

# Utilizzare PHP 5

Corso Interazione Uomo – Macchina  
AA 2005/2006

## Le variabili GET e POST

- La principale particolarità del 'web dinamico' è la possibilità di variare i contenuti delle pagine in base alle richieste degli utenti.
- Questa possibilità si materializza attraverso i meccanismi che permettono agli utenti, oltre che di richiedere una pagina ad un web server, anche di specificare determinati parametri che saranno utilizzati dallo script PHP per determinare quali contenuti la pagina dovrà mostrare
- Esistono due sistemi per passare dati ad uno script:
  - Metodo GET
  - Metodo POST

## Il metodo GET

- Il **metodo GET** consiste nell'accodare i dati all'indirizzo della pagina richiesta, facendo seguire il nome della pagina da un punto interrogativo e dalle coppie nome/valore dei dati che ci interessano. Nome e valore sono separati da un segno di uguale. Le diverse coppie nome/valore sono separate dal segno '&'

```
<a href="prodotto.php?cod=a7&cat=2">
```

- La stringa che si trova dopo il punto interrogativo, contenente nomi e valori dei parametri, viene detta **query string**

## Accesso ai dati GET

- Quando una pagina è utilizzata con il metodo GET, essa avrà a disposizione, al suo interno un array denominato `$_GET` che è un array **superglobale** in quanto è disponibile anche all'interno delle funzioni
- Nell'esempio precedente avremo a disposizione
  - `$_GET['cod']` (con valore 'a7')
  - `$_GET['cat']` (con valore '2')

## Il metodo POST

- Il **metodo POST** è utilizzato con i moduli: quando una pagina HTML contiene un tag `<form>`, uno dei suoi attributi è "method", che può valere GET o POST. Se il metodo è GET, i dati vengono passati nella query string, come abbiamo visto prima. Se il metodo è POST, i dati vengono invece inviati al server mediante lo standard input di questo

```
<form action="elabora.php" method="post">  
  <input type="text" name="nome">  
  <input type="checkbox" name="nuovo" value="si">  
  <input type="submit" name="submit" value="invia">  
</form>
```

## Accesso ai dati POST

- Quando una pagina è utilizzata con il metodo POST i dati che vengono passati sono memorizzati nell'array `$_POST`. Anche questo array, come `$_GET`, è un array superglobale
- Nell'esempio precedente avremo a disposizione:
  - `$_POST['nome']`
  - `$_POST['nuovo']`

## Cookies

- Come ogni buon linguaggio legato alla generazione dinamica di pagine web, anche il PHP consente di gestire i Cookies: piccoli file di testo contenenti informazioni utili e non dannose per la gestione delle sessioni sul web
- Le funzioni relative ai Cookies sono interne al PHP stesso, differentemente da come avviene per altri linguaggi, Perl su tutti

## Cookies: setcookie()

```
setcookie(Nome, Valore, Espirazione, Percorso, Dominio, Secure);
```

Vediamo di chiarire le opzioni che si possono passare alla funzione:

- *Nome*: è il nome del cookie, che può essere arbitrariamente scelto
- *Valore*: è il valore, anch'esso arbitrario, da assegnare al cookie
- *Espirazione*: è la data di espirazione del cookie
- *Percorso*: è la directory, a partire dal dominio (vedi sotto) per la quale il cookie è valido
- *Dominio*: è il dominio per il quale il dominio è valido
- *Secure*: è un valore che imposta se il cookie debba essere inviato tramite una connessione HTTPS

## Cookies: leggere

- La lettura dei Cookies avviene mediante la variabile pre-assegnata:
  - \$HTTP\_COOKIE\_VARS (tipo array)
- Mediante la variabile è possibile accedere a tutti i Cookies creati nel nostro browser
- Per accedere ad un particolare Cookie è necessario passare come indice dell'array il nome del Cookie creato

## Cookies: leggere

- Creare un cookie:

```
setcookie("Test", "Prova per il cookie Test", time()+60);
```

