PROGRAMMAZIONE SQL

Introduzione alle tecniche e linguaggio PHP

Programmazione per le basi di dati

- Vedremo quali sono le tecniche che sono state sviluppate per accedere alle basi di dati dai programmi

- Vedremo tre tecniche dedicate all'accesso da programmi applicativi
- Un esempio di accesso da applicazione web

(anche se il confine tra le due è "sottile")

Approcci alla programmazione per BDD

1. Incapsulamento di comandi di basi di dati in un linguaggio di programmazione general-purpose;
2. Utilizzo di una libreria di funzioni di base di dati;
3. Progettazione di un linguaggio completamente nuovo.

Conflitto di impedenza

- E' la locuzione usata per indicare i problemi che si verificano a causa delle differenze tra il modello della base di dati e il modello del linguaggio di programmazione.

Ad esempio:

- colonne (attributi) e loro tipi di dati
- righe (tuple o record)
- tabelle (relazioni)

sono i tre *costrutti principali* nel modello relazionale

Conflitto di impedenza: problema 1

- I tipi di dati del linguaggio di programmazione differiscono dai tipi di dati degli attributi nel modello dei dati della BDD.

- E' necessario definire un collegamento (*binding*) per ogni linguaggio di programmazione che specifichi per ogni tipo di attributo i tipi compatibili del linguaggio di programmazione.

Conflitto di impedenza: problema 2

- I risultati delle interrogazioni sono insiemi di tuple dove ogni tupla è formata da una sequenza di valori di attributi.

- E' necessario tradurre la struttura dei dati dei risultati delle interrogazioni, che è una tabella, in una struttura dati appropriata nel linguaggio di programmazione.
- E' necessario un meccanismo per *scorrere* le tuple del risultato di un'interrogazione

Sequenza tipica di accesso

In genere il sistema è implementato con un'architettura Client/Server: programma applicativo e server di BDD.

Una sequenza di interazione:

1. **Stabilire/aprire una connessione** al server della BDD. Viene fatto tramite un'URL di connessione + le credenziali di accesso.
2. **Operare sulla base di dati.** Interrogazioni, aggiornamenti, inserimenti e cancellazioni.
3. **Terminare/chiudere la connessione**.

Approccio 1: incapsulamento

- Le istruzioni di accesso e interazione con la BDD sono **incapsulate** nel linguaggio ospite di programmazione.
- Sono identificate da un prefisso speciale
- Ad esempio, in **embedded SQL:** EXEC SQL

- Un precompilatore esamina prima di tutto il codice sorgente del programma per identificare le istruzioni di interazione con la base di dati
- Le estrae per elaborarle con il sistema DBMS

Esempio: embedded SQL

- Dichiarazione delle variabili

```
0) int loop ;
1) EXEC SQL BEGIN DECLARE SECTION ;
2) varchar nome_d [16], nome_batt [16], cognome [16], indirizzo [31] ;
3) char ssn [10], data_n [11], sesso [2], iniz_int [2] ;
4) float stipendio, aumento ;
5) int n_d, numero_d ;
6) int SQLCODE ; char SQLSTATE [6] ;
7) EXEC SQL END DECLARE SECTION ;
```


Esempio: embedded SQL

- Ricerca di un impiegato in base al SSN

```
   //Segmento di programma E1:
 0) loop = 1 ;
 1) while (loop) {
 2)   prompt("Immettere un numero di previdenza sociale: ", ssn) ;
 3)   EXEC SQL
 4)     select NOME_BATT, INIZ_INT, COGNOME, INDIRIZZO, STIPENDIO
 5)     into :nome_batt, :iniz_int, :cognome, :indirizzo, :stipendio
 6)     from IMPIEGATO where SSN = :ssn ;
 7)   if (SQLCODE == 0) printf(nome_batt, iniz_int, cognome,
                              indirizzo, stipendio)
 8)     else printf ("Non esiste numero di previdenza sociale: ",
                     ssn) ;
 9)   prompt ("nuovo numero di previdenza sociale (immettere 1
              per Sì, 0 per No): ", loop) ;
10)   }
```

