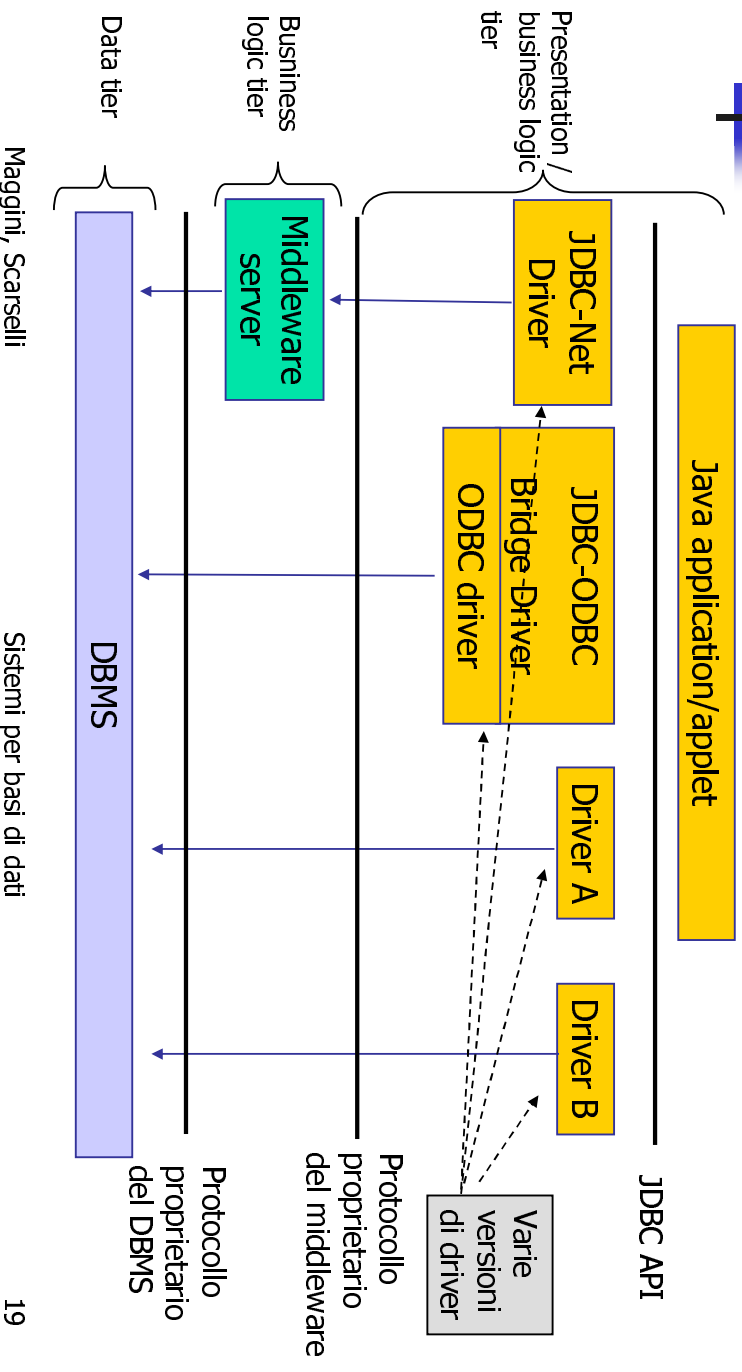
- **Tipo 1: Bridge ODBC più driver ODBC**
Converte le chiamate ai metodi JDBC in chiamate a ODBC
- **Tipo 2: Driver parzialmente in Java più API native**
Converte le chiamate ai metodi JDBC in chiamate alle librerie del DBMS
- **Tipo 3: Driver Java puro più middleware**
Comunica con un server intermedio attraverso un protocollo proprietario
- **Tipo 4: Driver Java puro con protocollo proprietario**
Comunica con un protocollo proprietario direttamente con il DBMS

Maggini, Scarselli

Sistemi per basi di dati

18

Un esempio: JDBC II



19

CGI: un esempio

I CGI e form: un uso tipico

- il form raccoglie login e password dell'utente
- e chiama la procedura /cgi-bin/listaStudenti.cgi

```
<HTML>
<BODY>
<FORM ACTION="/cgi-bin/listaStudenti.cgi" METHOD="POST">
  NOME: <INPUT TYPE="text" NAME="nome" VALUE=""><P>
  COGNOME: <INPUT TYPE="text" NAME="cognome" VALUE="">
  <INPUT TYPE="submit" VALUE="Cerca">. <P>
</FORM>
</BODY>
</HTML>
```

il metodo
di invio dei parametri

i campi da leggere

il cgi da attivare

C:\franco\didattica\SistemiPerBasiDiDati...