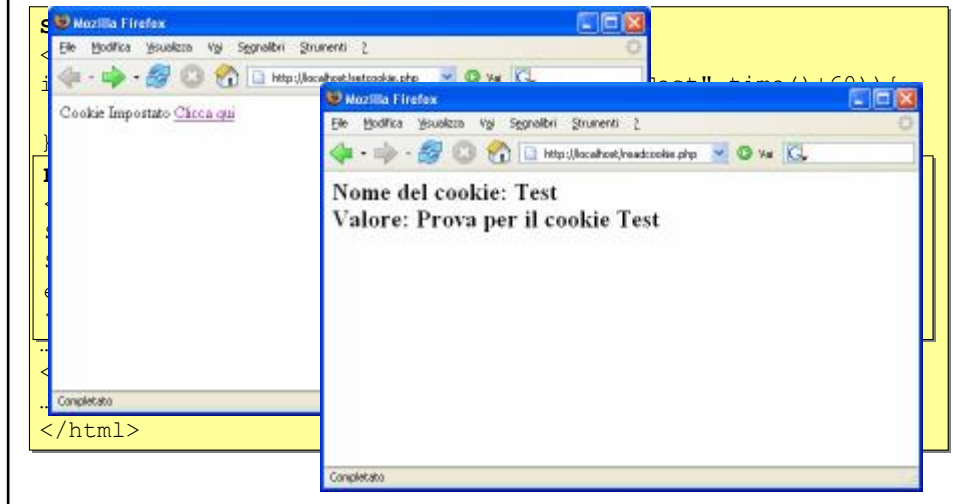
- Leggere un cookie:

```
$nome = "Test";  
$valore = $HTTP_COOKIE_VARS[$nome];  
echo "<h2>Nome del cookie: $nome <br> Valore: $valore</h2>";
```

- Leggere tutti i cookies:

```
while (list($nome,$valore)=each($HTTP_COOKIE_VARS)) {  
    echo "<h2>Nome del cookie: $nome - Valore: $valore</h2>";  
}
```

## Cookies: esempio



## Le sessioni

- La gestione delle sessioni è una delle novità più importanti introdotte dalla versione 4 di PHP. Essa infatti ci permette di stabilire un "dialogo" con l'utente del sito web, superando uno dei limiti del protocollo HTTP, che è quello di "non avere stato"
- La normale navigazione può avvenire senza il loro utilizzo
- Ma ad esempio in un sistema di e-commerce, nel quale l'utente prima riempie un carrello scegliendo una serie di prodotti, poi viene portato sulla schermata che gli consente di effettuare il pagamento. Quando è il momento di effettuare il pagamento il sistema deve "sapere" quali sono i prodotti che l'utente ha scelto
- Un altro esempio, più semplice, è quello dei siti con aree riservate: se un certo numero di pagine è visibile solo agli utenti registrati, questi dovranno presentarsi per poterle vedere, ma una volta effettuata la presentazione (login) dovranno essere messi in grado di vedere **tutte** le pagine dell'area riservata, senza, ovviamente, doversi ripresentare per ciascuna pagina

## Le sessioni: funzionamento

- Con le sessioni è possibile associare uno stato ad ogni client e mantenerlo nell'arco di più accessi
- Il mantenimento di una sessione richiede la collaborazione di client (il browser) e server. Tutte le informazioni della sessione sono sul server
- Il server codifica (serializzazione) le informazioni di sessione per la memorizzazione su file o database, e le ripristina alla successiva richiesta dello stesso client
- Il client deve fornire al server le informazioni per associare alla richiesta la sessione corrispondente
- Ciò è reso possibile da un identificativo di sessione (SID, session identifier) univoco generato all'avvio di una nuova sessione

## Le sessioni: implementazione

- La problematica principale riguardo le sessioni, è la propagazione del SID. Le tecniche di propagazione sono tramite cookies e tramite URL
- Grazie ad un cookie l'identificativo di sessione viene memorizzato sul browser, diventando accessibile in PHP
- Non è sufficientemente affidabile; il visitatore, potrebbe aver configurato il proprio browser con i cookie disabilitati
- La propagazione tramite URL è del tutto identica al passaggio di parametri con il metodo GET
- Si tratta di modificare i link alle varie pagine PHP in modo che contengano, nella parte di query string, il valore della costante SID

```
...  
<a href="successiva.php?php print SID?&gt;"&gt;Clicca qui&lt;/a&gt;<br/per propagare la sessione...  
...
```

