PHP

- introduzione
- il linguaggio di scripting
- interazione con gli oggetti HTTPresponse e HTTPrequest
- interazione con DB
- gestione delle sessioni

Open Source

Open Source Initiative (OSI) = non-profit corporation che ha l'obiettivo di gestire e promuovere l'*OpenSource*

Cos'è l'*Open Source*?

Il software *Open Source* deve rispettare una serie di **criteri** (si veda `http://www.opensource.org`)

- Il **codice sorgente** del programma deve essere disponibile e gratuito
- Il software può essere **modificato** e distribuito (con un nome diverso) alle stesse condizioni
- La licenza del software deve consentire a chiunque di **ridistribuire** il software in qualunque forma
- Chiunque può **partecipare** allo sviluppo del software
- Il software deve essere distribuito **gratuitamente** (senza diritti d'autore o profitti)
- …

Open Source

L'idea che sta alla base dell'***Open Source Initiative*** è la seguente: quando i programmatori hanno la possibilità di leggere, modificare e ridistribuire il **codice sorgente** di un programma, quel software **si evolve**. La gente lo migliora, lo adatta, lo corregge. E tutto questo può avvenire con una **rapidità** che appare impressionante a chi è abituato ai ritmi lenti dello sviluppo del software convenzionale.

La comunità dell'*Open Source* ha imparato che questo rapido processo evolutivo **produce software migliore** rispetto al tradizionale modello chiuso, nel quale solo pochissimi programmatori hanno accesso al codice sorgente e tutti gli altri devono avere a che fare con un imperscrutabile e oscuro blocco di bit.

[da **`http://www.opensource.org`**]

PHP: cos'è?

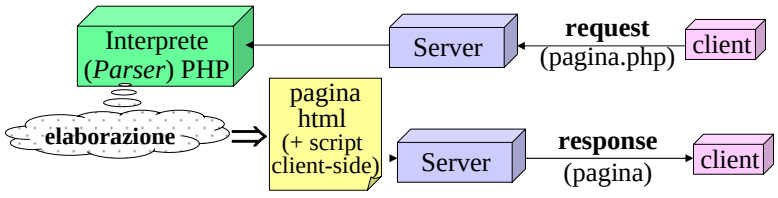
PHP = tecnologia ***server-side***, ***open source*** per realizzare siti Web dinamici

NOTA: **PHP** inizialmente (1994) significava *Personal Home Page*
Poi diventa *PHP Hypertext Preprocessor* (la versione corrente è PHP 5)

Applicazione PHP = insieme di pagine PHP

Pagina PHP = file di testo, con estensione ***.php***, che può contenere:

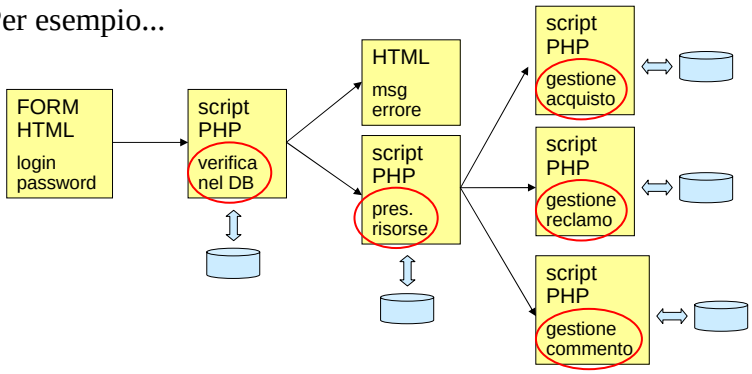
- HTML (+ script *client-side*; es. JavaScript) → interpretati dal **browser**
- Script *server-side* (PHP) → interpretati dal **server**



PHP: cos'è?

PHP è una tecnologia "***page-driven***":

- il "programma" server-side che gestisce la business logic dell'applicazione, interagisce con il database e genera l'interfaccia utente (pagine Web) è frammentato in una serie di **script interni alle pagine**
- Per esempio...



PHP: Web Server e interprete PHP

NB: Per utilizzare una tecnologia *server-side* è necessario che il **Web Server** sia in grado di interpretarla

	S.O.	Web Server	Tec. server-side
Tipicamente	Linux	Apache	PHP
Ma anche...	Windows	Apache	PHP
Oppure...	Windows	IIS	PHP

Apache Web Server (v. 2.0) si può scaricare dal sito:
http://httpd.apache.org/download.cgi
Documentazione su **Apache Web Server** (v. 2.0):
http://httpd.apache.org/docs/2.0/

PHP (v. 5.2.0) si può scaricare dal sito:
http://www.php.net/downloads.php
Documentazione su **PHP** (v. 5.2.0):
http://www.php.net/docs.php

PHP: EasyPhp e compagni

La soluzione più semplice su Windows è **EasyPhp** (http://www.easyphp.org/), un **pacchetto** che comprende:

- **Apache Web Server**
- **interprete PHP**
- **MySQL Server** (per l'interazione con un database)
  [lo vedremo più avanti…]

Download: http://www.easyphp.org/telechargements.php3

Documentazione per l'installazione:
http://www.easyphp.org/presentation.php3

Per **installare EasyPhp**: doppio click su *easyphp1-8_setup.exe*

Su Mac e Linux:

- MAMP (per Mac): http://www.mamp.info/en/home.php
- XAMPP (per Linux e Mac):
  http://www.apachefriends.org/en/xampp.html

PHP: Web Server e *document root*

- Quando installate un **Web server**, viene automanticamente creata la ***document root***, cioè la cartella (sul server) dove dovrete mettere le pagine php, html, le immagini, ecc... affinchè il server sia in grado di interpretarle
- Sul **client**, connettendovi con il **browser** al server, senza ulteriori indicazioni (per es. http://www.mydomain.it/), il server invia al client il contenuto della *document root*
- Se client e server risiedono sulla stessa macchina, potete connettervi al server (e così visualizzare i file contenuti nella *document root*) collegandovi all'URL http://localhost/