File Modifica Visualizza Preferiti Strumenti ?

NOME:

COGNOME:

Sistemi per basi di dati

Maggini, Scarselli

20

CGI: un esempio II

Un CGI in Java

- si crea una classe di nome listaStudenti
- i parametri in ingresso
 - se il FORM usa il metodo POST, vengono letti dallo standard input
 - se il FORM usa il metodo GET, vengono letti da una variabile d'ambiente

```
import java.cgi.lib.*;

class listaStudenti {

    public static void main( String args[] ) {

        Hashtable form_data = cgi_lib.ReadParse(System.in);

        String nome = (String)form_data.get("nome");
        String cognome = (String)form_data.get("cognome");
    }
}
```

si leggono i parametri
dallo standard input

Maggini, Scarselli

Sistemi per basi di dati

21

CGI: un esempio III

Per connettersi ad un DBMS occorre

- Caricare il driver JDBC che si intende usare

```
String driver= "COM.ibm.db2.jdbc.net.DB2Driver";
Class.forName(driver).newInstance();
```

- Aprire una connessione con il DBMS

Driver, server
e database

```
String url= "jdbc:db2://segreteria.ing.unisi.it/stud";
String userid= "pippo";
String passwd= "pippoPass";
Connection con = DriverManager.getConnection(url, userid, passwd);
```

Maggini, Scarselli

Sistemi per basi di dati

22

CGI: un esempio IV

- Per eseguire un'interrogazione occorre creare uno **statement** ed eseguirlo

```
Statement stmt = con.createStatement();  
ResultSet rs = stmt.executeQuery("select cognome,nome,dataNascita"+  
    "from studenti"+  
    "where nome like '%" + nome+ "%' and "+  
    "cognome like '%" + cognome+"%'");
```

- **executeQuery** invia il comando al DBMS e pone il risultato in un oggetto di tipo **ResultSet**

Maggini, Scarselli

Sistemi per basi di dati

23

CGI: un esempio V

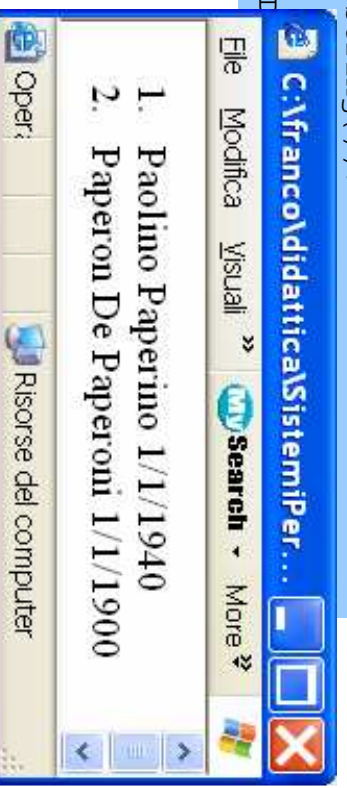
- L'oggetto **ResultSet** consente di scorrere le righe e accedere agli attributi
- I tipi di dati SQL sono trasformati in tipi di dati Java

```
System.out.println("<HTML><BODY><OL>");  
while (rs.next()) {  
    String cdgnome = rs.getString("cognome");  
    int nome = rs.getString("nome");  
    Date dn = rs.getDate("dataNascita");  
    System.out.println("<LI>" + cdgnome + " " + nome + " "  
        + dn.toString());  
}  
System.out.println("</OL></BODY>");
```

stampa inizio e
fine della pagina

stampa della
lista dei nomi

Iterazione su tutte le tuple
del risultato dell'interrogazione



Maggini, Scarselli

Alternative a JDBC

Per Java esistono alternative a JDBC

- **SQL**
 - una versione di **embedded SQL** specializzato per JAVA
 - il codice SQL è scritto direttamente all'interno di Java
 - un precompilatore trasforma l'SQL in chiamate al DBMS
- **JAVA Blend**
 - crea una connessione diretta fra **oggetti JAVA** e **oggetti del database**: si opera sulle tabelle come se fossero oggetti
- SQL e JAVA Blend si appoggiano su JDBC

