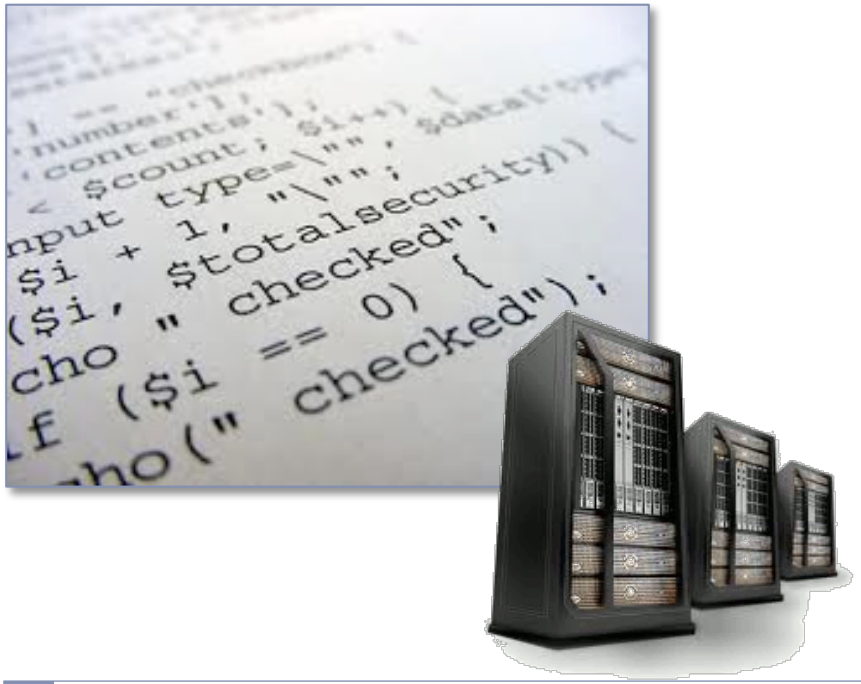
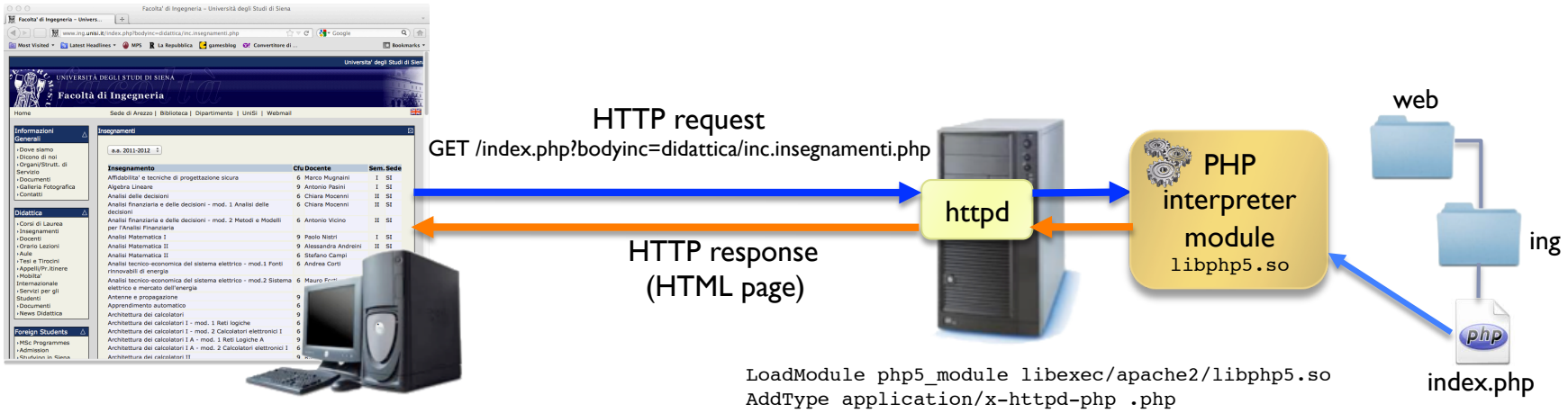


Reti di Calcolatori

Server-side programming & PHP

PHP

- Il PHP (Personal Home Page o PHP: Hypertext Preprocessor) è un linguaggio di scripting server-side utilizzato per lo sviluppo di pagine Web dinamiche
  - Il linguaggio è di tipo HTML-embedded ovvero le istruzioni sono inserite in un file HTML e vengono interpretate prima di essere trasmesse al client
  - E' un software open source disponibile su più piattaforme e compatibile con i principali server Web (es. apache, IIS)
  - Tecnologie simili sono ASP di Microsoft (Active Server Pages) e JSP (Java Server Pages)



Caratteristiche di PHP

- PHP è un linguaggio di scripting pensato per semplificare la programmazione lato server con caratteristiche orientate al Web ed una ampia dotazione di librerie
  - Può essere usato come linguaggio procedurale o ad oggetti
  - Fornisce il supporto per l'accesso ai dati inviati da form HTML
  - Permette di accedere ai valori di header della richiesta HTTP e di valorizzare header della risposta
  - Ha il supporto per la gestione dei cookie e delle sessioni
  - L'uso del PHP non è limitato alla generazione di HTML e file di testo in genere, ma sono a disposizione librerie di funzioni per la generazione di vari formati (es. immagini, pdf)
  - Sono disponibili librerie la connessione a vari DBMS (es. MySQL, PostsgreSQL,..) e per la gestione di svariati protocolli di rete (es. IMAP, POP3, LDAP,..)
  - Ha il supporto per l'uso di socket, per l'elaborazione avanzata di testi (es. espressioni regolari), per l'analisi e l'elaborazione di documenti XML

HTML-embedding

- Uno script PHP è un file di testo dove il codice PHP eseguibile è inserito fra i tag **<?php ... ?>**
  - L'interprete funziona da filtro ovvero invia in uscita il testo al di fuori dei tag **<?php.. ?>** ed esegue le istruzioni al loro interno
  - Le istruzioni creano contenuto dinamico che viene inserito al posto del blocco compreso fra i tag **<?php .. ?>**

```
<html>
  <body>
   <h2>Che ora &egrave; per il server?</h2>
    <?php
      /*
       *  istruzioni PHP
       */
      # scrive il nome del server
      echo "<b>Server: </b>".$_SERVER['SERVER_NAME']."<br />";
      // scrive la data del server
      date_default_timezone_set("Europe/Rome");
      echo "\n<b>Data: </b>".date("F j, Y, g:i a")."\n";
    ?>
  </body>
</html>
```

```
<html>
 <body>
   <h2>Che ora &egrave; per il server?</h2>
   <b>Server: </b>127.0.0.1<br />
   <b>Data: </b> April 8, 2012, 4:46 pm
 </body>
</html>
```

Che ora è per il server?

Server: 127.0.0.1
Data: April 8, 2012, 4:46 pm

Ancora su HTML-embedding...

- Il “modo” PHP si può interrompere e riaprire in ogni momento
  - In ogni caso il flusso di uscita segue quello di esecuzione delle istruzioni ovvero il testo HTML in una parte interna ad un blocco PHP non eseguito non viene inviato in uscita

```
<h2>Che browser usi?</h2>
Client IP: <?php  echo $_SERVER['REMOTE_ADDR'];?> <br />
Client Port: <?php  echo $_SERVER['REMOTE_PORT'];?> <br />
Browser model:
<?php $agent = $_SERVER['HTTP_USER_AGENT'];
      if(strpos($agent,"Firefox")) { ?>
  <!-- questo HTML è nel blocco true del primo if -->
  <em style="color: orange">Firefox</em>
<?php } else if(strpos($agent,"Chrome")) { ?>
  <!-- questo HTML è nel blocco true del secondo if -->
  <em style="color: green">Chrome</em>
<?php } else if(strpos($agent,"Safari")) { ?>
  <!-- questo HTML è nel blocco true del terzo if -->
  <em style="color: gold">Safari</em>
<?php } else { ?>
  <!-- questo HTML è nel blocco false del terzo if -->
  <em style="color: red">Unsupported browser</em>
<?php } ?>

<br />User-agent string: <em><?php echo $agent;?></em>
```

Che browser usi?