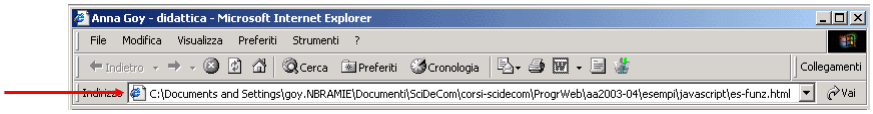
PHP: Web Server e *document root*

- Sotto la *document root* potete creare nuove cartelle, per es: corsoSW → per visualizzare i file contenuti nella cartella corsoSW, dovete connettervi a: http://localhost/corsoPW
- **NOTA:** la *document root* può essere modificata nel file di configurazione del Web Server
- Per esempio (*document root* di default):
  - **Apache** → C:\Programmi\Apache Group\Apache2\htdocs\

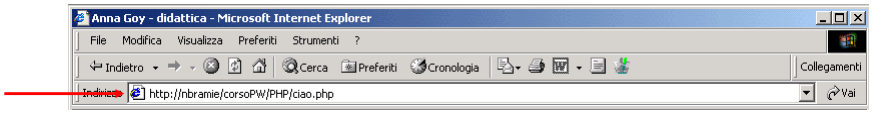
PHP: Web Server e browser

ATTENZIONE! Ricordatevi che ...

- per visualizzare una pagina Web che contiene **script client-side** NON HO bisogno di un server ⇒ posso aprire la pagina fornendo al browser il path sul file system locale (perché è il browser che interpreta gli script):



- per visualizzare una pagina Web che contiene degli **script server-side** HO bisogno di un server ⇒ devo connettermi al server (e richiedere la pagina) tramite un **URL** (perché è il server stesso che interpreta gli script)



PHP: linguaggio di scripting

Il **linguaggio di scripting** usato da PHP è… PHP!

- come tutti i linguaggi di scripting, è un linguaggio **interpretato**: il codice sorgente non deve essere compilato per essere eseguito
- è un linguaggio di **scripting server-side** e **NON** può essere utilizzato per scrivere degli script client-side
- è **case-sensitive**: `pippo` ≠ `Pippo` ≠ `PIPPO` ≠ `pipPo`
- interagisce con un Object Model, ma non lo fa esplicitamente ⇒ sintatticamente, non è un linguaggio object-oriented
- anche se, con PHP 5, il supporto all'interazione object-oriented con l'Object Model è completa!

PHP: script tag

Script Style:

```
<script language="php" runat="server" >
  script...
</script>
```

XML Style:

```
<?php
 script...
?>
```

Short style:

```
<?
  script...
?>
```

NB: attivato (default) nel file *php.ini*; in *php.ini* si può attivare anche il supporto per l'ASP Style: `<%  …  %>`

Nota: Negli esempi del corso useremo *short style*…

```
<?
  script...
?>
```

PHP: variabili

Non è necessaria una dichiarazione esplicita, ma i nomi delle **variabili** devono essere preceduti dal simbolo **$**, per es: **`$pippo`**

Gli **assegnamenti** si fanno con =, per es:

```
$pippo = 5;
```

assegnamento

Le **regole di visibilità** sono le solite

Tutte le istruzioni PHP devono **terminare con punto-e-virgola ;**

PHP: tipi

- Il **tipo** della variabile non viene dichiarato esplicitamente, ma definito implicitamente al primo assegnamento:

  `$pippo = 5;` → *pippo* è di tipo *Integer*
- PHP **converte** automaticamente i tipi durante l'esecuzione (quando possibile)

Tipi possibili in PHP:

- *Integer*: interi; es: *3*
- *Double/Float*: decimali (virgola mobile); es: *7.67*
- *String*: sequenze di caratteri; es: "pippo", "CV_45ie"
- *Boolean*: valori booleani (1/0 - vero/falso); es: *TRUE (1)*, *FALSE (0)*
- *Array*: liste con indice
- *Object*: oggetti [come in Java]; es: *data = new Date();*
- *Null*: "il Nulla" ;-) [come in Java]

PHP: operatori e commenti

Aritmetici: `+, -, *, /,`
`++ (i++ → i = i+1), -- (i-- → i = i-1)`

Di confronto: `==, !=, >, >=, <, <=`

Booleani: `&& (opp. and), || (opp.or), !`

Concatenazione (di stringhe): `.`
`("Buongiorno" . " a tutti" → "Buongiorno a tutti")`

Assegnamento: `= (totale = prezzo+iva),`

Commenti:

```
// commento su una riga
/* commento su + righe
   commento su + righe */
# commento su una riga
```


PHP: apici

VEDI es-apici.php

Nota:

```
<?
   $nome = "Anna";
   echo " Ciao $nome ! ";
   echo "<BR><BR>";
   echo ' Ciao $nome ! ';
?>
```

dentro ai doppi apici (") la variabile viene valutata!

"scrivi sulla pagina…"

dentro agli apici semplici (') la variabile **non** viene valutata!

⇒

```
Ciao Anna!
Ciao $nome!
```

PHP: liste (array)

Per rappresentare una **lista** (= sequenza ordinata di elementi) possiamo usare un **array**

Per scrivere:

```
$lista = array("primo", "secondo", "terzo");
```

oppure:

```
$lista[0] = "primo";
...
```

Per leggere:

```
$elem0 = $lista[0];
...
```

In generale:

```
$lista[i] = "iesimo";
$elem = $lista[i];
```

Per sapere quanti elementi ha un **array**:

```
$lungh = count($lista);
$lungh = sizeof($lista);
```


PHP: liste

E' possibile anche accedere agli elementi "chiamandoli per nome", anziché usare l'indice, creando una **lista associativa** (simile alle *HashMap* in Java…)

Per scrivere:

```
$lista = array("nome" ==> "anna",
               "cognome" ==> "goy", ...);
```

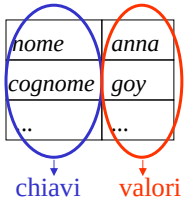
oppure:

```
$lista["nome"] = "anna";
$lista["cognome"] = "goy";
...
```

Per leggere:

```
$n = $lista["nome"];
$c = $lista["cognome"];
...
```

Possiamo immaginare una lista associativa come una tabella (in cui ogni riga è una coppia <*chiave*,*valore*>):



PHP: condizionali

```
if (condizione1) {
  sequenza_di_azioni_1
}
elseif (condizione2) {
  sequenza_di_azioni_2
}
else {
  sequenza_di_azioni_3
}
```

se la *condizione1* è vera esegui la *sequenza_di_azioni_1*
altrimenti, se la *condizione2* è vera esegui la *sequenza_di_azioni_2*
altrimenti esegui la *sequenza_di_azioni_3*