Esempio: embedded SQL

- Reperimento di record mediante cursore

```
    //Segmento di programma E2:
 0) prompt ("Immettere il nome di dipartimento: ",nome_d) ;
 1) EXEC SQL
 2)    select NUMERO_D into :numero_d
 3)    from DIPARTIMENTO where NOME_D = :nome_d ;
 4) EXEC SQL DECLARE EMP CURSOR FOR
 5)    select SSN, NOME_BATT, INIZ_INT, COGNOME, STIPENDIO
 6)    from IMPIEGATO where N_D = :numero_d
 7)    FOR UPDATE OF STIPENDIO ;
 8) EXEC SQL OPEN EMP ;
 9) EXEC SQL FETCH from EMP into :ssn, :nome_batt, :iniz_int,
       :cognome, :stipendio ;
10) while (SQLCODE == 0) {
11)    printf ("Il nome dell'impiegato è:", nome_batt,
       iniz_int, cognome)
12)    prompt ("Immettere l'importo dell'aumento : ", aumento) ;
13)    EXEC SQL
14)       update IMPIEGATO
15)       set STIPENDIO = STIPENDIO + :aumento
16)       where CURRENT OF EMP ;
17)    EXEC SQL FETCH from EMP into :ssn, :nome_batt, :iniz_int,
          :cognome, :stipendio ;
18)    }
19) EXEC SQL CLOSE EMP ;
```


Esempio: SQLJ

- Ricerca di un impiegato in base al SSN

```
    //Segmento di programma J1:
 1) ssn = readEntry("Immettere un numero di previdenza sociale: ") ;
 2) try {
 3)   #sql{select NOME_BATT, INIZ_INT, COGNOME, INDIRIZZO, STIPENDIO
 4)      into :nome_batt, :iniz_int, :cognome, :indirizzo, :stipendio
 5)      from IMPIEGATO where SSN = :ssn} ;
 6) }  catch (SQLException se) {
 7)      System.out.println("Non esiste numero di previdenza sociale:
         " + ssn) ;
 8)      Return ;
 9)    }
10) System.out.println(nome_batt + " " + iniz_int + " " + cognome +
      " " + indirizzo + " " + stipendio)
```

Approccio 2: utilizzo di librerie

- Per le *chiamate* alla base di dati, è resa disponibile una **libreria di funzioni** per il linguaggio ospite di programmazione
- Ad esempio, possono esserci funzioni per collegarsi alla BDD o per eseguire un'interrogazione
- Le librerie sono chiamate API (Application Program Interface) per accedere alla BDD

Esempio: JDBC

The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC API provides a call-level API for SQL-based database access.

- Getting started with JDBC API http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html
- JDBC Overview http://www.oracle.com/technetwork/java/overview-141217.html
- JDBC Tutorialhttp://docs.oracle.com/javase/tutorial/jdbc/

Esempio: JDBC

```
    //Segmento di programma JDBC1:
 0) import java.io.* ;
 1) import java.sql.*
    …
 2) class getEmpInfo {
 3)   public static void main (String args []) throws SQLException,
       IOException {
 4)    try { Class.forName("oracle.jdbc.driver.OracleDriver")
 5)    } catch (ClassNotFoundException x) {
 6)     System.out.println ("Il driver non potrebbe essere caricato") ;
 7)     }
 8)    String dbacct, password, ssn, cognome ;
 9)       Double stipendio ;
10)    dbacct = readentry("Immettere account di base di dati:") ;
11)    password = readentry("Immettere password:") ;
12)    Connection conn = DriverManager.getConnection
13)     ("jdbc:oracle:oci8:" + dbacct * "/" + passwrd) ;
14)    String stmt1 = "selezionare COGNOME, STIPENDIO from
         IMPIEGATO where SSN = ?" ;
15)    PreparedStatement p = conn.prepareStatement(stmt1) ;
16)    ssn = readentry ("Immettere un numero di previdenza
         sociale: ") ;
17)    p.clearParameters() ;
18)    p.setString(1, ssn) ;
19)    ResultSet r = p.executeQuery() ;
20)    while (r.next()){
21)      cognome = r.getString(1) ;
22)      stipendio = r.getDouble(2) ;
23)      system.out.printline(cognome + stipendio) ;
24)   } }
25) }
```