Maggini, Scarselli

Sistemi per basi di dati

25

SQL: un esempio

I comandi SQL sono identificati da "#sql"

```
...
#sql iterator studentDetails(String nome, String cognome);
studentDetails studentIterator=null;
#sql studentIterator ={select cognome,nome from studenti};

System.out.println("<HTML><BODY><OL>");
while (studentIterator.next()) {
    System.out.println("<LI>"
        + studentIterator.nome() + " "
        + studentIterator.cognome() + " " );
}
System.out.println("</OL></BODY></HTML>");
rs.close();
```

Maggini, Scarselli

Sistemi per basi di dati

26



- SQL facilita l'analisi statica
 - della sintassi
 - dei tipi
 - dello schema
- SQL permette la generazione di strategie di accesso da parte del DBMS in modo trasparente
- SQL richiede una fase precompilazione



CGI è molto semplice da usare ed è uno standard diffuso, ma

- ad ogni accesso allo script (programma) CGI,
 - si crea un nuovo processo
 - si apre e richiede una connessione con il DBMS
 - tutte le comunicazioni fra client e CGI passano dal Web server
 - ci sono problemi di sicurezza
- Per ovviare a questi problemi

- i server Web forniscono nuove API: Netscape Server API, IIS API
 - gli script CGI sono caricati come parte del server Web e rimangono attivi fra una transazione e l'altra
 - gli script possono accedere a funzionalità per la connessione ai DBMS, per l'autenticazione e la gestione delle connessioni

Servlets

Cosa sono

- sono **applicazioni Java in esecuzione** su una JVM residente **sul server**
- realizzate attraverso il package Java: java.servlet

Come CGI

- il client invoca la servlet nel contesto di una FORM HTML
- la servlet esegue le operazioni richieste
- la servlet ridirige l'output al client in forma di pagina HTML

A differenza di CGI

- la servlet corrisponde ad un processo che viene caricato solo una volta e utilizzato per eseguire più operazioni distinte
- il server Web deve essere esteso con un servlet engine

Maggini, Scarselli

Sistemi per basi di dati

29

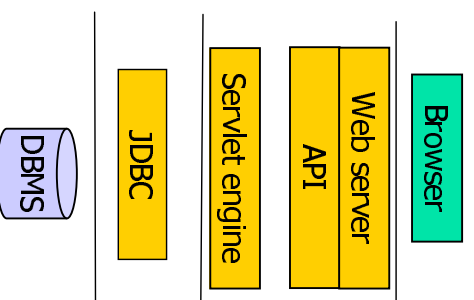
Servlets e DBMS

L'accesso a DBMS

- avviene attraverso JDBC, SQL, JavaBlend
- poiché' la servlet rimane attiva, le connessioni con il DBMS sono aperte una sola volta, solo al momento del caricamento

Osservazioni

- le servlet sono **indipendenti dalla piattaforma**
- permettono l'aggiunta di **ulteriori strati** all'architettura



Maggini, Scarselli

Sistemi per basi di dati

30

Servlets: un esempio

Una servlet deve contenere i metodi

- **init()**
 - chiamato dal server nel momento del caricamento della servlet
- **destroy()**
 - chiamato dal server per distruggere la servlet: es. dopo un periodo di inattività
- **doPost()** e **doGet()**
 - chiamati tutte le volte che un browser richiede l'esecuzione della servlet

```
public class ListaStudenti extends HttpServlet {  
    init() {...}  
    destroy() {...}  
    doGet() {...}  
    doPost() {...}  
}
```

Maggini, Scarselli

Sistemi per basi di dati

31

Servlets: un esempio II

- L'inizializzazione contiene la connessione al DBMS

```
String driver= "COM.ibm.db2.jdbc.net.DB2Driver";  
String url= "jdbc:db2://segreteria.ing.unisi.it/stud";  
:  
:  
:  
public void init()throws ServletException {  
    try{  
        class.forName(driver).newInstance();  
        Connection con = DriverManager.getConnection(url, userid, passwd);  
        PreparedStatement prep = con.prepareStatement(  
            "select cognome,nome,dataNascita"+  
            "from studenti"+  
            "where nome like %?% and "+  
            "cognome like %?%" );  
        catch (SQLException ex) {...}  
    }  
}
```