PHP: cicli

```
while (condizione) {
  sequenza_di_azioni
}
```

finché la *condizione* è vera esegui la *sequenza_di_azioni*

```
do {
  sequenza_di_azioni
} while (condizione);
```

esegui la *sequenza_di_azioni* finché la *condizione* è vera

```
for (inizio; test; increm.) {
    sequenza_di_azioni
}
```

esegui la *sequenza_di_azioni* a partire da *inizio*, finché *test* è vero, avanzando ad ogni passo di quanto è indicato da *increm.*

```
foreach (array as elem) {
    sequenza_di_azioni
}
```

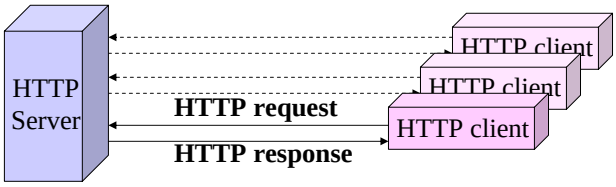
per ogni *elem* in *array* esegui la *sequenza_di_azioni*

PHP: gli oggetti *request* e *response*

HTTP

Un **Web Server** (un server che fornisce servizi sul Web) è sostanzialmente un ***HTTP Server*** (un server che comunica mediante il protocollo HTTP) e **gestisce 2 flussi di informazioni**:

- *HTTPrequest*: le richieste in arrivo dai client (*HTTP client*)
- *HTTPresponse*: le risposte del server, inviate ai client (*HTTP client*)



PHP: gli oggetti *request* e *response*

- L'oggetto ***HTTPrequest*** contiene tutte le informazioni relative alla richiesta che il client (browser) fa al server

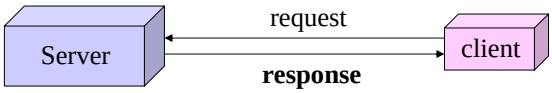
  Per es: l'utente compila un **modulo on-line** (*form* HTML); quando clicca sul pulsante di invio del modulo, parte la richiesta al server; l'oggetto *request* conterrà varie informazioni relative alla richiesta del client, tra cui tutti i **dati del modulo** (per es: i nomi dei campi e i rispettivi valori, il metodo utilizzato: GET o POST)

- L'oggetto ***HTTPresponse*** viene utilizzato dal server per inviare informazioni al client (browser)

  L'oggetto ***HTTPresponse*** contiene:

  - ***Header***: meta-informazioni (data, content-type, response-code: 200 se tutto ok, 404 se file not found, …)
  - ***Body***: contenuto (per es. la pagina web richiesta)

PHP: l'oggetto *response*



In PHP, per inviare una *response* al client:

```
<?
  echo "Hello!";
?>
<?
  print "Ciao!";
?>
```

NOTA: *echo* e *print* sono equivalenti: l'unica differenza è che echo funziona anche con più stringhe (separate da una virgola: `echo "pippo", "pluto", ...`)

NB: Non sono vere e proprie funzioni, per questo **le parentesi non sono obbligatorie**

PHP: l'oggetto *request*

Abbiamo detto che l'oggetto ***HTTPrequest*** contiene tutte le informazioni relative alla richiesta che il client (browser) fa al server; per es., se l'utente compila un **modulo on-line**, *request* contiene tutti i **dati del modulo**

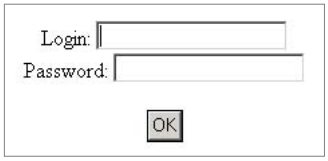
HTTPrequest può utilizzare due diversi metodi per inviare dati al server:

- **GET**: i dati, per es. i nomi dei campi e i valori del modulo, sono inviati al server in coda all'URL e sono quindi visibili
- **POST**: i dati, per es. i nomi dei campi e i valori del modulo, sono inviati al server come parte della richiesta HTTP e sono quindi "nascosti"

PHP: l'oggetto *request*

Abbiamo inoltre due modi per inviare esplicitamente dati (parametri = coppie <*attributo, valore*>) al server mediante *HTTP request*:

- **Moduli on-line**



HTTPrequest contiene:
`identif.php`
`<login,admin>`
`<password,pippo>`

indicato nell'attributo ACTION del tag FORM

- **Link**

Kit

```
<A HREF="param.php?risposta=kit&id=2">Kit</A>
```

HTTPrequest contiene:
`param.php`
`<risposta,kit>`
`<id,2>`

PHP: l'oggetto *request*

Se utilizziamo un **modulo on-line**, possiamo scegliere con quale metodo inviare i dati del modulo al server

- Se il metodo utilizzato è **GET**, i dati del modulo (nomi dei campi e valori) sono inviati al server **in coda all'URL** e sono quindi **visibili**:

```
http://localhost/corsoPW/
identif.php?login=admin&password=pippo
```

nome del campo valore (dato dall'utente nel modulo)

- Se il metodo utilizzato è **POST**, i dati del modulo (nomi dei campi e valori) sono inviati al server come **parte della richiesta HTTP** e sono quindi "**nascosti**"

Se utilizziamo un **link**, automaticamente **i dati sono inviati con il metodo GET** (quindi visibili nell'URL)

PHP: l'oggetto *request*

In **PHP**:

- se il metodo utilizzato è **GET** (**modulo** o **link**), i dati sono memorizzati come coppie <*attributo, valore*> nell'**array associativo**:

  `$_GET[]`

- se il metodo utilizzato è **POST** (**modulo**), i dati sono memorizzati come coppie <*attributo, valore*> nell'**array associativo** :

  `$_POST[]`

NOTA:

Nelle versioni precedenti alla 4.1 i due array associativi si chiamavano **`$HTTP_GET_VARS[]`** e **`$HTTP_POST_VARS[]`**

PHP: l'oggetto *request*

VEDI es-form2.html

File **es-form2.html**:

```
<HTML><BODY>
 <FORM METHOD="POST" ACTION="identif2.php" >
  Nome: <INPUT TYPE="TEXT" NAME="nome">
  Cognome: <INPUT TYPE="TEXT" NAME="cognome">
  Età: <SELECT NAME="eta">
   <OPTION VALUE="meno20">meno di 20</OPTION>
   <OPTION VALUE="tra20e40">tra 20 e 40</OPTION>
   <OPTION VALUE="tra40e60">tra 40 e 60</OPTION>
   <OPTION VALUE="piu60">più di 60</OPTION>
  </SELECT>
  M <INPUT TYPE="RADIO" NAME="sesso" VALUE="m">
  F <INPUT TYPE="RADIO" NAME="sesso" VALUE="f">
  <INPUT TYPE="Submit" VALUE="OK">
 </FORM>
</BODY></HTML>
```