Esempio: JDBC

```
    //Segmento di programma JDBC2:
 0) import java.io.* ;
 1) import java.sql.*
    …
 2) class printDepartmentEmps {
 3)   public static void main (String args [])
       throws SQLException, IOException {
 4)   try { Class.forName("oracle.jdbc.driver.OracleDriver")
 5)   } catch (classNotFoundException x) {
 6)    System.out.println ("Il driver non potrebbe essere caricato") ;
 7)    }
 8)   String dbacct, password, cognome ;
 9)   Double stipendio ;
10)   Integer n_d ;
11)   dbacct = readentry ("Immettere account di base di dati:") ;
12)   passwrd = readentry ("Immettere password:") ;
13)   Connection conn = DriverManager.getConnection
14)    ("jdbc:oracle:oci8:" + dbacct + "/" + password) ;
15)   dno = readentry("Immettere un numero di dipartimento: ") ;
16)   String q = "selezionare COGNOME, STIPENDIO from IMPIEGATO
       where N_D = " + dno.tostring() ;
17)   Statement s = conn.createStatement() ;
18)   ResultSet r = s.executeQuery(q) ;
19)   while (r.next()) {
20)    cognome = r.getString(1) ;
21)    stipendio = r.getDouble(2) ;
22)    system.out.printline(cognome + stipendio) ;
23)  }  }
24) }
```

Approccio 3: nuovo linguaggio

- Viene progettato da zero un **nuovo linguaggio di programmazione** per basi di dati per essere compatibile con il modello della BDD e con il linguaggio di interrogazione.

- Le varie strutture di programmazione (cicli e istruzioni condizionali) vengono aggiunte al linguaggio per convertirlo in un linguaggio di programmazione completo

- Un esempio è PL/SQL di Oracle.

Esempio: PL/SQL

- Dichiarazione di una funzione

```
      // Funzione PSM1:
0)    CREATE FUNCTION DeptsSize(IN deptno INTEGER)
1)    RETURNS VARCHAR [7]
2)    DECLARE NoOfEmps INTEGER ;
3)    SELECT COUNT(*) INTO NoOfEmps
4)    FROM IMPIEGATO WHERE N_D = n_dip ;
5)    IF NoOfEmps > 100 THEN RETURN "ENORME"
6)       ELSEIF NoOfEmps > 25 THEN RETURN "GRANDE"
7)       ELSEIF NoOfEMPS > 10 THEN RETURN "MEDIO"
8)       ELSE RETURN "PICCOLO"
9)    END IF ;
```

Accesso alle basi di dati tramite PHP

- Vediamo come realizzare semplici interfacce web per accedere/interagire con le basi di dati.
- Useremo PHP come linguaggio di scripting per realizzare pagine web *dinamiche* (HTML)
- Useremo MySQL come DBMS

HTML – un esempio

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

- La dichiarazione DOCTYPE definisce il tipo di documento.
- Il testo tra <html> e </html> descrive la pagina web
- Il testo tra <body> e </body> è il contenuto visibile della pagina
- Il testo tra <h1> e </h1> è visualizzato come un'intestazione
- Il testo tra e è visualizzato come un paragrafo
- La dichiarazione <!DOCTYPE html> è il doctype per l'HTML5.

Cos'è l'HTML?

L'HTML è un linguaggio per descrivere pagine web.

- HTML è l'acronimo di **H**yper **T**ext **M**arkup **L**anguage
- HTML è un linguaggio basato su un insieme di **marcatori (tag)**
- I tag **descrivono** il contenuto del documento
- I documenti HTML **contengono tag e testo *semplice***
- I documenti HTML sono chiamati anche **pagine web**

Tag HTML

- I marcatori HTML sono spesso chiamati tag HTML
- I tag sono keyword circondate da **parentesi angolari** come, ad esempio, <html>
- I tag HTML **sono abitualmente in coppia** come, ad esempio, e
- Il primo tag è il **tag di apertura**, il secondo quello di **chiusura** (ed è come quello di apertura + una barra)

<tagname>content</tagname>

Elementi HTML

- “Tag HTML” e “elementi HTML” sono spesso usati per descrivere la stessa cosa
- In realtà un elemento HTML è tutto quello tra il tag di apertura e quello di chiusura (tag compresi):

- Un esempio di un elemento HTML:

** This is a paragraph. **

Struttura di una pagina HTML

```
<html>
  <head>
  </head>
  <body>
    <h1>This a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Intestazioni HTML

Le intestazioni HTML sono definite con i tag da <h1> a <h6>

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```


Paragrafi HTML

I paragrafi HTML sono definiti con il tag <p>.

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

`<br />` per andare a capo nella pagina HTML.

Link HTML

I link HTML sono definiti tramite il tag <a>.