Apertura
connessione

creazione
prepared
statement

Maggini, Scarselli

Sistemi per basi di dati

32

Servlets: un esempio II

```
Public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter toClient = res.getWriter();
    String nome = req.getParameter("nome");
    String cognome = req.getParameter("cognome");

    toClient.println("<HTML><BODY><OL>");
    prep.setString(1,nome); prep.setString(2,cognome);
    resultSet rs=prep.executeQuery();

    while (rs.next()) {
        String c = rs.getString("cognome");
        String n = rs.getString("nome");
        Date d = rs.getDate("dataNascita");
        toClient.println("<LI>" + c + " " + n + " " + d.toString());
    }
    toClient.println("</OL></BODY></HTML>");
    toClient.close();
}
```

preparazione stream di uscita

Lettura parametri

esecuzione interrogazione

chiusura connessione con il browser

Maggini, Scarselli

Sistemi per basi di dati

33

Linguaggi di scripting lato server

Cosa sono

- linguaggi che permettono di generare **pagine dinamiche**
 - ASP, PHP, Active Perl,...
- Una pagina dinamica è costituita da
 - HTML**
 - codice** compreso tra tag speciali (<% e %>, <?php, ?> ..)

```
<HTML>
<TITLE>La pagina di Mago Mago' <TITLE>
<BODY>
Ecco l'ambo da giocare:
<?php
strand((double)microtime()*1000000);
$n1=rand(1,90);
$n2=rand(1,90);
while ($n1=$n2){
    $n2=rand(1,90);
}
echo "$n1 $n2 <P>";
?>
</BODY>
</HTML>
```

Maggini, Scarselli

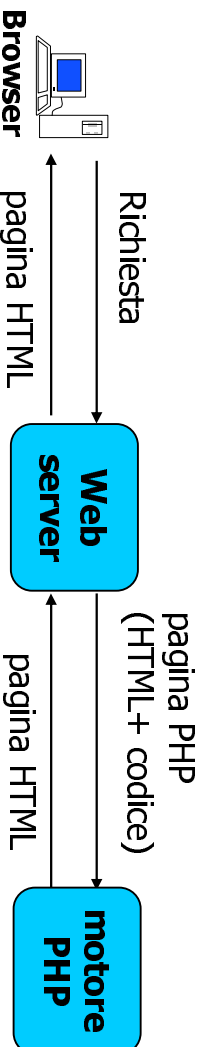
Sistemi per basi di dati

34

Linguaggi di scripting lato server

Come funzionano

- Il Web Server è **esteso** con un **motore di scripting** (Script Engine)
 - quando arriva una richiesta per una pagina consentente PHP, il Web server invia la pagina al motore di scripting
 - il motore di scripting **sostituisce al codice PHP** con TAG e testo HTML e produce una pagina HTML che invia al Web server
 - il Web server invia la pagina al browser



Maggini, Scarselli

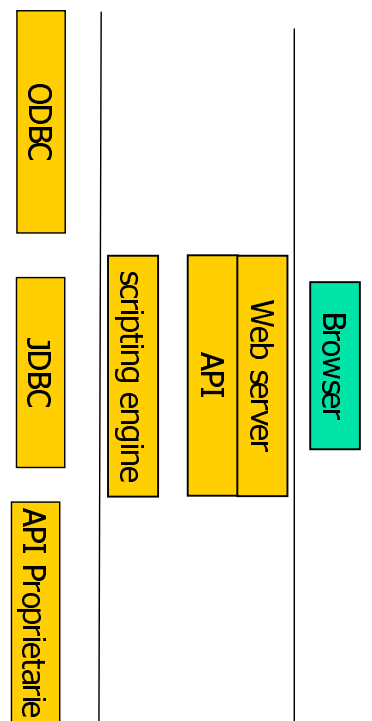
Sistemi per basi di dati

35

Connessione ai DBMS

Come ci si connette ai DBMS

- si usano connettori standard o API proprietarie
- il linguaggio di scripting deve fornire le funzionalità per la connessione