PHP: l'oggetto *request*

VEDI identif2.php

NB: `$_POST` è un **array associativo**, in cui:

- **chiavi** = nomi del campi del modulo (form)
- **valori** = valori scritti/selezionati dall'utente

⇒ `array_keys($_POST)` → nomi dei campi
`array_values($_POST)` → valori
`$_POST ≡ array_values($_POST)`

File **identif2.php**:

```
<HTML><BODY>
 <?
   foreach (array_keys($_POST) as $campo) {
     echo "<P>" . $campo . " = ";
     $valore = $_POST[$campo];
     echo $valore . "</P>";
   }
 ?>
</BODY></HTML>
```


PHP: interazione tra `ACTION` e `onSubmit` (Javascript)

```
<HTML><BODY>
 <FORM METHOD="POST" ACTION="identif1.php"
 onSubmit = "alert('Benvenuto!'); return false;">
   Login: <INPUT TYPE="TEXT" NAME="login">
   Password: <INPUT TYPE="PASSWORD" NAME="password">
   <INPUT TYPE="Submit" VALUE="OK">
 </FORM>
</BODY></HTML>
```

Al click sul pulsante **`Submit`** [OK] cosa viene eseguito?

- l'invio di *HTTP request* per *identif1.php* (**`ACTION`**), oppure
- l'*alert* (**`onSubmit`**)?

Viene eseguito prima l'*alert* (**`onSubmit`**) e poi l'invio di *HTTP request* per *identif1.php* (**`ACTION`**)

NB: NON funziona se si aggiunge `return false;` !!!

PHP: l'oggetto *request*

VEDI es-param.php

VEDI quiz.html

Es: proponiamo all'utente un quiz (file **es-param.php**):

```
<HTML><BODY>
 <? include("quiz.html") ?>
</BODY></HTML>
```

File **quiz.html**:

```
<P ALIGN="CENTER">
 Come si chiama il figlio di Tex? <BR><BR>
 <A HREF="param.php?risposta=alan">Alan</A> <BR>
 <A HREF="param.php?risposta=kit">Kit</A> <BR>
 <A HREF="param.php?risposta=john">John</A>
</P>
```

passo a *param.php* una coppia nome-valore (`risposta=alan/kit/john`): aggiungendola in coda all'URL, in realtà la metto in `$_GET[]` (da dove poi potrò recuperarla…)

PHP: l'oggetto *request*

VEDI param.php

Finché la risposta è sbagliata, glielo riproponiamo; quando è corretta, gli inviamo un messaggio di congratulazioni

File **param.php**:

leggo il valore del parametro *risposta*, passato nell'URL dall'array $_GET

```
<HTML><BODY>
 <?
   $risp = $_GET["risposta"];
   if ($risp == "kit") {
 ?>
     <P ALIGN="CENTER">CONGRATULAZIONI!</P>
 <?
   }
   else {
 ?>
     <P ALIGN="CENTER">Risposta sbagliata: riprova!</P>
 <?
     include("quiz.html");
   }
 ?>
</BODY></HTML>
```

Databases: MySQL Server

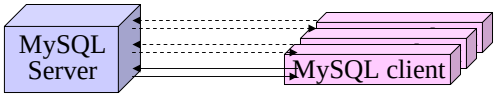
Tipicamente, con PHP si utilizza **MySQL Server**:

- **MySQL** (`www.mysql.com`) è un database **Server**, basato su **SQL**, multi-processo, multi-utente; è veloce e robusto
- **MySQL Server** è distribuito gratuitamente, con una licenza GNU (`www.gnu.org/gnu/thegnuproject.it.html`)
- La distribuzione di MySQL Server include il **database server (DBMS)** e diversi **client**: il database server risiede sulla macchina su cui si trovano i dati: riceve le richieste (query) dai client, accede al database e fornisce le risposte (risultati delle query)
  - ***MySQL*** = database server (DBMS)
  - ***mysql*** = un client
  - ***mysqladmin*** = programma per l'amministrazione

Databases: MySQL Server

Attenzione!

- Il DBMS MySQL funziona con **un'architettura client-server**



- Non confondete **MySQL Server** con il **Web (HTTP) Server** (per es. Apache)
- Aprire una **connessione al DB** significa:
  - Nel caso, per es, di **Microsoft Access**: creare un oggetto *Connection* e aprirlo passandogli il path del database (sul file system locale)
  - Nel caso di **MySQL Server**: aprire una connessione con MySQL Server e poi chiedergli di selezionare un database (senza indicargli alcun path)

PHP e MySQL: interazione

Cosa si deve fare per **accedere ad un DB** (MySQL) da una pagina web (PHP)?

All'interno di uno **script PHP** dobbiamo:

1. Aprire una **connessione** con MySQL Server
2. Selezionare un **database**
3. Inviare al DB Server una **query SQL**
4. Eventualmente, **estrarre** (e visualizzare) **i dati** contenuti nel risultato della query (*recordset*)
5. Chiudere la connessione

NB: la modalità di interazione che vedremo nelle prossime slide è un po' diversa da quella proposta da Wandschneider (vedi testi di riferimento)...