## Le sessioni: funzioni PHP

- Le funzioni PHP per la gestione delle sessioni sono:
  - `session_start()`, crea una nuova sessione o la ripristina se esistente
  - `session_register()`, registra variabili come variabili di sessione
  - `session_destroy()`, distrugge una sessione, liberando le risorse
- L'interprete PHP verifica, in base ai cookie e ai parametri nella URL, se è stato fornito un identificativo (SID) di una sessione
- In caso affermativo provvede a ripristinare la sessione corrispondente diversamente ne avvia una nuova
- Affinché `session_start()` possa impostare il cookie contenente il SID, dovrà essere invocata prima che l'output sia inviato al browser
- A questo punto qualsiasi variabile può essere resa variabile di sessione, tramite la funzione `session_register()`

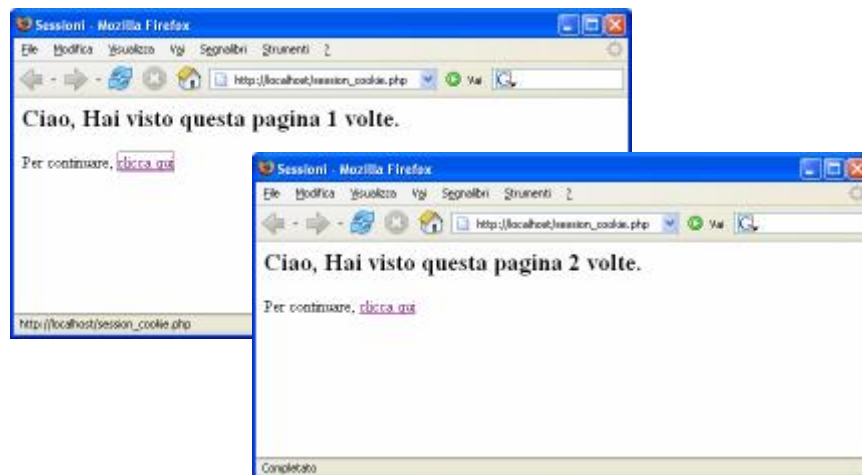
## Le sessioni: un esempio (session\_cookie.php)

```
<?php
session_start();
if (!session_is_registered('contatore')) {
    session_register('contatore');
    $_SESSION[contatore] = 1;
}
else {
    $_SESSION[contatore]++;
}
?>
<html>
  <head>
    <title>Sessioni</title>
  </head>
  <body>
    <?php
        echo "<h2>Ciao, Hai visto questa
            pagina $_SESSION[contatore] volte.</h2>";
        echo "Per continuare,
            <A HREF=\"session_cookie.php\">clicca qui</A>";
    ?>
  </body>
</html>
```

## Le sessioni: un esempio (session\_url.php)

```
<?php
session_start();
if (!session_is_registered('contatore')) {
    session_register('contatore');
    $_SESSION[contatore] = 1;
}
else {
    $_SESSION[contatore]++;
}
?>
<html>
<head>
<title>Sessioni</title>
</head>
<body>
<?php
    echo "<h2>Ciao, Hai visto questa
        pagina $_SESSION[contatore] volte.</h2>";
    echo "Per continuare,
        <A HREF=\"session_url.php?\".SID.\">clicca qui</A>";
    ?>
</body>
</html>
```

## Le sessioni: output



## Utilizzare un database

- La possibilità di interagire con i database è una delle potenzialità più interessanti offerte da PHP
- I database relazionali sono infatti lo strumento universalmente utilizzato per conservare basi di dati di qualsiasi dimensione
- PHP ci dà la possibilità di connetterci con un numero elevatissimo di database server (MySQL, PostgreSQL, Oracle, Access, Sybase, Informix, mSql ecc.)
- L'interazione con un database è il seguente:
  - Collegamento al database
  - Scelta del catalogo del database sul quale lavorare
  - Esecuzione di query (creazione, interrogazione, cancellazione)
  - Chiusura del collegamento al database