```
<a href="http://www.w3schools.com">This is a
link</a>
```

- L'indirizzo del link è definito nell'attributo href

Immagini HTML

Le immagini HTML sono definite tramite il tag <img>

```
<img src="w3schools.jpg" width="104"
height="142">
```

- Il nome del file e la dimensione dell'immagine è fornita come attributo del tag <img>

Tabelle HTML

Tag	Description
<table>	Defines a table
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<caption>	Defines a table caption
<colgroup>	Specifies a group of one or more columns in a table for formatting
<col>	Specifies column properties for each column within a <colgroup> element
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

Tabelle HTML

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Form HTML (1)

- I form HTML sono usati per inviare dati al server.
- Un form HTML può contenere elementi di INPUT come campi di testo, checkbox, radio-button, bottoni di submit e altro.
- Per creare un form HTML si usa il tag <form>:

```
<form>
...
input elements
...
</form>
```


Form HTML (2)

- L'elemento più importante è l'elemento <input>. E' utilizzato per selezionare informazioni utente.
- L'elemento può variare in tanti modi, in base al valore dell'attributo *type*.

Esempio: Text Fields. <input type="text"> defines a one-line input field that a user can enter text into:

```
<form>
First name: <input type="text"
name="firstname"><br>
Last name: <input type="text" name="lastname">
</form>
```

Form HTML (3)

- <input type="submit"> definisce un bottone di submit (invio).
- E' utilizzato per inviare i dati del form al server.
- I dati sono inviati alla pagina specificata nell'attributo *action*
- In genere la pagina specificata in *action* fa qualcosa con i dati ricevuti.

```
<form name="input" action="html_form_action.asp"
method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```


Per saperne di più

- http://www.w3schools.com/html/default.asp

Il linguaggio PHP

- PHP è un linguaggio di scripting server-side.
- E' un potente strumento per realizzare pagine web dinamiche e interattive.
- PHP è molto diffuso, è free, è un'alternativa efficiente ai suoi concorrenti (come ASP di Microsoft).

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
//This is a PHP comment line
?>

</body>
</html>
```


Cos'è il PHP?

- PHP è l'acronimo di **P**HP: **H**ypertext **P**reprocessor
- Il PHP è un linguaggio di scripting open source
- Gli script PHP sono eseguiti sul server
- Il PHP è libero (e gratis) da scaricare e usare

- E' semplice per chi inizia
- Offre caratteristiche avanzate per i programmatori professionisti

Cos'è un file PHP?

- I file PHP possono contenere testo semplice, codice HTML, codice JavaScript e codice PHP.
- Il codice PHP è eseguito sul server e il risultato è restituito al browser come HTML semplice.
- I file PHP hanno un'estensione di deafult: ".php"

Cosa può fare PHP?

- Con il PHP è possibile generare pagine dal contenuto dinamico
- Con il PHP è possibile create, aprire, leggere, scrivere e chiudere file sul server
- Con il PHP è possibile raccogliere dati da un FORM
- Con il PHP è possibile inviare e ricevere cookie
- Con il PHP è possibile aggiungere, cancellare e modificare dati in una base di dati

PHP e HTML

- Il codice PHP deve essere compreso fra appositi tag di apertura e di chiusura, che sono i seguenti:

```
<?php //tag di apertura
?>    //tag di chiusura
```

- Tutto ciò che è contenuto fra questi tag deve corrispondere alle regole sintattiche del PHP, ed è codice che sarà eseguito dall'interprete e non sarà inviato direttamente al browser al browser.
- Per generare l'output da inviare al browser attraverso codice PHP viene normalmente utilizzato il costrutto `echo`.

Variabili in PHP

- Le variabili PHP sono utilizzate per memorizzare valori (x=5) o espressioni (z=x+y).
- Regole per le variabili in PHP:
  - Una variabile inizia con il simbolo $ seguito dal nome della variabile
  - Il nome di una variabile deve iniziare con una lettera o con underscore
  - Il nome di una variabile può contentere solo caratteri alfanumerici e underscore
  - Il nome non può contenere spazi
  - I nomi delle variabili sono case sensitive ($y e $Y sono due differenti variabili)

- In PHP non ci sono comandi per dichiarare variabili.
- Una variabile è creata nel momento in cui, per la prima volta, gli viene assegnato un valore.