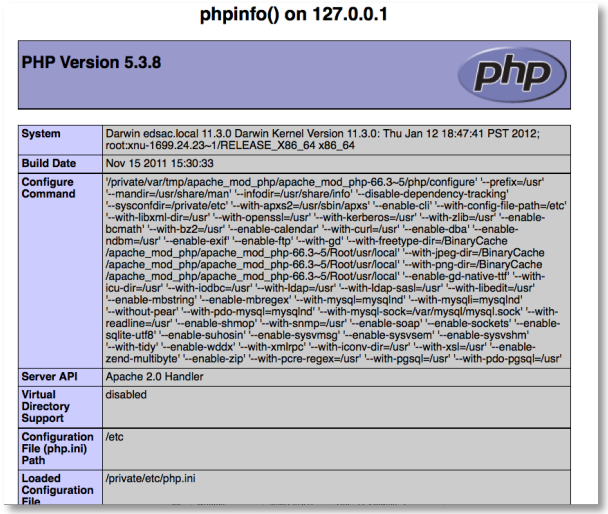
Client IP: 127.0.0.1
Client Port: 51921
Browser model: *Safari*
User-agent string: *Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/534.55.3 (KHTML, like Gecko) Version/5.1.5 Safari/534.55.3*

```
<h2>Che browser usi?</h2>
Client IP: 127.0.0.1 <br />
Client Port: 51921 <br />
Browser model:
<!-- questo HTML è nel blocco true del
terzo if -->
<em style="color: gold">Safari</em>
<br />User-agent string: <em>Mozilla/5.0
(Macintosh; Intel Mac OS X 10_7_3)
AppleWebKit/534.55.3 (KHTML, like Gecko)
Version/5.1.5 Safari/534.55.3</em>
```

Configurazione di PHP

- Il file **`php.ini`** permette di configurare l'interprete PHP
  - Il file è commentato in dettaglio e la configurazione di default di solito è adeguata
  - La configurazione del PHP può essere visualizzata con uno script che chiama la funzione **`phpinfo()`**

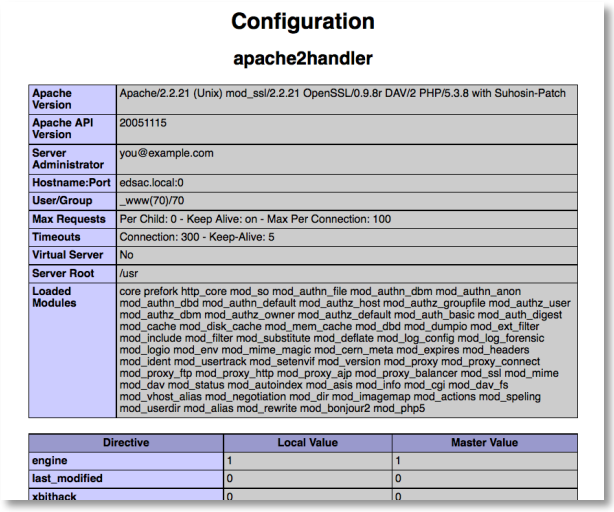
```
<body>
 <h2 style="text-align: center">
   phpinfo() on <?php echo $_SERVER['SERVER_NAME'];?>
 </h2>
 <?php phpinfo(); ?>
<body>
```



phpinfo() on 127.0.0.1

PHP Version 5.3.8

System	Darwin edsac.local 11.3.0 Darwin Kernel Version 11.3.0: Thu Jan 12 18:47:41 PST 2012; root:xnu-1699.24.23~1/RELEASE_X86_64 x86_64
Build Date	Nov 15 2011 15:30:33
Configure Command	'/private/var/tmp/apache_mod_php/apache_mod_php-66.3~5/php/configure' '--prefix=/usr' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--disable-dependency-tracking' '--sysconfdir=/private/etc' '--with-apxs2=/usr/sbin/apxs' '--enable-cli' '--with-config-file-path=/etc' '--with-libxml-dir=/usr' '--with-openssl=/usr' '--with-kerberos=/usr' '--with-zlib=/usr' '--enable-bcmath' '--with-bz2=/usr' '--enable-calendar' '--with-curl=/usr' '--enable-dba' '--enable-ndbm=/usr' '--enable-exif' '--enable-ftp' '--with-gd' '--with-freetype-dir=/BinaryCache /apache_mod_php/apache_mod_php-66.3~5/Root/usr/local' '--with-jpeg-dir=/BinaryCache /apache_mod_php/apache_mod_php-66.3~5/Root/usr/local' '--with-png-dir=/BinaryCache /apache_mod_php/apache_mod_php-66.3~5/Root/usr/local' '--enable-gd-native-ttf' '--with-icu-dir=/usr' '--with-iodbc=/usr' '--with-ldap=/usr' '--with-ldap-sasl=/usr' '--with-libedit=/usr' '--enable-mbstring' '--enable-mbregex' '--with-mysql=mysqlnd' '--with-mysqli=mysqlnd' '--without-pear' '--with-pdo-mysql=mysqlnd' '--with-mysql-sock=/var/mysql/mysql.sock' '--with-readline=/usr' '--enable-shmop' '--with-snmp=/usr' '--enable-soap' '--enable-sockets' '--enable-sqlite-utf8' '--enable-suhosin' '--enable-sysvmsg' '--enable-sysvsem' '--enable-sysvshm' '--with-tidy' '--enable-wddx' '--with-xmlrpc' '--with-iconv-dir=/usr' '--with-xsl=/usr' '--enable-zend-multibyte' '--enable-zip' '--with-pcre-regex=/usr' '--with-pgsql=/usr' '--with-pdo-pgsql=/usr'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/private/etc/php.ini

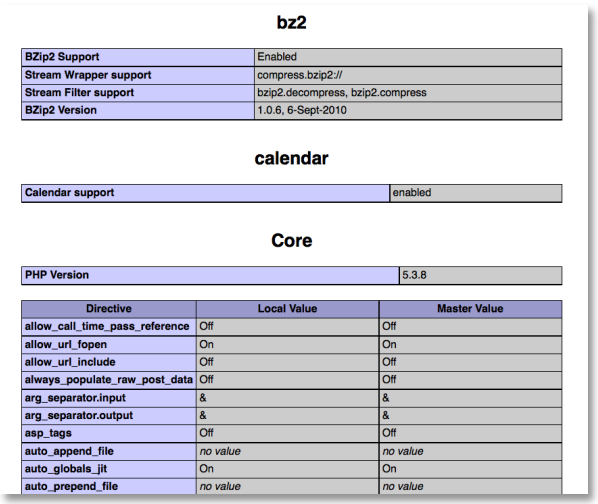


Configuration

apache2handler

Apache Version	Apache/2.2.21 (Unix) mod_ssl/2.2.21 OpenSSL/0.9.8r DAV/2 PHP/5.3.8 with Suhosin-Patch
Apache API Version	20051115
Server Administrator	you@example.com
Hostname:Port	edsac.local:0
User/Group	_www(70)/70
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	No
Server Root	/usr
Loaded Modules	core prefork http_core mod_so mod_authn_file mod_authn_dbm mod_authn_anon mod_authn_dbd mod_authn_default mod_authz_host mod_authz_groupfile mod_authz_user mod_authz_dbm mod_authz_owner mod_authz_default mod_auth_basic mod_auth_digest mod_cache mod_disk_cache mod_mem_cache mod_dbd mod_dumpio mod_ext_filter mod_include mod_filter mod_substitute mod_deflate mod_log_config mod_log_forensic mod_logio mod_env mod_mime_magic mod_cern_meta mod_expires mod_headers mod_ident mod_usertrack mod_setenvif mod_version mod_proxy mod_proxy_connect mod_proxy_ftp mod_proxy_http mod_proxy_ajp mod_proxy_balancer mod_ssl mod_mime mod_dav mod_status mod_autoindex mod_asis mod_info mod_cgi mod_dav_fs mod_vhost_alias mod_negotiation mod_dir mod_imagemap mod_actions mod_speling mod_userdir mod_alias mod_rewrite mod_bonjour2 mod_php5

Directive	Local Value	Master Value
engine	1	1
last_modified	0	0
xbithack	0	0



bz2

BZip2 Support	Enabled
Stream Wrapper support	compress.bzip2://
Stream Filter support	bzip2.decompress, bzip2.compress
BZip2 Version	1.0.6, 6-Sept-2010

calendar

Calendar support	enabled

Core

PHP Version	5.3.8

Directive	Local Value	Master Value
allow_call_time_pass_reference	Off	Off
allow_url_fopen	On	On
allow_url_include	Off	Off
always_populate_raw_post_data	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	*no value*	*no value*
auto_globals_jit	On	On
auto_prepend_file	*no value*	*no value*