## Interazione con MySQL

- L'interazione viene ottenuta attraverso una serie di funzioni, il cui nome inizia sempre per "mysql" seguito da un underscore e dall'indicazione specifica della funzione
- Per chi avesse la necessità di utilizzare un database di tipo diverso, diciamo subito che **buona parte** di queste funzioni ha delle funzioni corrispondenti, relative agli altri tipi di database, che si differenziano in base al prefisso nel nome (ad esempio "pg" per PostgreSQL, "ora" per Oracle, "mssql" per Sql Server, etc.)

## Connessione al database

- La connessione si realizza attraverso la funzione **mysql\_connect**. Che necessita dei seguenti parametri:
  - **server** : l'URL del database
  - **utente** : nome di accesso
  - **password** : password dell'utente

```
mysql_connect(server, utente, password);
```

## Scelta del database

- Una volta stabilita la connessione, il passo successivo è selezionare il database col quale vogliamo lavorare. Per questo si usa la funzione **mysql\_select\_db** che necessita dei seguenti parametri:
  - **nomedb**: il nome del db al quale vogliamo connetterci
  - **connessione**: l'identificativo di connessione (cioè quello che abbiamo ottenuto da mysql\_connect)
- Questa funzione **restituisce** un valore booleano

```
mysql_select_db(nomedb, connessione);
```

## Esecuzione di una query

- Siamo quindi arrivati alla parte fondamentale del colloquio con un database, cioè l'esecuzione di una query. Per eseguire la query si usa la funzione **mysql\_query** i cui parametri sono:
  - **query**: query da eseguire
  - **connessione**: identificativo di connessione
- Anche questa funzione **restituisce** un valore, per il quale però dobbiamo distinguere due possibilità rispetto al tipo di query che abbiamo lanciato:
  - **query di interrogazione** (SELECT, SHOW, EXPLAIN, DESCRIBE), la funzione restituisce un **identificativo del risultato** (cioè un'altra variabile di tipo resource), che ci servirà successivamente, se la query è andata a buon fine; se invece MySQL ha rilevato degli errori, la funzione restituisce FALSE
  - **query di aggiornamento** (INSERT, UPDATE, DELETE), la funzione restituirà in ogni caso un valore booleano, ad indicare se l'esecuzione è andata a buon fine oppure no

## Verifica dei risultati della query

- Il fatto che una query sia stata eseguita correttamente non significa necessariamente che abbia prodotto dei risultati. Può infatti verificarsi il caso in cui una query, pur essendo perfettamente corretta, non produce alcun risultato
- Se vogliamo sapere **quante** righe sono state restituite da una SELECT, possiamo usare la funzione **mysql\_num\_rows(risultato)**, che ci **restituisce** il numero di righe contenute dall'identificativo del risultato che le passiamo.
- Se invece abbiamo eseguito una query di aggiornamento (INSERT, UPDATE, DELETE) e vogliamo sapere quante righe sono state modificate, possiamo usare **mysql\_affected\_rows(connessione)**, che ci **restituisce** il numero di righe modificate dall'ultima query di aggiornamento

## Lettura dei risultati di una SELECT

- Come abbiamo visto prima, una volta effettuata una query di interrogazione abbiamo a disposizione un identificativo del suo risultato. Per poter leggere questo risultato possiamo utilizzare la funzione **mysql\_fetch\_array(risultato)**, la quale, **ogni volta** che viene chiamata, ci **restituisce una riga** (array) del nostro risultato; quando non ci sono più righe da leggere, la funzione **restituisce FALSE**
- Quindi, per scorrere tutto il risultato, dovremo usare questa funzione come condizione di un ciclo, che si concluderà quando restituisce FALSE (in questo modo **non abbiamo bisogno** di sapere a priori quante sono le righe contenute nel risultato stesso)

```
$query = 'SELECT * FROM tabella';  
$ris = mysql_query($query,$conn) or  
    die("Errore nella query: " . mysql_error());  
while($riga = mysql_fetch_array($ris)) {  
    //codice che elabora i dati  
}
```