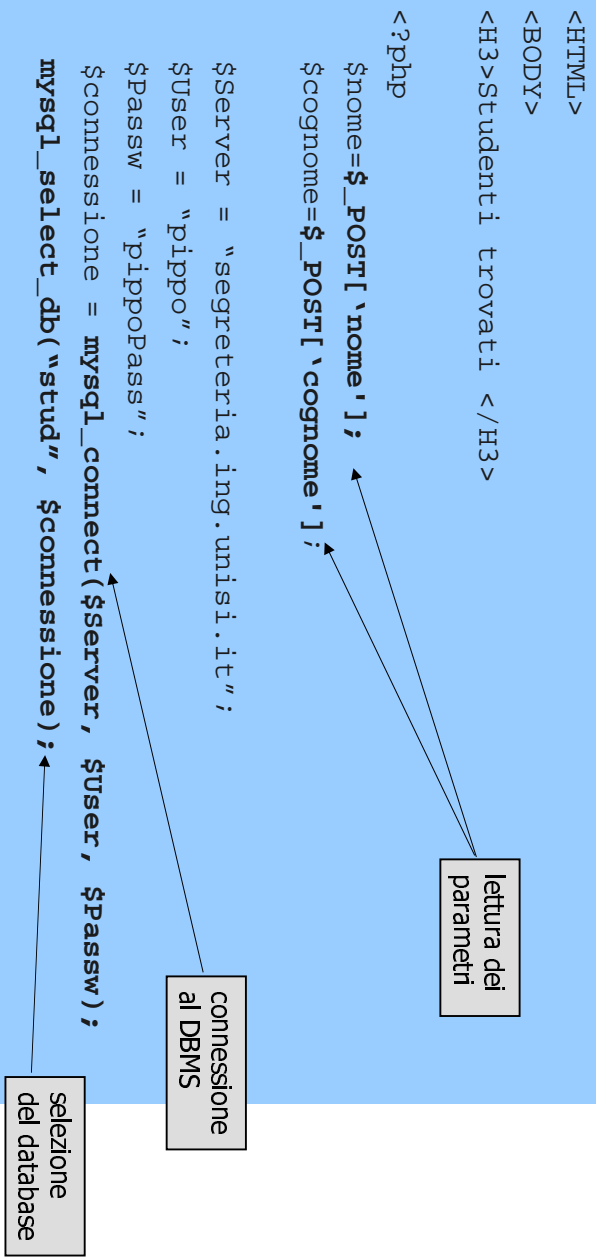
Maggini, Scarselli

Sistemi per basi di dati

36

PHP: un esempio

PHP e Form



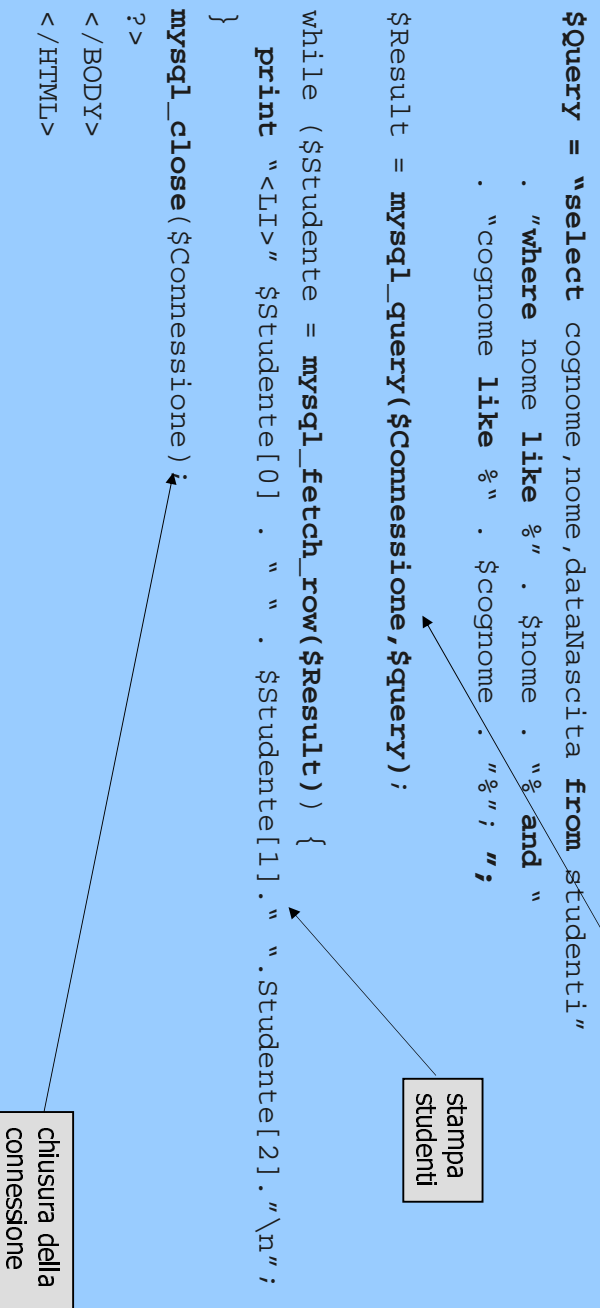
Maggini, Scarselli

Sistemi per basi di dati

37

PHP: un esempio II

Recupero e stampa degli studenti



Maggini, Scarselli

Sistemi per basi di dati

38

Java Server Pages (JSP)