Variabili PHP

- Le variabili seguono un meccanismo di dichiarazione implicita
  - Sono create al momento dell'uso
  - Il tipo dipende dal dato che viene assegnato alla variabile e può quindi variare nel flusso del programma (anche se non è una buona norma...)
  - I nomi delle variabili iniziano col carattere **$**
    - il **$** può essere intepretato come un operatore di riferimento alla tabella dei simboli in cui sono memorizzate le variabili indicizzate per nome
    - Un nome di variabile può contenere caratteri alfanumerici (il primo carattere deve essere una lettera) e l'underscore _
      - **`$i, $i1, $_input, ....`**
    - I nomi sono case-sensitive
      - **`$var`** e **`$Var`** sono variabili diverse
  - L'assegnazione copia il valore assegnato nella variabile

```
$v = 1; $v1 = $v; $v = 2;
echo "v = ".$v."<br />v1 = ".$v1;
```

v = 2
v1 = 1

Inizializzazione delle variabili

- Non è necessario inizializzare le variabili in PHP
  - Al primo uso assumono un valore di default per il tipo che meglio si adatta al contesto in cui sono usate
    - **`false`** per boolean, **`0`** per integer e float, la stringa vuota "", un array vuoto
    - Ad una variabile non inizializzata corrisponde il valore speciale NULL
    - In ogni caso non è una buona pratica lasciare variabili non inizializzate
  - Il PHP mette a disposizione costrutti per verificare se una variabile è stata inizializzata o se contiene valori "vuoti"
    - **`bool isset($var)`** produce **`false`** se **`$var`** non è stata inizializzata
    - **`bool empty($var)`** produce **`false`** anche se **`$var`** è inizializzata a **`false`**, **`""`** (la stringa vuota), **`"0"`** (la stringa col carattere 0), **`0`** o **`0.0`**, **`array()`**

```
if(!isset($_REQUEST['prodotto']))
  echo "FORM variable prodotto is not set <br />";
if(empty($_REQUEST['prodotto']))
  echo "FORM variable prodotto is empty <br />";

$v = 0;
if(!isset($v))
  echo "variable v is not set <br />";
if(empty($v))
  echo "variable v is empty<br />"
```

FORM variable prodotto is not set
FORM variable prodotto is empty
variable v is empty

Tipi di dato

- Il PHP supporta 8 tipi di dato primitivi - **`gettype($var)`**
  - Tipi scalari - **`is_scalar($var)`**
    - boolean – **`is_bool($var)`**
    - integer – **`is_int($var), is_numeric($var)`**
    - float – **`is_float($var), is_numeric($var)`**
    - string – **`is_string($var)`**
  - Tipi composti
    - array – **`is_array($var)`**
    - object – **`is_object($var)`**
  - Tipi speciali
    - resource – **`is_resource($var)`**
    - NULL – **`is_null($var)`**
    - callable

Le funzioni indicate permettono di verificare se una variabile contiene un valore di un dato tipo (producono **`true/false`**)

Variabili e tipi dato

- Il PHP non richiede di dichiarare eplicitamente il tipo di una variabile
  - Il tipo dipende dal valore assegnato
  - Il valore di una variabile viene convertito nel tipo appropriato (type juggling) in base al contesto in cui è usato senza variare la definizione della variabile stessa

```
Alla variabile <tt>$var</tt> viene assegnato il valore
della costante <tt>10</tt> <?php $var= 10; ?> e quindi
assume il tipo <tt><?php echo gettype($var); ?></tt> con
valore <tt><?php echo '$var = '.$var; ?></tt>.

Se si calcola l'espressione <tt>$var *= 1.5</tt>
<?php $var *= 1.5; ?> la variabile diventa di tipo
<tt><?php echo gettype($var); ?></tt> con valore <tt>
<?php echo '$var = '.$var; ?></tt>.

Se si esegue l'assegnazione <tt>$var+="44 cats"</tt>
<?php $var += "44 cats"; ?> la variabile è
ora di tipo <tt><?php echo gettype($var); ?></tt> con
valore <tt><?php echo '$var = '.$var; ?></tt>.
La costante stringa "44 cats" viene convertita nel tipo
della variabile (double) facendo una scansione da sinistra
fino a che si trovano cifre.
```

Alla variabile `$var` viene assegnato il valore della costante `10` e quindi assume il tipo `integer` con valore `$var = 10`.

Se si calcola l'espressione `$var *= 1.5` la variabile diventa di tipo `double` con valore `$var = 15`.

Se si esegue l'assegnazione `$var+="44 cats"` la variabile è ora di tipo `double` con valore `$var = 59`. La costante stringa "44 cats" viene convertita nel tipo della variabile (double) facendo una scansione da sinistra fino a che si trovano cifre.

Conversione di tipo e cast

- Se una variabile viene utilizzata nella definizione di una costante stringa con *double quoted* **`"..$var.."`**, ne viene utilizzato il valore
- Se si utilizza una stringa in un contesto che richiede un numero, la stringa viene scandita da sinistra a destra fino a che si trovano cifre
  - Questa operazione implicita è particolarmente utile per l'elaborazione dei dati che provengono da form HTML che in origine sono sempre in variabili di tipo string
- Si può utilizzare il cast **`(tipo)`** (es. **`(int), (float)`**) per forzare la conversione di un'espressione ad un tipo dati specifico

```
Se si esegue l'assegnazione <tt>$s = "2+$var"</tt>
<?php $s = "2+$var"; ?> la variabile <tt>$var</tt> si trova nel
contesto della definizione di una costante stringa con
<em>double quotes</em> e ne viene usato il valore scritto come
stringa di caratteri. Quindi il tipo di <tt>$s</tt> è
<tt><?php echo gettype($s); ?></tt> e il suo valore è
<tt><?php echo $s;?></tt>.

Se si utilizza la variabile <tt>$s</tt> in un contesto numerico
essa viene convertita con scansione delle cifre decimali da
sinistra a destra. Si può vedere come avviene la conversione
facendo un'assegnazione con cast <tt>$yav = (int)$s</tt>
<?php $yav = (int)$s;?> ottendo che <tt>$yav</tt> è di tipo
<tt><?php echo gettype($yav); ?></tt> con valore
<tt><?php echo $yav; ?></tt>. Come si può vedere la conversione
non implica fare una valutazione di un'eventuale espressione
contenuta nella stringa.
```

Se si esegue l'assegnazione `$s = "2+$var"` la variabile `$var` si trova nel contesto della definizione di una costante stringa con *double quotes* e ne viene usato il valore scritto come stringa di caratteri. Quindi il tipo di `$s` è `string` e il suo valore è `2+59`.

Se si utilizza la variabile `$s` in un contesto numerico essa viene convertita con scansione delle cifre decimali da sinistra a destra. Si può vedere come avviene la conversione facendo un'assegnazione con cast `$yav = (int)$s` ottendo che `$yav` è di tipo `integer` con valore `2`. Come si può vedere la conversione non implica fare una valutazione di un'eventuale espressione contenuta nella stringa.

Costanti stringa

- Le stringhe sono sequenze di caratteri di 1 byte
  - Il supporto nativo permette di usare solo codifiche di caratteri su byte e non c'è supporto diretto a Unicode
    - Questo non significa che non si possono mettere codifiche Unicode nelle stringhe, ma che le operazioni sulle stringhe non vedono come un solo carattere i codici multi-byte
  - Una stringa costante può essere specificata con 4 diversi metodi
    - double quoted **`"questa è una stringa"`** – le variabili che compaiono all'interno della definizione sono espanse nel loro valore e sono convertite le sequenze di escape (**`\n`**, **`\t`**,...)
    - single quoted **`'anche questa è una stringa'`** – le variabili non sono espanse e solo le sequenze di escape \' e \\ sono convertite nel carattere corrispondente
    - heredoc **`<<<LABEL ... LABEL;`** simile a double quoted
    - newdoc **`<<<'LABEL' ... LABEL;`** simile a single quoted