## Chiusura connessione

- Rimane da citare la funzione **mysql\_close** accetta come parametro
  - **connessione**: puntatore alla connessione aperta con `mysql_connect()`
- In pratica questa funzione è usata pochissimo, in quanto PHP si preoccupa da solo, al termine dello script, di chiudere le connessioni che abbiamo aperto

## Esempio di utilizzo PHP - MySQL

- Creiamo un applicativo che consenta di
  - Creare la struttura del database
  - Inserire dati nelle tabelle create
  - Visualizzare tutti i dati inseriti
  - Visualizzare un dato in particolare
- Realizzeremo il sistema mediante i seguenti file:
  - config.inc.php: configurazione del sistema
  - install.php: creazione del database
  - insert.php: form per l'inserimento dei dati
  - save.php: salvataggio dei dati nel database
  - index.php: visualizzazione di tutti i dati del database
  - view.php: visualizzazione completamente di un prodotto

### Es: Configurazione del sistema (config.inc.php)

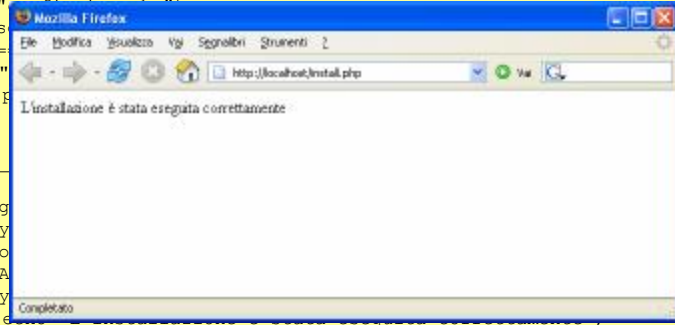
- Prima di tutto avremo bisogno di alcune informazioni relative all'accesso al database e altre relative all'applicazione:
  - **db\_host** da cui si può raggiungere MySQL (*localhost*)
  - **db\_user** e **db\_password** per l'accesso al database
  - **db\_nome** del database
  - **password** per la verifica dell'utente che esegue l'inserimento

```
CONFIG.INC.PHP
//parametri del database
$db_host = "localhost";
$db_user = "root";
$db_password = "prova";
$db_name = "test";
$password = "super";
```

## Es: Connessione e creazione del database (install.php)

- Effettuiamo la connessione al server database e scegliamo il database sul quale lavorare e creiamo una tabella sulla quale lavoreremo

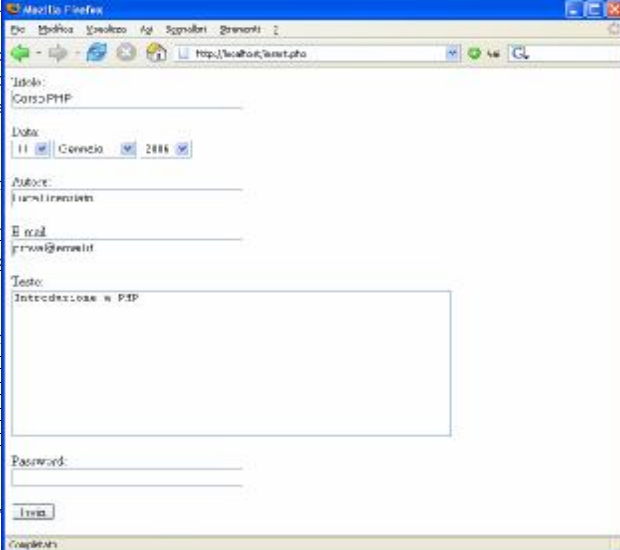
```
INSTALL.PHP
include("
$db = mys
if ($db =
die ("
}
else{
mysql_
or
config
$query
titolo
VARCHA
if (my
else
echo "Errore durante l'installazione";
}
```