PHP e MySQL: interazione

1. Aprire una **connessione** con MySQL Server

   Per connettersi a MySQL Server bisogna avere:
   - il nome dell'**host** (o l'indirizzo IP) su cui risiede il DB Server
   - un **nome-utente** e una **password**

```
$db_conn = mysql_connect("localhost", "root", "")
 or die ("Non riesco a creare la connessione");
```

apro una connessione a MySQL Server: la funzione **`mysql_connect`** ha 3 argomenti: *db server*, *user-id*, *password*

NB: Se avete installato EasyPhp (senza modificare le impostazioni di default) MySQL Server è accessibile su *localhost* con *user-id*="root" e *password* = ""

Per verificare: nella prima pagina di phpMyAdmin cliccatre su *Privilegi*

Utente	Host	Password	Privilegi globali	Grant
root	localhost	No	ALL PRIVILEGES	Si

PHP e MySQL: interrogazione

```
$db_conn = mysql_connect("localhost", "root", "")
or die ("Non riesco a creare la connessione");
```

se la funzione **`mysql_connect`** provoca un errore, termina il processo e scrivi sulla pagina il messaggio (cioè, o va bene la funzione, oppure esegui la funzione **`die`**)

2. Selezionare un **database**

```
mysql_select_db("clienti") or die ("Non trovo il DB");
```

seleziono un db: la funzione **`mysql_select_db`** ha 2 argomenti: il **nome del db** (obbligatorio) e la **connnessione** (opzionale: se non viene specificata si usa l'ultima connessione aperta)

se la funzione **`mysql_select_db`** provoca un errore, termina il processo e scrivi sulla pagina il messaggio (cioè, o va bene la funzione, oppure esegui la funzione **`die`**)

VEDI provaDB1.php

PHP e MySQL: interazione

3. Inviare al DB Server una **query SQL**:
   a. **definire** la query
   b. **inviarla** al DB Server

```
$sql = "SELECT nome, cognome, punti FROM dati_clienti
                                    WHERE punti > 0";
$ris = mysql_query($sql) or die ("Query fallita!");
```

creo una **stringa** contenente la **query SQL**…

...e la invio al DB Server: la funzione **`mysql_query`** ha 2 argomenti [in realtà ne ha 3, ma il terzo non lo commentiamo...]: il **nome del db** (obbligatorio) e la **connnessione** (opzionale: se non viene specificata si usa l'ultima connessione aperta)

`$ris` conterrà il **risultato della query** (diverso a seconda del tipo di query: SELECT, INSERT, UPDATE, DELETE, ...)

Supponiamo di aver inviato una SELECT...

PHP e MySQL: interazione

4. Eventualmente [nel caso di una SELECT], **estrarre** (e visualizzare) **i dati** contenuti nel risultato della query (*recordset*):

```
while ($riga = mysql_fetch_array($ris)) {
  echo $riga["nome"] . " ";
  echo $riga["cognome"] . ": ";
  echo $riga["punti"] . " punti";
}
```

con un ciclo, leggo uno per uno i record contenuti nel risultato della query: la funzione **`mysql_fetch_array`** estrae dal suo argomento **`($ris)`** i record uno per volta (ad ogni ciclo);
il record estratto di volta in volta viene messo nella variabile **`$riga`**; quando non ci sono più record da estrarre, la condizione del *while* diventa automaticamente falsa (e il ciclo termina)

→ il **record** corrente è una **lista** (array) **associativa**:
`$riga[`*`nomecampo`*`]` estrae il valore del campo *nomecampo*;
`echo` lo scrive sulla pagina

PHP e MySQL: interazione

5. Chiudere la connessione:

```
mysql_close();
```

chiudo la connessione a MySQL Server (la funzione **`mysql_close`** ha un argomento opzionale, la **connnessione**: se non viene specificata si usa l'ultima connessione aperta)

In questo primo esempio abbiamo visto l'uso di una query di tipo SELECT, cioè abbiamo **letto dei dati dal database** (e li abbiamo visualizzati sulla pagina web) = **interrogazione** del database

Vediamo adesso come fare a:
- inserire nuovi dati (un nuovo record)
- modificare dati in un record esistente
- cancellare dati (cancellare un record)

NB: l'unica cosa che cambia è la query SQL (e naturalmente viene omesso il passo 4)

VEDI provaDB2.php

PHP e MySQL: inserimento

Per **inserire** un nuovo record (scrittura):

```
<?
   $db_conn = mysql_connect("localhost", "root", "")
         or die ("Non riesco a creare la connessione");
```
apro una connessione a MySQL Server

```
   mysql_select_db("clienti")
   or die ("Non trovo il DB");
```
seleziono un DB

```
   $sql = "INSERT INTO dati_clienti (nome, cognome, email,
      punti) VALUES ('Gianni','Verdi','gianni@tin.it',0)";
   $ris = mysql_query($sql)
         or die ("Query fallita!");
```
creo una query SQL…
e la invio al DB Server

```
   mysql_close();
?>
```
chiudo la connessione a MySQL Server

VEDI provaDB3.php

PHP e MySQL: modifica

Per **modificare** dei dati (scrittura - update):

```
<?
   $db_conn = mysql_connect("localhost", "root", "")
         or die ("Non riesco a creare la connessione");
```
apro una connessione a MySQL Server

```
   mysql_select_db("clienti")
   or die ("Non trovo il DB");
```
seleziono un DB

```
   $sql = "UPDATE dati_clienti SET punti=10
           WHERE cognome='Verdi'";
   $ris = mysql_query($sql)
         or die ("Query fallita!");
```
creo una query SQL…
e la invio al DB Server

```
   mysql_close();
?>
```
chiudo la connessione a MySQL Server

VEDI provaDB4.php

PHP e MySQL: cancellazione

Per **cancellare** un record (scrittura):

apro una connessione a MySQL Server

```
<?
   $db_conn = mysql_connect("localhost", "root", "")
         or die ("Non riesco a creare la connessione");
   mysql_select_db("clienti")
   or die ("Non trovo il DB");
```

seleziono un DB

```
   $sql = "DELETE FROM dati_clienti WHERE cognome='Verdi'";
   $ris = mysql_query($sql)
         or die ("Query fallita!");
```

creo una query SQL…

e la invio al DB Server

```
   mysql_close();
?>
```

chiudo la connessione a MySQL Server

PHP e MySQL: parametri connessione

Per evitare di avere i **parametri necessari alla connessione** ripetuti in molti file (per modificarli, occorrerebbe editare e modificare tutti i file che li contengono) è buona norma definire, **in un file separato**, tre variabili; tale file viene incluso quando necessario

Per es, definiamo un file, *varDB.php*, in cui:

```
<?
   $host="localhost";
   $user="root";
   $pwd="";
?>
```

PHP e MySQL: parametri connessione

... e dall'interno della pagine che accedono ai database **includiamo il file**:

Per es:

```
<?
  include("varDB.php");
  ...
  $db_conn = mysql_connect($host, $user, $pwd) or die...
  ...
?>
```

NB: le variabili *$host*, *$user*, *$pwd* sono definite nel file *varDB.php*

PHP e MySQL: miglioramenti...