Costanti stringa: esempio

```
Una stringa è una sequenza di caratteri codificati su un byte.
Una stringa può essere definita usando i doppi apici
(<em>double quoted</em>) come in <tt>$str = "Questa è una
stringa";</tt><?php $str = "Questa è una stringa"; ?> che si
stampa in <tt><?php echo $str; ?></tt> ed è di lunghezza
<tt><?php echo strlen($str); ?></tt>. Il fatto che la codifica
è a 8 bit si vede cercando di stampare il carattere nella
posizione 7 che dovrebbe corrispondere a 'è' ma invece risulta
<tt><?php echo $str[7]; ?></tt> che è il primo carattere della
sua codifica Unicode che è su 2 byte.

Se si usa la definizione <em>single quoted</em> eventuali
variabili o sequenze di escape non vengono convertite nel loro
valore. Se si assegna <tt>$cats=44</tt><?php $cats=44 ?> e si
stampa la stringa <tt>'$cats cats\n'</tt> si ottiene
<tt><?php echo '$cats cats\n'; ?></tt> mentre con <tt>"$cats
cats\n"</tt> si stampa <tt><?php echo "$cats cats\n"; ?></tt>.

In alcuni casi può essere necessario delimitare le variabili (o
il loro nome) con parentesi graffe {} per separarle dal testo
che deve comparire subito dopo senza caratteri di separazione.
Ad esempio se si definiscono le variabili <tt>$fruit="apple";
$howmany="four";</tt><?php $fruit="apple"; $howmany="four";?> e
si stampa la stringa <tt>"I ate $howmany $fruits"</tt> si
ottiene <tt><?php echo "I ate $howmany $fruits";?></tt>, mentre
con <tt>"I ate $howmany ${fruit}s"</tt> si ha la stampa
corretta <tt><?php echo "I ate $howmany ${fruit}s";?></tt>.
```

Una stringa è una sequenza di caratteri codificati su un byte. Una stringa può essere definita usando i doppi apici (*double quoted*) come in `$str = "Questa è una stringa";` che si stampa in `Questa è una stringa` ed è di lunghezza `21`. Il fatto che la codifica è a 8 bit si vede cercando di stampare il carattere nella posizione 7 che dovrebbe corrispondere a 'è' ma invece risulta `?` che è il primo carattere della sua codifica Unicode che è su 2 byte.

Se si usa la definizione *single quoted* eventuali variabili o sequenze di escape non vengono convertite nel loro valore. Se si assegna `$cats=44` e si stampa la stringa `'$cats cats\n'` si ottiene `$cats cats\n` mentre con `"$cats cats\n"` si stampa `44 cats .`

In alcuni casi può essere necessario delimitare le variabili (o il loro nome) con parentesi graffe {} per separarle dal testo che deve comparire subito dopo senza caratteri di separazione. Ad esempio se si definiscono le variabili `$fruit="apple"; $howmany="four";` e si stampa la stringa `"I ate $howmany $fruits"` si ottiene `I ate four` , mentre con `"I ate $howmany ${fruit}s"` si ha la stampa corretta `I ate four apples.`

Operatori e funzioni su stringhe

- Due stringhe possono essere concatenate con l'operatore .
- Sono disponibili numerose funzioni utili per la manipolazione di stringhe e per la ricerca di sottostringhe con espressioni regolari (perl compatible RE)

```
<?php
  $s = '=?utf-8?Q?perch=C3=A9 =C3=A8 una prova (utf-8)?=';
  $parts = explode('?',$s);
?>

<head>
 <title>Quoted printable decoding</title>
 <meta http-equiv="Content-Type"
      content="text/html;charset=<?php echo $parts[1];?>" />
 </head>
 <body>
   <?php echo 'Input string: <em>'.$s.'</em>'; ?><br />
   The converted string is <em>
     <?php echo quoted_printable_decode($parts[3]); ?>
  </em><br />
</body>
</html>
```

```
<html>
<head>
  <title>Quoted printable decoding</title>
  <meta http-equiv="Content-Type"
       content="text/html;charset=utf-8" />
</head>
<body>
 Input string: <em>
  =?utf-8?Q?perch=C3=A9 =C3=A8 una prova (utf-8)?=
  </em><br />
 The converted string is <em>
    perché è una prova (utf-8)</em><br />
</body>
</html>
```

Input string: *=?utf-8?Q?perch=C3=A9 =C3=A8 una prova (utf-8)?=*
The converted string is *perché è una prova (utf-8)*

Array

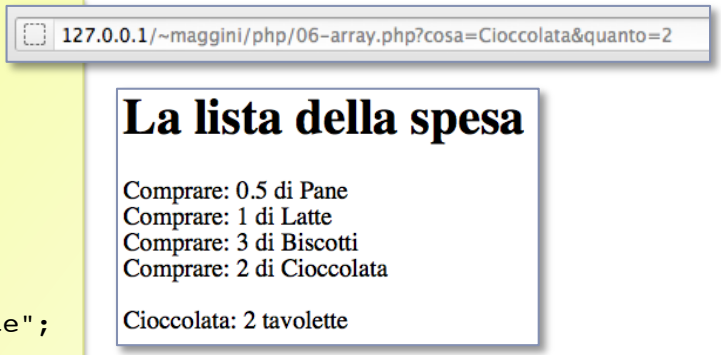
- In PHP gli array sono tabelle associative (map)
  - Un array crea un'associazione **`key => value`**
  - Gli elementi di un array possono essere di qualsiasi tipo; se sono a loro volta array si creano array multidimensionali (es. matrici)
  - La chiave può essere un integer o una string
  - Un array può essere creato col costrutto array() specificando come argomento (opzionale) la lista degli elementi (dalla versione 5.4 si può usare anche la definizione compatta con [...])

```
$Lista_Spesa = array("Pane"=>0.5, "Latte"=>1,"Biscotti"=>3);

if(isset($_REQUEST['cosa'])&&isset($_REQUEST['quanto']))
  $Lista_Spesa[$_REQUEST['cosa']] += $_REQUEST['quanto'];

foreach ($Lista_Spesa as $cosa => $quanto)
    print "Comprare: $quanto di $cosa <br />\n";

if(isset($Lista_Spesa['Cioccolata']))
 print "<br />Cioccolata: {$Lista_Spesa['Cioccolata']} tavolette";
else
 print "<br />Cioccolata: non comprare!!";
```



Array: indici e funzioni

- Gli indici (chiavi) di un array possono essere numeri interi o stringhe
  - Possono coesistere insieme
  - Se si aggiungono valori ad un array senza specificare un valore per la chiave, viene utilizzato l'intero successivo corrispondente all'ultima chiave numerica inserita (eventualmente 0)
  - Si accede ad un elemento di un array con l'indice specificato in [..]
  - Per rimuovere una coppia **`key=>value`** da un array si può utilizzare il costrutto **`unset($arr['key'])`**
- Esiste un'ampia libreria di funzioni e costrutti per manipolare array (**`count, sort, array_diff, array_fill..`**)

```
$impegni[] = array('month'=>'Jan','day'=>10,
                   'hour'=>12,'desc'=>'Lecture');
$impegni[] = array('month'=>'Feb','day'=>5,'hour'=>14,
                   'desc'=>'Conference');
$impegni[] = array('month'=>'Mar','day'=>22,'desc'=>'Birthday');

for($i=0;$i<count($impegni);$i++) {
  print "$i. {$impegni[$i]['month']} {$impegni[$i]['day']}";
  if(isset($impegni[$i]['hour']))
    print " at {$impegni[$i]['hour']}";
  print ": <em>{$impegni[$i]['desc']}</em><br />";
}
```

La lista degli impegni

0. Jan 10 at 12: *Lecture*
1. Feb 5 at 14: *Conference*
2. Mar 22: *Birthday*

Variabili predefinite e "superglobali"

- Il PHP definisce un ampio numero di variabili predefinite
  - Permettono di riferirsi a dati esterni (es. le variabili provenienti da un form HTML), all'ambiente interno PHP, ai messaggi di errore (es. **`$php_errormsg`**)
  - Le variabili superglobali sono array associativi accessibili da ogni contesto in uno script PHP
    - **`$_SERVER[]`**: informazioni relative al server su cui è eseguito lo script
    - **`$_ENV[]`**: variabili di ambiente di esecuzione dell'interprete PHP
    - **`$_COOKIE[]`**: informazioni sui cookies nella richiesta HTTP
    - **`$_GET[]`**: informazioni relative ad una richiesta con metodo GET
    - **`$_POST[]`**: informazioni relative ad una richiesta con metodo POST
    - **`$_REQUEST[]`**: è l'insieme di **`$_COOKIE`**, **`$_GET`** e **`$_POST`**
    - **`$_SESSION[]`**: contiene le variabili registrate per la sessione corrente
    - **`$_FILES[]`**: contiene gli oggetti passati con il metodo POST