Cos'è

- JSP è costruito sopra alla tecnologia servlet
- una pagina JSP contiene:
 - codice HTML
 - codice Java incluso in tag specifici

```
<% String nome = request.getParameter("nome");  
String cognome = request.getParameter("cognome");  
ResultSet rs = stmt.executeQuery("select cognome,nome,dataNascita"+  
    "from studenti"+  
    "where nome like '%" + nome + "%' and "+  
    "cognome like '%" + cognome+"%'");  
  
while (rs.next()) { %  
    String cognome = rs.getString("cognome");  
    int nome = rs.getInt("nome");%>  
    <LI> <%=cognome%> <%=nome%>  
%> }>
```

Maggini, Scarselli

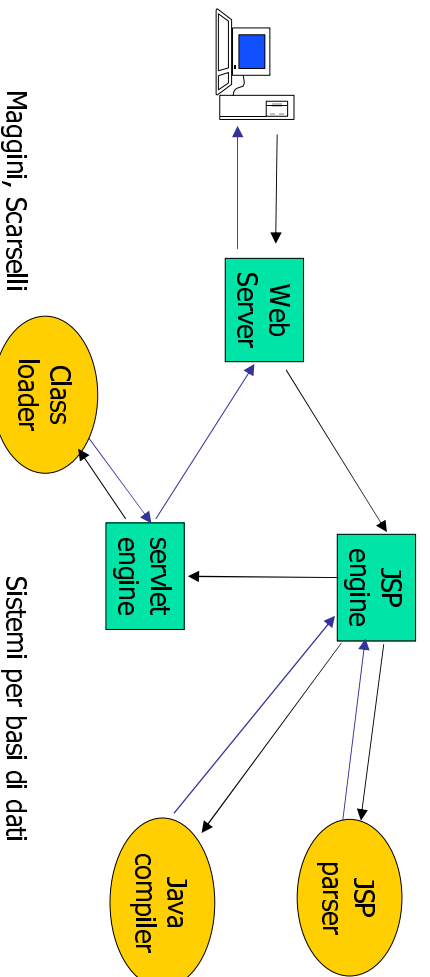
Sistemi per basi di dati

39

Java Server Pages (JSP) II

Come funziona

1. il JSP engine **analizza** la pagina e crea un file sorgente Java
2. il file viene compilato in un class file che **contiene** una servlet
3. il servlet engine **carica** la **servlet** per l'esecuzione
4. la servlet viene eseguita e restituisce i risultati



Maggini, Scarselli

40



CGI, Servlets e PHP

Vantaggi e svantaggi

- **Efficienza**
 - CGI è poco efficiente perche' per ogni richiesta si crea un nuovo processo
 - PHP è meno efficiente delle servlets perche' **completamente interpretato**, mentre le Servlets e JSP usano una codifica intermedia
- **Portabilita'**
 - CGI è uno standard, ma le applicazioni devono essere **ricomilate**
 - PHP e Servlets sono entrambe portabili, ma in PHP la connessione con DBMS puo' avvenire attraverso **API diverse**
- **Estensibilita'**
 - Java è un linguaggio che dispone di **numerosa API** per la connessione a DBMS (JDBC), l'interoperabilita' fra processi distribuiti (RMI e CORBA), ...

Maggini, Scarselli

Sistemi per basi di dati

41



CGI, Servlets e PHP

- **Gestione delle sessioni**
 - CGI usa i cookies per tenere traccia delle sessioni. Comunque, ad ogni richiesta occorre **riaprire la connessione** con il DBMS
 - Servlets e PHP possono mantenere aperta la connessione fra una richiesta e l'altra
- **Sicurezza**
 - Le servlets possono usare la **sicurezza** e la **tipizzazione esistenti in Java**
- **Semplicita'**
 - Il **PHP e JSP** forniscono un metodo molto rapido per scrivere pagine dinamiche

Maggini, Scarselli

Sistemi per basi di dati

42

Application servers

Cosà sono

- sono server **posizionati fra il Web server e DBMS**
- gestiscono l'integrazione di DBMS diversi
- forniscono servizi classici come la gestione delle transazioni distribuite
- permettono la creazione e il mantenimento di oggetti
- eseguono script
- implementano tecniche di clustering e bilanciamento del carico
- possono fornire le funzionalità per la gestione di mail, accesso tramite cellulare....