Le interazioni viste sin qui sono i "mattoncini" di base; affinché un'interazione sia "corretta" sono necessari maggiori controlli

Per es, quando inseriamo un nuovo utente nel database è bene controllare che non sia già presente

Supponiamo di dover inserire il cliente *Gianni Verdi (email = gianni@tin.it)*

⇒ il passo 3 diventa:

```
$sql_1 = "SELECT * FROM dati_clienti
                  WHERE email = 'gianni@tin.it'";
$ris_1 = mysql_query($sql_1) or die ("Query 1 fallita!");
if (mysql_num_rows($ris_1) != 0) {
  echo "l'email è già presente nel database";
}
else {
  $sql_2 = "INSERT INTO dati_clienti (nome, cognome, email,
            punti) VALUES ('Gianni','Verdi','gianni@tin.it',0)";
  $ris_2 = mysql_query($sql_2) or die ("Query 2 fallita!");
}
```

PHP e MySQL: miglioramenti...

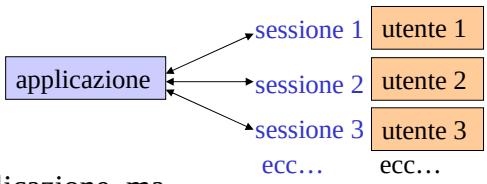
NB:

- `mysql_num_rows(risID)` restituisce il numero di righe (record) presenti nel risultato della query (recordset = *risID*)
  *** usato con SELECT ***
- `mysql_affected_rows()` restituisce il numero di righe (record) modificate dalla query
  *** usato con INSERT, UPDATE, DELETE ***

Per es, se volete controllare l'esito di un'UPDATE, potete fare:

```
$sql = "UPDATE dati_clienti SET punti='10'
        WHERE cognome='Verdi'";
$ris = mysql_query($sql) or die ("Query fallita!");
if (mysql_affected_rows() == 0) {
  echo "Attenzione! L'update non è stato effettuiato";
}
```


PHP: sessioni



Il Web Server **crea automaticamente un oggetto *session*** per ogni utente che si connette all'applicazione, ma…
PHP non è un linguaggio orientato agli oggetti! ⇒ non si fa riferimento esplicito a tale oggetto…

`session_start();` → funzione che verifica se l'utente ha già un sessionID (cioè se c'è già una sessione attiva): se non lo trova ne crea uno, altrimenti "rende visibili" le **variabili di sessione**

NB: anche se è controintuitivo, `session_start()` deve essere **invocata ogni volta** che si accede a delle informazioni sulla sessione!

PHP: sessioni

Le **variabili di sessione** sono contenute nell'**array associativo** $_SESSION, che contiene una **lista di coppie** <*nome*, *valore*>

Se abbiamo bisogno di una variabile "globale al sito", cioè **accessibile (visibile) da tutte le pagine del sito**, possiamo dichiarare e inizializzare una **variabile di sessione**:

```
session_start();
$_SESSION["s_globale"] = "ok";
```

In **tutte le pagine** successive (visitate dopo aver inizializzato la variabile di sessione) sarà possibile **leggerne il valore**; **NB:** se questo non è possibile, significa che **la sessione è scaduta**!

```
session_start();
if (isset($_SESSION["s_globale"])) {
 //sessione valida
 ... $_SESSION["s_globale"] ...
} else
 //sessione scaduta...
}
```

funzione che restituisce *true* se la variabile (passata come parametro) esiste, *false* altrimenti

PHP: sessioni

Note:

Le informazioni sulla sessione (session-id, ecc.) vengono salvate nella cartella indicata nel file di configurazione dell'interprete PHP (*php.ini*):

```
session.save_path = "${path}\tmp\"
```

⇒ la cartella *tmp* deve esistere ed essere scrivibile!

Per gestire la sessione, PHP utilizza dei **cookies** (file temporanei che il server salva sul client); i cookies sono inviati nell'**intestazione** (*header*) di *HTTP response*

⇒ bisogna invocare le funzioni relative alla sessione prima che il contenuto (*body*) di *HTTP response* venga generato, cioè **all'inizio della pagina** (prima di qualunque altra cosa, anche di uno spazio bianco...)

PHP: un esempio complessivo

Vediamo adesso un **esempio complessivo** che mette insieme:

1. gestione della sessione utente
2. gestione di form, cioè interazione con request (e response)
3. interazione con un database

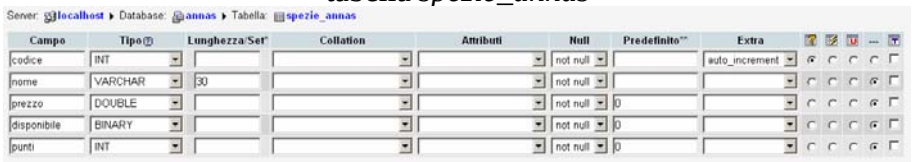
ANNA'S
negozio di spezie on-line

ANNA'S: database

database = *annas*

tabella *clienti_annas*

tabella *spezie_annas*



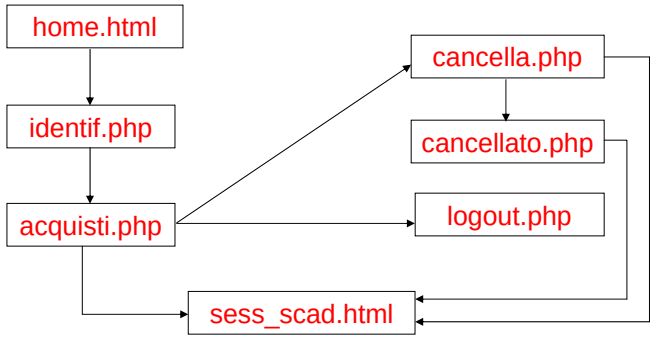
ANNA'S: database

tabella *acquisti_annas*

contenuto tabella *spezie_annas* (catalogo)

	codice	nome	prezzo	disponibile	punti
	1	chili	2.5	1	20
	2	cumino	2	1	10
	3	origano	1.3	0	5
	4	zafferano	10.6	1	30

ANNA'S: flusso di interazione



ANNA'S: funzionalità delle pagine

home.html	- login/registrazione utente
identif.php	- controllo login (utente registrato) o registrazione (nuovo utente) - impostazione nuova sessione - visualizzazione catalogo
acquisti.php	- controllo validità sessione - gestione acquisti
cancella.php	- richiesta cancellazione utente
cancellato.php	- cancellazione utente
logout.php	- chiusura sessione
sess_scad.html	- notifica sessione scaduta

ANNA'S: funzionalità delle pagine (dettaglio)

home.html

```
<FORM METHOD="POST" ACTION="identif.php?utente=registrato">
 <P>
  E-mail: <INPUT TYPE="TEXT" NAME="email">
  <BR>
  Password: <INPUT TYPE="PASSWORD" NAME="pwd">
  <BR><BR>
  <INPUT TYPE="Submit" VALUE="entra">
 </P>
</FORM>
```

La form per il nuovo utente è analoga...