La variabile $_SERVER

- La variabile **`$_SERVER`** permette di ottenere informazioni su
  - Le intestazioni della richiesta HTTP (**`HTTP_USER_AGENT`, `HTTP_ACCEPT_LANGUAGE`, `HTTP_HOST`**,..)
  - Sulla connessione col client (**`REMOTE_ADDR`, `REMOTE_PORT`**)
  - Sul server (**`SERVER_SOFTWARE`, `DOCUMENT_ROOT`**,...)

```
$vars = array('SERVER'=>$_SERVER,
      'ENV'=>$_ENV,
      'COOKIE'=>$_COOKIE,
      'GET'=>$_GET,
      'POST'=>$_POST,
      'REQUEST'=>$_REQUEST,
      'FILES'=>$_FILES,
      'SESSION'=>$_SESSION);

foreach($vars as $vartype=>$var)
  echo "<a href=\"#_${vartype}\">\$_${vartype}</a><br />";

foreach($vars as $vartype=>$var) {
  echo "<h3>Elenco \$_${vartype}</h3>";
  echo "<a name=_${vartype}></a>";

  foreach($var as $element=>$val)
    echo "<b>$element</b>: $val<br />";
}
```

$_SERVER
$_ENV
$_COOKIE
$_GET
$_POST
$_REQUEST
$_FILES
$_SESSION

Elenco $_SERVER

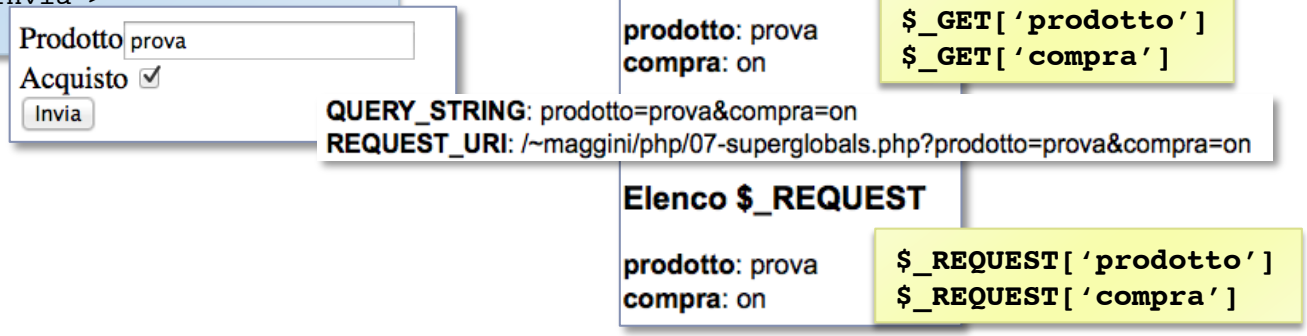
HTTP_HOST: 127.0.0.1
HTTP_USER_AGENT: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:11.0) Gecko/20100101 Firefox/11.0
HTTP_ACCEPT: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE: it,en-us;q=0.7,en;q=0.3
HTTP_ACCEPT_ENCODING: gzip, deflate
HTTP_CONNECTION: keep-alive
HTTP_REFERER: http://127.0.0.1/~maggini/php/
HTTP_COOKIE: PHPSESSID=aq2klvf3qsg7baqsh9ogth94k2
PATH: /usr/bin:/bin:/usr/sbin:/sbin
SERVER_SIGNATURE:
SERVER_SOFTWARE: Apache/2.2.21 (Unix) mod_ssl/2.2.21 OpenSSL/0.9.8r DAV/2 PHP/5.3.8 with Suhosin-Patch
SERVER_NAME: 127.0.0.1
SERVER_ADDR: 127.0.0.1
SERVER_PORT: 80
REMOTE_ADDR: 127.0.0.1
DOCUMENT_ROOT: /Library/WebServer/Documents
SERVER_ADMIN: you@example.com
SCRIPT_FILENAME: /Users/maggini/Sites/php/07-superglobals.php
REMOTE_PORT: 52981
GATEWAY_INTERFACE: CGI/1.1
SERVER_PROTOCOL: HTTP/1.1
REQUEST_METHOD: GET
QUERY_STRING:
REQUEST_URI: /~maggini/php/07-superglobals.php
SCRIPT_NAME: /~maggini/php/07-superglobals.php
PHP_SELF: /~maggini/php/07-superglobals.php
REQUEST_TIME: 1334241634

Accesso ai dati inviati con FORM HTML

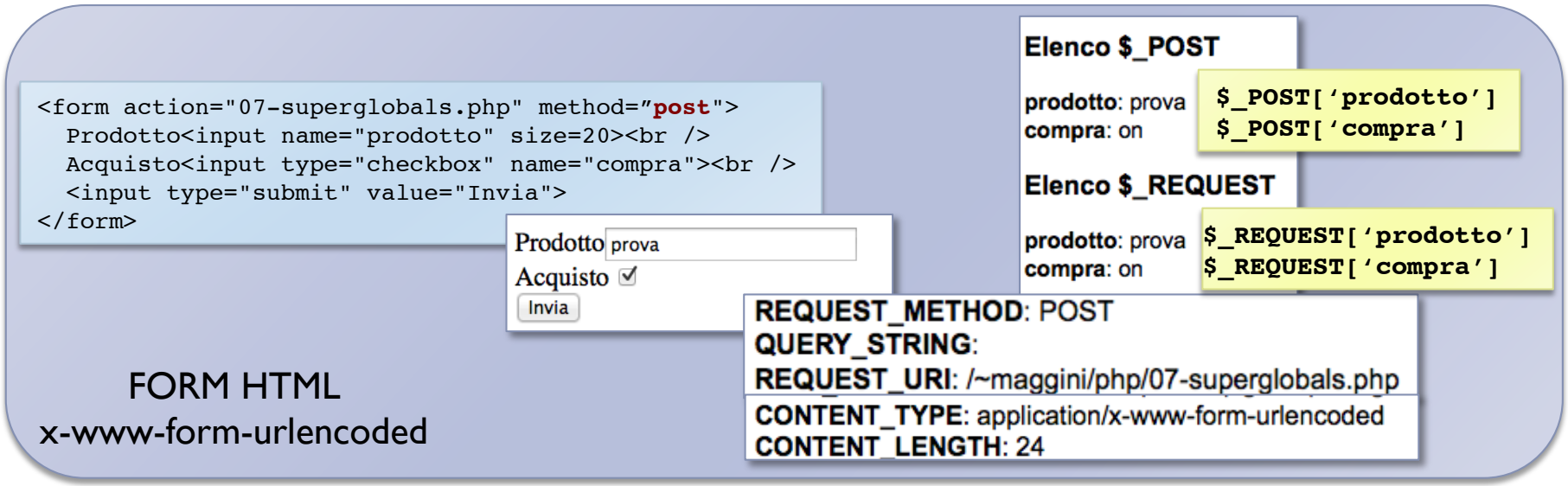
- Gli array **`$_GET`**, **`$_POST`** e **`$_REQUEST`** permettono di accedere direttamente ai valori delle variabili inviati con un form
  - L'indice è la stringa corrispondente al nome della variabile associata all'elemento di input nel form
  - Il valore è una stringa corrispondente al valore inserito o selezionato
  - L'array in cui è inserita la variabile dipende dal metodo indicato nel form (GET o POST)
  - La variabile è sempre accessibile nell'array **`$_REQUEST`** per cui facendo riferiemento ad esso si rende il codice PHP indipendente dalla scelta fatta nel form

```
<form action="07-superglobals.php" method="get">
  Prodotto<input name="prodotto" size=20><br />
  Acquisto<input type="checkbox" name="compra"><br />
  <input type="submit" value="Invia">
</form>
```