file  
REMENT,  
) , autore

## Es: Form di inserimento dati (insert.php)

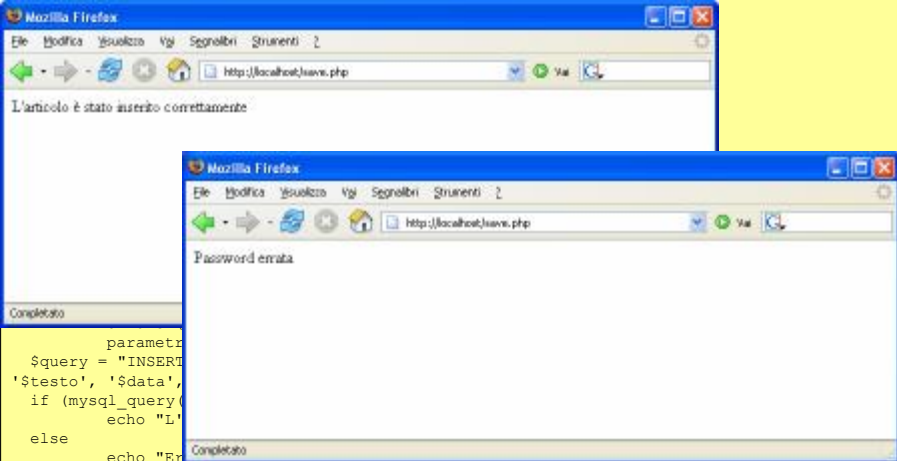
```
INSERT.PHP
<form metho
Titolo:<br>
<input type
<br>
Data:<br>
<select nar
<?php
for ($i=
echo "<op
?>
</select>
<select nar
<option va
<option va
<option va
<option va
...
</select>
```



re><br>  
><br>  
pass><br>  
a> </form>

## Es: Salvataggio dati nel database (save.php)

```
SAVE.PHP
include("config.inc.php");
if ($_POST[pass] != $password){
```



The image shows two overlapping Mozilla Firefox browser windows. The top window displays the message "L'articolo è stato inserito correttamente" (The article has been correctly inserted) and a "Completato" (Completed) status bar. The bottom window displays the message "Password errata" (Wrong password) and a "Completato" status bar. Both windows have the address bar set to "http://localhost/save.php".

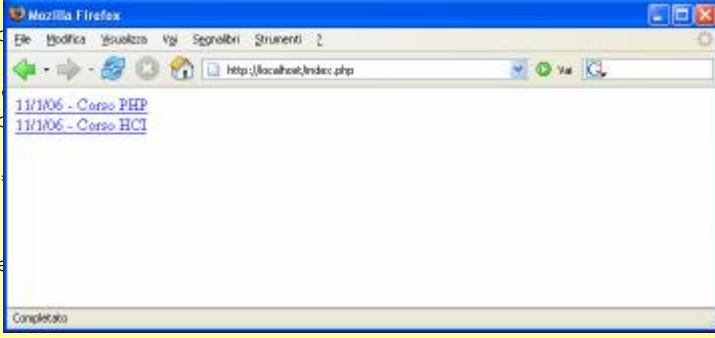
```

    paramet
    $query = "INSERT
    '$stesto', '$data',
    if (mysql_query(
        echo "L
    else
        echo "Er
    }
}

```

## Es: Lettura di un indice dei dati dal database (index.php)

```
INDEX.PHP
include("config.inc.php");
$db = mysql_connect($db_host, $db_user, $db_password);
if ($db
```



The image shows a Mozilla Firefox browser window displaying search results. The address bar shows "http://localhost/index.php". The page content lists two items: "11/106 - Corso PHP" and "11/106 - Corso HCI". The status bar at the bottom indicates "Completato".

```

    C
    metri
    mysql_s
    C
    ficare
    ESC
    $query
    $result
    while (
    e
    }
}

```

## Es: Lettura di un particolare dato (view.php)

### VIEW.PHP

```
include("config.inc.php");
$db = my... (word);
if ($db... e i parametri nel file

mysql_se... i

$query =
id='$_GE
$result
$row = m

$data =
echo "<br>";
echo "$row[...];
if ($row[mail] != "") ec
href=mailto:$row
else echo "$data, $row[a
echo "<br><a href=index.php>Torna alla pagina iniziale</a><br>";
```