```
$txt="Hello world!";
$x=5;
```

- Quando si assegna un valore testuale ad una variabie, usare gli apici (o doppi apici).

- Negli esempi sopra non abbiamo specificato il TIPO della variabile.
- PHP converte automaticamente la variabile al tipo di dato corretto, in base al suo valore.

Stringhe in PHP

- C'è un solo operatore per le stirnghe in PHP.
- L'operatore di concatenazione (.) è usato per unire due stringhe tra di loro.

```
<?php
$txt1="Hello world!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```


Operatori in PHP

- L'operatore di assegnamento = è usato per assegnare valori alle variabili.

Operator	Name	Description	Example	Result
x + y	Addition	Sum of x and y	2 + 2	4
x - y	Subtraction	Difference of x and y	5 - 2	3
x * y	Multiplication	Product of x and y	5 * 2	10
x / y	Division	Quotient of x and y	15 / 5	3
x % y	Modulus	Remainder of x divided by y	5 % 2 10 % 8 10 % 2	1 2 0
- x	Negation	Opposite of x	- 2	
a . b	Concatenation	Concatenate two strings	"Hi" . "Ha"	HiHa

Operatori in PHP

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus
a .= b	a = a . b	Concatenate two strings

Operator	Name	Description
++ x	Pre-increment	Increments x by one, then returns x
x ++	Post-increment	Returns x, then increments x by one
-- x	Pre-decrement	Decrements x by one, then returns x
x --	Post-decrement	Returns x, then decrements x by one

Operatori in PHP

Operator	Name	Description	Example
x == y	Equal	True if x is equal to y	5==8 returns false
x === y	Identical	True if x is equal to y, and they are of same type	5==="5" returns false
x != y	Not equal	True if x is not equal to y	5!=8 returns true
x <> y	Not equal	True if x is not equal to y	5<>8 returns true
x !== y	Not identical	True if x is not equal to y, or they are not of same type	5!=="5" returns true
x > y	Greater than	True if x is greater than y	5>8 returns false
x < y	Less than	True if x is less than y	5<8 returns true
x >= y	Greater than or equal to	True if x is greater than or equal to y	5>=8 returns false
x <= y	Less than or equal to	True if x is less than or equal to y	5<=8 returns true

Operatori in PHP

Operator	Name	Description	Example
x and y	And	True if both x and y are true	x=6 y=3 (x < 10 and y > 1) returns true
x or y	Or	True if either or both x and y are true	x=6 y=3 (x==6 or y==5) returns true
x xor y	Xor	True if either x or y is true, but not both	x=6 y=3 (x==6 xor y==3) returns false
x && y	And	True if both x and y are true	x=6 y=3 (x < 10 && y > 1) returns true
x || y	Or	True if either or both x and y are true	x=6 y=3 (x==5 || y==5) returns false
! x	Not	True if x is not true	x=6 y=3 !(x==y) returns true

Esempio pratico di costrutto if...esle if...esle...

```
<?php
$t=date("H");
if (($t<"10")
  {
  echo "Have a good morning!";
  }
else if (($t<"20")
  {
  echo "Have a good day!";
  }
else
  {
  echo "Have a good night!";
  }
?>
```

Istruzioni di iterazione.

In PHP ci sono i seguenti costrutti per l'iterazione:

- **while** – *cicla* su un blocco di codice mentre una specifica condizione è VERA
- **do...while** – *cicla* su un blocco di codice una volta e ripete il ciclo finché una specifica condizione è VERA
- **for** – *cicla* su un blocco di codice uno specifico numero di volte
- **foreach** – *cicla* su un blocco di codice per ogni elemento in un array

Istruzioni di iterazione. Esempio.

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

Array (1).

In PHP si usa la funzione array() per creare un array.
In PHP ci sono tre tipi di array:

- **Array (indicizzati)** – Array con un indice numerico
- **Array associativo** – Array con chiavi nominali
- **Array multidimensionali** – Array contenenti uno o più array

Array (indicizzati). Ci sono due modi per crearli:

- L'indice è assegnato automaticamento (il primo è sempre 0)
  $cars=array("Volvo","BMW","Toyota");
- L'indice è assegnato manualmente:
  $cars[0]="Volvo";
  $cars[1]="BMW";
  $cars[2]="Toyota";

Esempio:

```
<?php  $cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " .
$cars[2] . ".";
?>
```


Array (2).