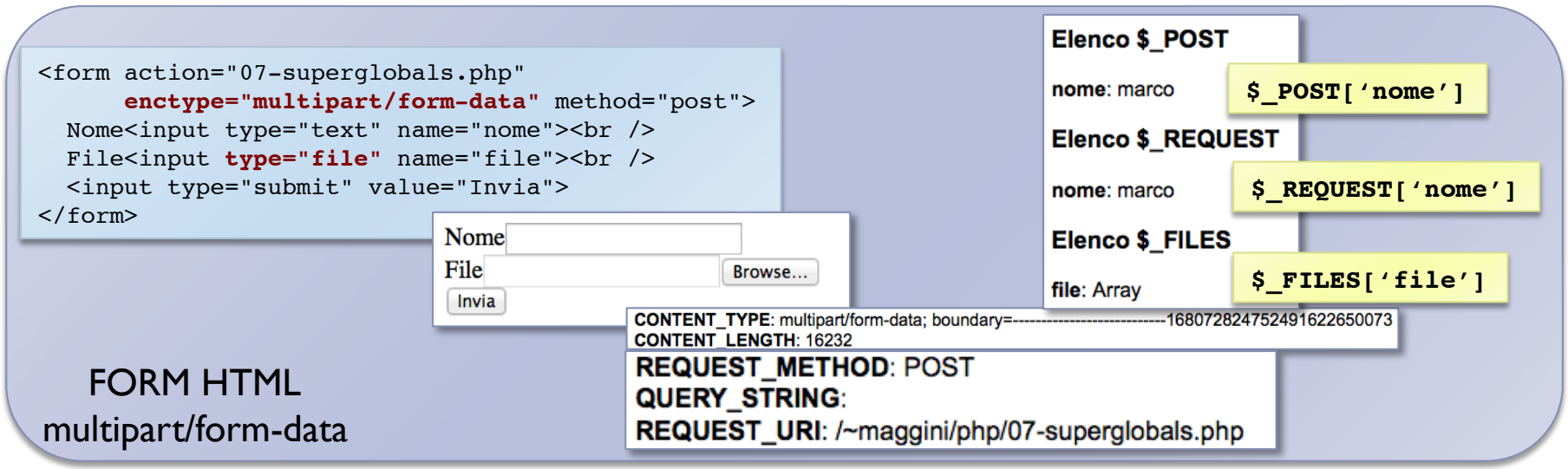
FORM HTML



Invio dati con POST




```
<form action="07-superglobals.php" method="post">
  Prodotto<input name="prodotto" size=20><br />
  Acquisto<input type="checkbox" name="compra"><br />
  <input type="submit" value="Invia">
</form>
```




```
<form action="07-superglobals.php"
      enctype="multipart/form-data" method="post">
  Nome<input type="text" name="nome"><br />
  File<input type="file" name="file"><br />
  <input type="submit" value="Invia">
</form>
```

Upload di file

- La variabile superglobale **`$_FILES`** contiene informazioni su file inviati con form HTML attraverso l'input di tipo **`file`**
  - Ad ogni file inviato col form è associato un array **`$_FILES['nome']`**
  - L'array contiene il nome, il tipo, la dimensione, il nome del file temporaneo sul server e il codice di errore (0 per nessun errore)
  - Il file temporaneo può essere copiato nella destinazione finale con la funzione **`move_uploaded_file($tmp_file,$destination)`**

```
Name: <?php echo $_FILES['file']['name']; ?><br />
Type: <?php echo $_FILES['file']['type']; ?><br />
Size: <?php echo $_FILES['file']['size']; ?><br />
Tmp_name: <?php echo $_FILES['file']['tmp_name']; ?><br />
Error: <?php echo $_FILES['file']['error']; ?><br />

<?php
if($_FILES['file']['error']==0) {
  $uploaddir = dirname($_SERVER['SCRIPT_FILENAME'])
               ."/upload/";
  $filename = $uploaddir.basename($_FILES['file']['name']);

  if(move_uploaded_file($_FILES['file']['tmp_name'], $filename))
    echo "File uploaded successfully<br />";
  else
    echo "Error: File not uploaded!<br />";
} ?>
```

Uploaded file info

Name: *Untitled.tiff*
Type: *image/tiff*
Size: *15910*
Tmp_name: */private/var/tmp/phpL3FFnY*
Error: *0*
File uploaded successfully

Campo di visibilità delle variabili

- Il campo di visibilità di una variabile (scope) è il contesto in cui è definita
  - comprende anche file inclusi sia con la direttiva **`include`** che **`require`**
  - All'interno delle funzioni è definito un campo di visibilità locale
    - ogni variabile usata all'interno di una funzione è per default limitata all'interno del corpo della funzione
    - le variabili definite all'esterno non sono accessibili nella funzione a meno di non dichiararle **`global`** o di usare l'array predefinito **`$GLOBALS`**

```
$pi = 3.14;

function circle($r) {
  return 2*$pi*$r;
}

function circleG($r) {
  global $pi;
  return 2*$pi*$r;
}

function circleGA($r) {
  return 2*$GLOBALS['pi']*$r;
}
```

Global variables in function scope

$pi is local: 0 `circle(1)`
$pi is declared global: 6.28 `circleG(1)`
$pi is from $GLOBALS array: 6.28 `circleGA(1)`

Variable variables

- Il PHP permette di definire variabili che contengono il nome con cui riferire altre variabili
  - Una "variabile variabile" prende il valore di una variabile e lo tratta come nome di un'altra variabile
    - Se `$varname = "var"` allora `$$varname` è equivalente a `$var`
  - Nel caso di array occorre specificare con parentesi a chi si riferisce l'indice
    - `$$arr[1]` deve essere scritto come `${$arr[1]}` o `$${arr}[1]`

```
$users = array('admin'=>'12345',
               'marco'=>'6789',
               'pippo'=>'pluto');
foreach($_REQUEST as $var=>$val)
  $$var = $val;

if(!empty($users[$user])&&($users[$user]==$pass)) {
  echo "Welcome ${user}!<br />";
  if($user=='admin')
    echo 'You have administration privileges!';
} else {
 echo "Login failed: you have guest privileges";
}
```

Form con selezione multipla

- L'elemento di input **<select multiple="yes">** permette all'utente di selezionare più opzioni
  - Il problema è che tutti i valori selezionati sono passati con lo stesso nome di variabile
  - La soluzione è usare la funzione "*array da elemento di form*"

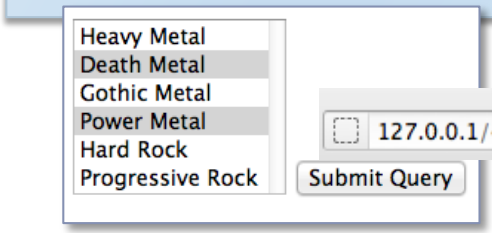
```
<select multiple="yes" name="var[]">
```

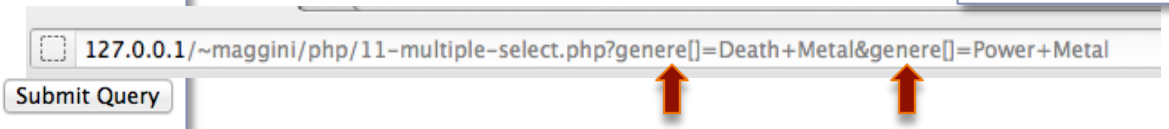
  - Il PHP tratta l'elemento come array e aggiunge un elemento per ogni valore selezionato con indici 0,1,...

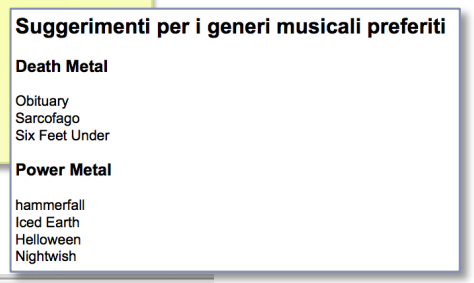
```
<form action="11-multiple-select.php">
<select name="genere[]" multiple="true" size=6>
 <option>Heavy Metal</option>
 <option>Death Metal</option>
 <option>Gothic Metal</option>
 <option>Power Metal</option>
 <option>Hard Rock</option>
 <option>Progressive Rock</option>
</select>
<input type=submit>
</form>
```

```
foreach($_REQUEST['genere'] as $genere) {
  echo "<h3>$genere</h3>";

$sugg = $gruppi[$genere];
 foreach($sugg as $gruppo)
   echo "$gruppo <br />";
```