ANNA'S: funzionalità delle pagine (dettaglio)

identif.php

All'inizio, prima del tag **<HTML>** (in realtà prima di **qualunque cosa!** anche di uno spazio bianco...)

```
<?
//attiviamo una nuova sessione utente
  session_start();
//inclusione file con definizione variabili per collegamento db
  include("varDB.php");
//imposto la connessione al DB Server e seleziono il DB
  $db_conn = mysql_connect($host, $user, $pwd) or
             die ("Non riesco a creare la connessione");
  mysql_select_db("annas") or
  die ("Non riesco a trovare il database annas");
//lettura dei dati dall'url (get)
  $utente = $_GET["utente"];
  if ($utente == "registrato") {
//lettura dei dati dal form (post)
   $email = $_POST["email"];
   $pwd = $_POST["pwd"];
...
```


ANNA'S: funzionalità delle pagine (dettaglio)

```
//controllo nel DB email e password
   $sql_reg = "SELECT * FROM clienti_annas
               WHERE email='".$email."' AND password='".$pwd."'";
   $ris_reg = mysql_query($sql_reg) or
              die ("Query utente registrato fallita!");
   if (mysql_num_rows($ris_reg) == 0) { //se il recordset è vuoto
    $messaggio = "<P><B>Login sbagliato!</B><BR>
                  <A HREF='home.html'>Riprova</A></P>";
   }
   else {
//se il login e' OK, imposto una variabile di sessione
    $_SESSION["s_login"]="ok";
//leggo nome e cognome dal DB
//(assumo che $ris_reg contenga un solo record)
    $riga = mysql_fetch_array($ris_reg);
    $nome = $riga["nome"];
    $cognome = $riga["cognome"];
   }
  }
  else { //utente nuovo
//lettura dei dati dal form (post)
   $nome = $_POST["nome"];
   $cognome = $_POST["cognome"];
   ...
```

ANNA'S: funzionalità delle pagine (dettaglio)

```
   ...
   $email = $_POST["email"];
   $pwd = $_POST["pwd"];
//controllo nel DB che l'utente non sia gia' presente...
//se non c'e', lo inserisco
   $sql_new_1 = "SELECT * FROM clienti_annas
                 WHERE email = '" . $email . "'";
   $ris_new_1 = mysql_query($sql_new_1) or
                die ("Query 1 nuovo utente fallita!");
   if (mysql_num_rows($ris_new_1) != 0) {
    $messaggio = "<P><B>L'email " . $email . " è già presente nel
                  database: impossibile inserire!</B><BR>
                  <A HREF='home.html'>Torna indietro</A></P>";
   }
   else {
    $sql_new_2 = "INSERT INTO clienti_annas (nome,cognome,email,password)
     VALUES ('".$nome."','".$cognome."','".$email."','".$pwd."')";
    $ris_new_2 = mysql_query($sql_new_2) or
                 die ("Query 2 nuovo utente fallita!");
//imposto una variabiledi sessione
    $_SESSION["s_login"]="ok";
   }
  }
?>
```


ANNA'S: funzionalità delle pagine (dettaglio)

Nel body

```
<?
//se il login (nuovo o registrato) e' avvenuto correttamente
//mostro il catalogo
  if (isset($_SESSION["s_login"]) and $_SESSION["s_login"] == "ok") {
   echo "<P>Buongiorno " . $nome . " " . $cognome . "!";
   echo "<BR>";
   echo "<B>Cosa vuoi comprare?</B>";
   echo "</P>";
   echo "<FORM METHOD=\"POST\"
                ACTION=\"acquisti.php?email=".$email."\">";
   echo "<P>";
//lettura dei prodotti dalla tabella spezie_annas
   $sql_spe = "SELECT nome, prezzo, punti FROM spezie_annas
               WHERE disponibile=1";
   $ris_spe = mysql_query($sql_spe) or
              die ("Query spezie fallita!");
   while ($riga_spe = mysql_fetch_array($ris_spe)) {
    echo "<INPUT TYPE='checkbox' NAME='".$riga_spe["codice"]."'>";
    echo $riga_spe["nome"];
    echo " - <FONT COLOR='red'>Vale ".$riga_spe["punti"]." punti</FONT>";
    echo " [".$riga_spe["prezzo"]." euro]<BR>";
   }
   ...
```

ANNA'S: funzionalità delle pagine (dettaglio)

```
    ...
    echo "<BR><BR>";
    echo "<INPUT TYPE='Submit' VALUE='OK'>":
    echo "</P>";
    echo "</FORM>";
   }
   else {//se ci sono stati problemi di login (nuovo o reg.)
    echo $messaggio;
   }
   mysql_close();
?>
```


ANNA'S: funzionalità delle pagine (dettaglio)

acquisti.php

All'inizio, prima del tag **<HTML>** (in realtà prima di **qualunque cosa!** anche di uno spazio bianco...)

```
<?
//controllo il valore della variabile di sessione s_login
/* se vale "ok" vuol dire che l'utente è correttamente passato dalla
   pagina identif.php, altrimenti (se s_login != "ok") vuol dire che
   la sessione avviata in identif.php non è (piu') valida
 */
  session_start();
  if (!isset($_SESSION["s_login"]) or $_SESSION["s_login"] != "ok") {
   include ("sess_scad.html");
  }
  else {
?>
```

NB: questo else va chiuso al fondo della pagina!!!

```
...
</HTML>
<?
  }
?>
```

ANNA'S: funzionalità delle pagine (dettaglio)