A cosa servono

- **implementano la business logic**
- **facilitano l'integrazione di risorse distribuite**
- **vantaggiosi in grandi aziende**

Maggini, Scarselli

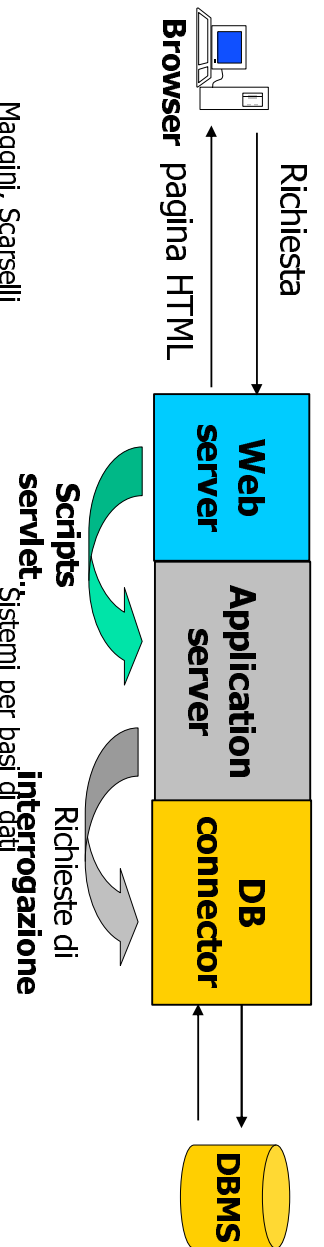
Sistemi per basi di dati

43

Application servers II

Come funzionano

- il Web server controlla il tipo di documento richiesto dal browser
 - se si tratta di una semplice pagina HTML, **provvede ad inviarla direttamente**
 - se si tratta di una pagina dinamica supportata dall'applicazion server, **invia la richiesta a quest'ultimo**
- L'applicazione server **elabora la pagina dinamica**
 - L'elaborazione può comportare l'interazione con oggetti distribuiti e DBMS



Maggini, Scarselli

44



Application servers: un esempio

Oracle Application Server (iAS) contiene

- Una JVM che supporta CORBA, Enterprise Java Beans (EJB...)
- un servlet engine
- un JSP engine
- un interprete perl
- un motore per PSP (il linguaggio proprietario PL/SQL Server Pages)
- un servizio di cache
- una estensione di Apache che permette anche connessioni sicure (mod_ssl) e invia ai vari motori le pagine contente script

Maggini, Scarselli

Sistemi per basi di dati

45



Java Applet

Sono applicazioni JAVA

- compilate e memorizzate in forma di bytecode sui server
- possono essere scaricate ed *eseguite dal browser*
- si connettono ai DBMS attraverso
 - **JDBC** (Java Database Connectivity)
 - fornisce un insieme di API con ci si puo' connettere ad un database SQL, inviare delle interrogazione, ricevere e analizzare le risposte
 - **SQLJ**
 - una versione di *embedded SQL* specializzato per JAVA
 - **JAVA Blend**
 - crea una connessione diretta fra *oggetti JAVA* e *oggetti del database*: si opera sulle tabelle come se fossero oggetti
- **SQLJ** e **JAVA Blend** si appoggiano su **JDBC**

Maggini, Scarselli

Sistemi per basi di dati

46

Esecuzione di un'applet: un esempio

La pagina HTML che permette di caricare l'applet

```
<HTML>
<TITLE> Pagina per lanciare l'applet</TITLE>
<BODY>
Vaì...
<APPLET CODE="archiviostudenti.class" WIDTH=300 Height=300>
</BODY>
</HTML>
```

Il codice dell'applet

```
public class archiviostudenti extends java.applet.Applet{
    public void init() {
        myTextArea.setText('In esecuzione ...')
        ...
    }
}
```

Chiamata dal browser per segnalare il caricamento dell'applet