Operatori 1

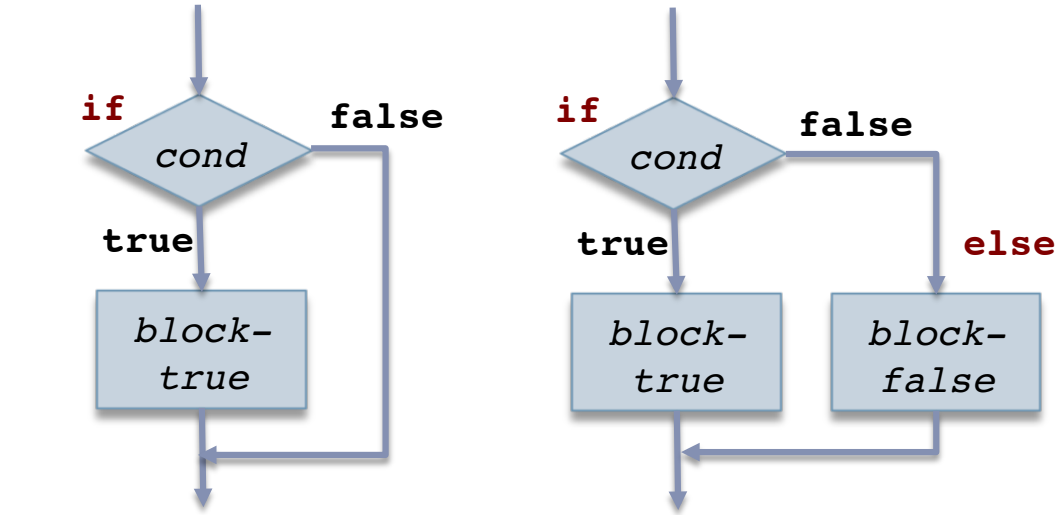
- Operatori Aritmetici
  - `$a+$b`, `$a-$b`, `$a*$b`, `$a/$b`, `$a%$b` (modulo), `-$a`, `$a.$b` (concatenazione di stringhe)
- Operatori di assegnazione
  - `$a=$b`, `$a+=$b`, `$a-=$b`, `$a/=$b`, `$a%=$b`, `$a.=$b` (`$a=$a.$b`)
- Operatori di autoincremento/autodecremento
  - pre `++$a`, `--$a` ; post `$a++`, `$a--`
- Operatori di comparazione
  - Producono un valore boolean (true/false)
  - `$a==$b` (uguale come valore), `$a===$b` (uguale come valore e tipo), `$a!=$b` e `$a<>$b` (diverso), `$a!==$b` (non identico), `$a>$b`, `$a<$b`, `$a>=$b`, `$a<=$b`
  - Gli operatori >, < con le stringhe tengono conto dell'ordinamento alfabetico (`"cane">"canile"` è `false` e `"cane"<"canile"` è `true`)

Operatori 2

- Operatori logici
  - `$a and $b` o `$a && $b` (AND logico) , `$a or $b` o `$a||$b` (OR logico), `$a xor $b` (XOR logico), `!$a` (NOT logico)
- Operatori su array
  - `$a + $b` (unione), `$a == $b` (`$a` e `$b` contengono le stesse coppie chiave/valore), `$a===$b` (`$a e $b` contengono le stesse coppie chiave/valore nello stesso ordine e degli stessi tipi), `$a != $b` o `$a <> $b` (disuguaglianza), `$a !== $b` (non identità)
- Operatori su bit
  - `$a&$b` (AND bit a bit), `$a|$b` (OR bit a bit), `$a^$b` (XOR bit a bit), `~$a` (NTO bit a bit), `$a<<$b` (SHIFT a sinistra di `$b` bit), `$a>>$b` (SHIFT a destra di `$b` bit)

Strutture di controllo: if

- **`if (`*`cond`*`) {`*`block-true`*`}`** [**`else {`***`block-false`***`}`**]
  - *`cond`* è un'espressione che viene valutata come boolean
    - per integer (float) è **`true`** se il valore è diverso da **`0`** (**`0.0`**)
    - per string è **`true`** se la stringa non è la stringa vuota `""` o `"0"`
    - per array è **`true`** se l'array contiene almeno un elemento

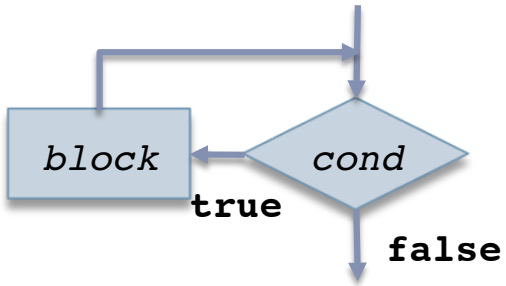



```
if(cond) {block-true}
```

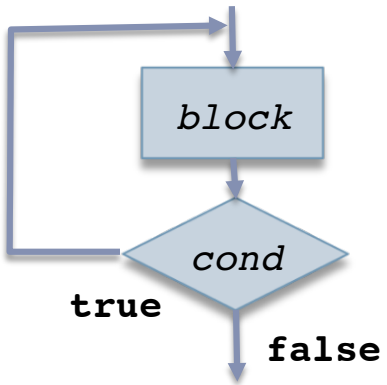
```
if(cond) {block-true}
  else {block-false}
```

Strutture di controllo: cicli

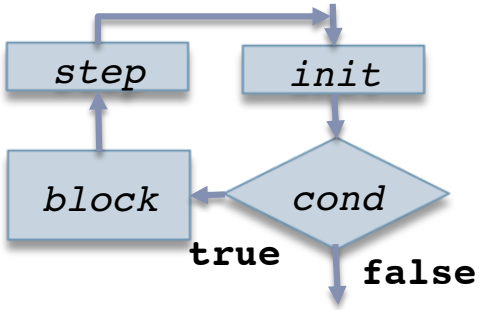
- **`while (`***`cond`***`) {`***`block`***`}`**
  - La condizione è verificata all'inizio del ciclo per cui il blocco di istruzioni può anche non essere eseguito mai
- **`do {`***`block`***`} while (`***`cond`***`)`**
  - La condizione è verificata alla fine del ciclo e il blocco di istruzioni è eseguito almeno una volta
- **`for(`***`init;cond;step`***`) {`***`block`***`}`**
  - Le espressioni init, cond e step possono anche essere vuote o essere composte di più espressioni separate dal carattere **`,`**



`while(`*`cond`***`) {`***`block`***`}`**



`do {`*`block`***`} while(`***`cond`***`)`**



`for(`*`init;cond;step`***`) {`***`block`***`}`**

Strutture di controllo: foreach, break

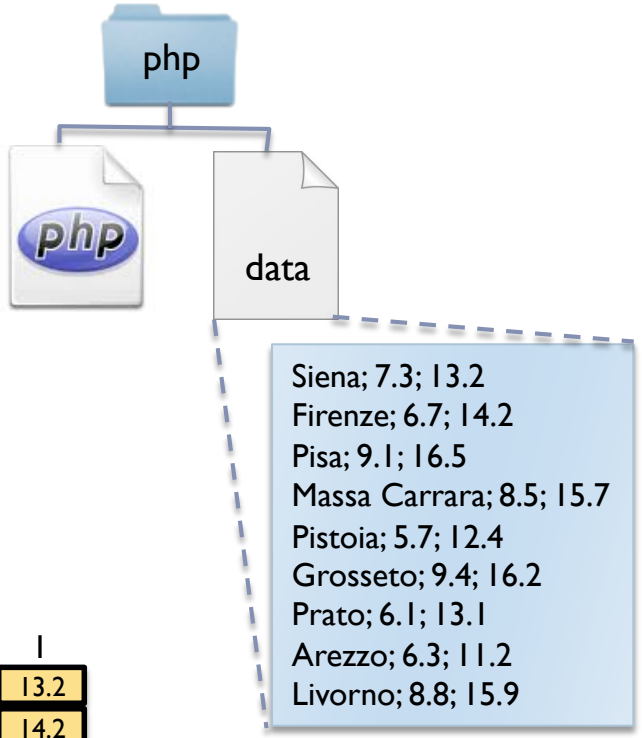
- `foreach($arr as $val){block}`

  `foreach($arr as $key=>$val){block}`
  - funziona solo con array o oggetti
  - effettua la scansione dell'array assegnando ad ogni iterazione l'elemento corrente alla variabile `$val` o alla coppia (`$key`,`$val`)
  - se si vuole modificare `$val` nel ciclo occorre assegnarlo come riferimento usando l'operatore `&$val`

- In un ciclo o struttura di controllo è possibile usare
  - `break;` per uscire dal blocco in cui ci si trova
  - `continue;` per saltare all'iterazione successiva del ciclo
  - in entrambi i casi si può specificare un intero *n* per indicare il livello di ciclo a cui saltare nel caso di più cicli annidati

Strutture di controllo: esempio 1

```
<?php
  $file = @fopen('data','r');
  if(!$file) { ?>
    <h2>Error: cannot open data file</h2>
    </body>
    </html> <?php
    die();
  }
  while($line=fgets($file)) {
    $data = explode(';',$line);
    if(count($data)!=3)
      continue;
    $temp[$data[0]] = array($data[1],$data[2]);
  }
  ksort($temp);
  fclose($file);
?>
```



$temp[][]

		0	1
Siena	=>	7.3	13.2
Firenze	=>	6.7	14.2
Pisa	=>	9.1	16.5
Massa Carrara	=>	8.5	15.7
Pistoia	=>	5.7	12.4
Grosseto	=>	9.4	16.2
Prato	=>	6,1	13.1
Arezzo	=>	6.3	11.2
Livorno	=>	8.8	15.9