La funzione count() è isata per restituitre la lunghezza (il numero degli elmenti) di un array:

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
```

Scorrere un array:

```
<?php
$cars=array("Volvo","BMW","Toyota");
$arrlength=count($cars);

for($x=0;$x<$arrlength;$x++)
  {
  echo $cars[$x];
  echo "<br />";
  }
?>
```

Array (3).

Array associativi. Due modi per crearli:

```
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
```

oppure

```
$age['Peter']="35";
$age['Ben']="37";
$age['Joe']="43";
```

Le chiavi possono essere usate in uno script. Esempio:

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```


Array (4).

Scorrere un array associativo.

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>
"43");

foreach($age as $x=>$x_value)
  {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br />";
  }
?>
```

Funzioni. Esempio.

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br>";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```


Gestione dei FORM (1).

Ogni elemento di un FORM in una pagina HTML è **automaticamente** disponibile negli script PHP.

Esempio, la pagina esempio.php contiene:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

Gestione dei FORM (2).

- Quando un utente compila il form e clicca sul botton di submit, i dati del form sono inviati a un file PHP chiamato "welcome.php":
- Se "welcome.php" contiene:

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br>
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

- L'output sarà qualcosa di simile a :

```
Welcome John!
You are 28 years old.
```


Variabili $_GET e $_POST

- La variabile predefinita $_GET è usata per memorizzare i valori di un form inviato con *method="get"*
- Le informazioni inviate con un form con il metodo GET sono visibili a tutti (sono visualizzate nella barra degli indirizzi del browser) e hanno un limite di dimensioni.

- La variabile predefinita $_POST è usata per memorizzare i valori di un form inviato con *method="get"*
- Le informazioni inviate con un form con il metodo POST non sono visibili e "non" hanno un limite di dimensioni.

Accesso al DB.

- In PHP 5 esistono diverse soluzioni per connettersi ai database.
- In PHP 5 è possibile accedere a MySQL attraverso **i layer di astrazione** distribuiti nella release standard (PDO ed SDO), ma anche utilizzando **la libreria mysql** (quella utilizzata anche nella versione precedente di PHP) e **la nuova libreria mysqli** che fornisce un supporto più completo al linguaggio ed espone anche un'interfaccia ad oggetti.

Aprire una connessione.

```
mysqli_connect(host,username,password,dbname);
```

Parameter	Description
host	Optional. Either a host name or an IP address
username	Optional. The MySQL user name
password	Optional. The password to log in with
dbname	Optional. The default database to be used when performing queries

```
<?php
// Create connection
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
?>
```

Chiudere una connessione.

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");

// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

mysqli_close($con);
?>
```


SELECT FROM

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

$result = mysqli_query($con,"SELECT * FROM Persons");

while($row = mysqli_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }

mysqli_close($con);
?>
```

SELECT FROM (in una tabella HTML)

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno()){
  echo "Failed to connect to MySQL: " . mysqli_connect_error(); }

$result = mysqli_query($con,"SELECT * FROM Persons");

echo "<table border='1'> <tr> <th>Firstname</th> <th>Lastname</th> </tr>";

while($row = mysqli_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";

mysqli_close($con);
?>
```


SELECT FROM WHERE

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

$result = mysqli_query($con,"SELECT * FROM Persons
WHERE FirstName='Peter'");

while($row = mysqli_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br>";
  }
?>
```

INSERT da un form (1)

- Il form:

```
<html>
<body>

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname">
Lastname: <input type="text" name="lastname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```


INSERT da un form (2)

- insert.php:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error(); }

$sql="INSERT INTO Persons (FirstName, LastName, Age)VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";

if (!mysqli_query($con,$sql)) {
  die('Error: ' . mysqli_error($con));}
echo "1 record added";

mysqli_close($con);
?>
```

Per saperne di più

- http://www.php.net
- http://www.w3schools.com/php/default.asp

Esercizio.

- Visualizzare la tabella impiegati del database AZIENDA su una pagina web (usando PHP e mysqli per collegarsi al DB)