Nel body

```
<?
//lettura dell'email dall'url (get)
  $email = $_GET["email"];
//inclusione file con definizione variabili per collegamento db
  include("varDB.php");
//lettura dati form e ricerca punti/prezzo nel DB (tab. spezie_annas)
  $db_spe = mysql_connect($host, $user, $pwd) or
            die ("Non riesco a creare la connessione");
  mysql_select_db("annas") or
  die ("Non riesco a trovare il database annas");
  $punti_nuovi = 0;
  $costo = 0.0;
//per ogni spezia acquistata, registro nel DB l'acquisto
//(tabella acquisti_annas) e visualizzo i dati
  foreach (array_keys($_POST) as $campo) {
    $sql_spe1 = "INSERT INTO acquisti_annas (email_cliente,id_spezia)
                 VALUES ('".$email."', ".$campo.")";
    $ris_spe1 = mysql_query($sql_spe1) or
                die ("Query 1 spezie fallita!");
```


ANNA'S: funzionalità delle pagine (dettaglio)

```
    $sql_spe2 = "SELECT * FROM spezie_annas
                 WHERE codice = '".$campo."'";
    $ris_spe2 = mysql_query($sql_spe2) or
                die ("Query 2 spezie fallita!");
    $riga_spe = mysql_fetch_array($ris_spe2);
    echo $riga_spe["nome"]." - ";
    echo $riga_spe["punti"]." punti <BR>";
    $punti_nuovi = $punti_nuovi + $riga_spe["punti"];
    $costo = $costo + $riga_spe["prezzo"];
  }
  echo "<BR>Con la tua spesa hai totalizzato ".$punti_nuovi." punti!";
  echo "<BR>... e hai speso ".$costo." euro.";
//lettura punti precedenti del cliente dal DB (tab. clienti_annas)
  $sql_cli1 = "SELECT punti FROM clienti_annas
               WHERE email = '".$email."'";
  $ris_cli1 = mysql_query($sql_cli1) or
              die ("Query (SELECT) fallita!");
//(assumo che $ris_cli1 contenga un solo record)
  $riga_pun = mysql_fetch_array($ris_cli1);
  $punti_vecchi = $riga_pun["punti"];
  $punti_totali = $punti_vecchi + $punti_nuovi;
  ...
```

ANNA'S: funzionalità delle pagine (dettaglio)

```
//aggiorno totale punti accumulati dal cliente (tab. clienti_annas)
  $sql_cli2 = "UPDATE clienti_annas SET punti='".$punti_totali."'
              WHERE email='".$email."'";
  $ris_cli2 = mysql_query($sql_cli2) or die ("Query fallita!");
  echo "<P><B>PUNTI TOTALI</B>: ".$punti_totali."</P>";
  mysql_close();
?>
<!-- link (gestito con javascript) per cancellare la registrazione -->
<P>
 <A HREF="#"
<?
 echo "onClick=\"window.open('cancella.php?email=".$email."',
       'cancella', ";
 echo "'scrollbars=yes,resizable=yes,width=500,height=600,
       status=no,location=yes,toolbar=no,menubar=no');
        return false;\"";
?>
 >Vuoi cancellare la tua registrazione?</A>
</P>
```

primo parametro della funzione *open*

secondo parametro

terzo parametro

ANNA'S: funzionalità delle pagine (dettaglio)

cancella.php

All'inizio, prima del tag **<HTML>**: controllo della sessione come in *acquisti.php*

Nel body

```
<?
//lettura dell'email dall'url (get)
  $email = $_GET["email"];
  echo "<FORM METHOD=\"POST\"
              ACTION=\"cancellato.php?email=".$email."\">";
?>
<P>
 SI <INPUT TYPE="radio" NAME="canc" VALUE="si">
 NO <INPUT TYPE="radio" NAME="canc" VALUE="no">
 <BR><BR>
 <INPUT TYPE="Submit" VALUE="OK">
</P>
</FORM>
```

ANNA'S: funzionalità delle pagine (dettaglio)

cancellato.php

All'inizio, prima del tag **<HTML>**: controllo della sessione come in *acquisti.php*

Nel body

```
...
//lettura dell'email dall'url (get)
  $email = $_GET["email"];
//lettura dei dati dell'utente dal DB
  ...
  $sql1 = "SELECT * FROM clienti_annas WHERE email='".$email."'";
  $ris1 = mysql_query($sql1) or die ("Query fallita!");
 //(assumo che $ris1 contenga un solo record)
  $riga = mysql_fetch_array($ris1);
  $cognome = $riga["cognome"];
//lettura della form (post): si'/no
  $canc = $_POST["canc"];
  ...
```


ANNA'S: funzionalità delle pagine (dettaglio)

```
if ($canc == "si") {
 $sql2 = "DELETE FROM clienti_annas WHERE email='".$email."'";
 $ris2 = mysql_query($sql2) or die ("Query fallita!");
 echo "<P><B>".$cognome." cancellato!</B></P>";
}
else {
 echo "<P><B>".$cognome." NON è stato cancellato...</B></P>";
}
...
```

NOTA

Per capire **a cosa serve il controllo di sessione** provate ad accedere direttamente alla pagina *acquisti.php*

(**NB:** se prima avete navigato nel negozio, cliccate su *Esci (chiudi sessione)* prima di fare questa prova!)

ANNA'S: funzionalità delle pagine (dettaglio)

logout.php

All'inizio, prima del tag **<HTML>**:

```
<?
  session_start();
  if (isset($_SESSION["s_login"]) and $_SESSION["s_login"] == "ok") {
    session_unset();      → "azzera" tutte le variabili di sessione
    session_destroy();    → "distrugge" (chiude) la sessione
  }
?>
```


PHP: uso delle variabli di sessione

Nell'esempio abbiamo passato l'identificativo dell'utente (email) come **parametro dell'url**

Un modo alternativo per "passare" dei dati da una pagina all'altra è archiviarli in **variabili di sessione**, che sono disponibili (accessibili) da qualunque pagina del sito (purché la sessione sia valida)

Per es, per passare, da *identif.php* a *acquisti.php* (e poi a *cancella.php* e poi ancora a *cancellato.php*) l'email dell'utente avremmo potuto:

- in *identif.php* **inizializzare una variabile di sessione**:
  `$_SESSION["email_utente"]= $_POST["email"];`
- in *acquisti.php* **leggere questa variabile** (anzichè *$_GET*):
  `$email = $_SESSION["email_utente"];`