Strutture di controllo: esempio 1

```
<h2>Temperature in Toscana</h2>
<table>
<tr><th style="text-align: left">Citt&agrave;</th>
<th style="text-align: center">Tmin</th>
<th style="text-align: center">Tmax</th></tr>
<?php
  $minTmin = 100;
  $maxTmax = -100;

  foreach($temp as $citta=>$minmax){
    list($Tmin,$Tmax)=$minmax;
    echo "<tr><td>$citta</td><td>$Tmin</td><td>$Tmax</td></tr>";

    if($Tmin<$minTmin) {
      $minTmin = $Tmin;
      $minCitta = $citta;
    }
    if($Tmax>$maxTmax) {
     $maxTmax = $Tmax;
     $maxCitta = $citta;
    }
  }
?>
</table>
<p>La citt&agrave; pi&ugrave; fredda &egrave;
   <?php echo "$minCitta ($minTmin)";?><br />
La citt&agrave; pi&ugrave; calda &egrave;
  <?php echo "$maxCitta ($maxTmax)";?></p>
```

Temperature in Toscana

Città	Tmin	Tmax
Arezzo	6.3	11.2
Firenze	6.7	14.2
Grosseto	9.4	16.2
Livorno	8.8	15.9
Massa Carrara	8.5	15.7
Pisa	9.1	16.5
Pistoia	5.7	12.4
Prato	6.1	13.1
Siena	7.3	13.2

La città più fredda è Pistoia (5.7)
La città più calda è Pisa (16.5)

Strutture di controllo: switch

- **`switch(`*`$v`*`) {case` *`val`*`:` .... `default:` .... `}`**
  - Funziona anche se *`$v`* è una stringa (i casi *`val`* sono stringhe)
  - L'esecuzione passa all'istruzione corrispondente al **`case`** che fa match con il valore di *`$v`* e poi le istruzioni sono eseguite in sequenza
  - Se si vuole che ogni case sia un blocco in alternativa, occorre usare **`break`** come ultima istruzione per interrompere l'esecuzione in sequenza

```
switch($l) {
case 'it':
  $msg="Preferisci l'Italiano con valore ";
  break;
case 'en-us':
  $msg="You prefer English American with score ";
  break;
case 'en':
  $msg="You prefer English with score ";
  break;
default:
  $msg="$l is an unknown language specification";
}
```

Parsing Browser's Accept-Language

Input string: *it,en-us;q=0.8,en;q=0.5,fr;q=0.3*

Preferisci l'Italiano con valore 1
You prefer English American with score 0.8
You prefer English with score 0.5
fr is an unknown language specification 0.3

inclusione di file

- **`include`**, **`require`**, **`include_once`**, **`require_once`**
  - Questi costrutti permettono di includere il contenuto di un file
  - Quando si inizia l'inclusione l'interprete esce dalla modalità di esecuzione e codice PHP nel file incluso deve essere fra **`<?php ?>`**
  - Se il nome del file incluso non contiene un *path* viene ricercato nella directory dello script corrente e nelle directory specificate nella configurazione PHP con l'opzione **`include_path`**
  - Il file eredita il campo di visibilità delle variabili del contesto in cui si trova la linea **`include`**/**`require`**
  - Variabili e funzioni definite nel file incluso hanno visibilità che dipende dal contesto in cui è incluso
  - La differenza fra **`include`** e **`require`** è che se non si trova il file **`include`** genera un *warning* mentre **`require`** un *errore fatale*
  - Le versioni con **`_once`** includono il file una volta sola, anche se è argomento di più costrutti che vengono eseguiti

Include: esempio 1

```php
<?php
  error_reporting(E_ALL);
  ini_set("display_errors", 1);
  require '14-functions.inc';
?>
<html>
<head>
<title>Esempio di include</title>
</head>
<body>
<?php
  $content = @$_REQUEST['content'];
  if(empty($content)) { ?>
      <h2>Error: no content selected</h2>
      </body></html>
     <?php die(); }
  ?>
<h2> <?php echo include_title($content); ?></h2>
<div style="text-align: justify;
        border: 5px solid DodgerBlue;padding:5px">
<?php include_content($content); ?></div>
<em>Last modified: </em><?php echo @$last_modified ?>
</body>
</html>
```

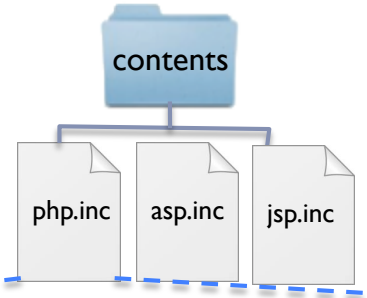
```php
<?php
$contentdir = "./14-contents/";
$index = array(
  "php"=>array("titolo"=>"PHP language", "file"=>"php"),
  "asp"=>array("titolo"=>"ASP language", "file"=>"asp"),
  "jsp"=>array("titolo"=>"JSP language", "file"=> "jsp"));

function include_title($id) {
  global $index;
  if(!empty($index[$id]['titolo']))
    return $index[$id]['titolo'];
  else
    return "Contenuto non disponibile";
}

function include_content($id) {
  global $index, $contentdir, $last_modified;
  if(!empty($index[$id]['file']))
    include "${contentdir}{$index[$id]['file']}.inc";
  else
    echo "Contenuto non disponibile";
} ?>
```

14-functions.inc

include: esempio 2



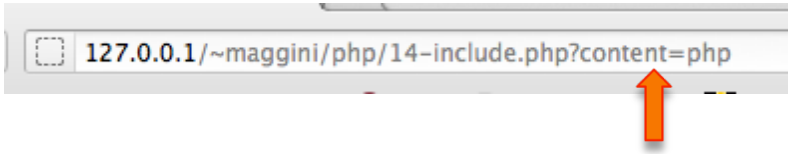

```
<?php $last_modified = "Apr 17, 2012 12:45 am"; ?>

<p>PHP (acronimo ricorsivo di "PHP: Hypertext
Preprocessor", preprocessore di ipertesti;
originariamente acronimo di "Personal Home Page"[1]) è
un linguaggio di scripting interpretato,
con licenza open source e libera (ma incompatibile con
la GPL), originariamente concepito per la
programmazione Web ovvero la realizzazione di pagine web
dinamiche.</p>

<p>Attualmente è utilizzato principalmente per
sviluppare applicazioni web lato server
ma può essere usato anche per scrivere script a riga di
comando o applicazioni standalone
con interfaccia grafica.</p>

<p>L'elaborazione di codice PHP sul server produce
codice HTML da inviare al browser dell'utente che ne
fa richiesta. Il vantaggio dell'uso di PHP e degli altri
linguaggi Web come ASP e .NET rispetto al
classico HTML derivano dalle differenze profonde che
sussistono tra Web dinamico e Web statico.</p>
```

./contents/php.inc



PHP language

PHP (acronimo ricorsivo di "PHP: Hypertext Preprocessor", preprocessore di ipertesti; originariamente acronimo di "Personal Home Page"[1]) è un linguaggio di scripting interpretato, con licenza open source e libera (ma incompatibile con la GPL), originariamente concepito per la programmazione Web ovvero la realizzazione di pagine web dinamiche.

Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server ma può essere usato anche per scrivere script a riga di comando o applicazioni standalone con interfaccia grafica.

L'elaborazione di codice PHP sul server produce codice HTML da inviare al browser dell'utente che ne fa richiesta. Il vantaggio dell'uso di PHP e degli altri linguaggi Web come ASP e .NET rispetto al classico HTML derivano dalle differenze profonde che sussistono tra Web dinamico e Web statico.

Last modified: Apr 17, 2012 12:45 am

Funzioni

- Il PHP permette di definire funzioni specificando
  - il nome della funzione
  - la lista dei parametri
    - è supportato il passaggio per valore (default), per riferimento (usando l'operatore &), i valori di default e un numero variabile di parametri
    - col passaggio per valore l'argomento è copiato e quindi ogni modifica non si riflette nella variabile passata come parametro
  - il codice eseguito dalla funzione
    - La funzione può produrre un valore con l'istruzione **`return`** *`expr`*`;`

```
function mymin($arr) {
  if(!is_array($arr))
    return $arr;

  reset($arr);
  $minval = current($arr);
  foreach($arr as $value) {
    if($value<$minval)
      $minval = $value;
    }
  return $minval;
}
```

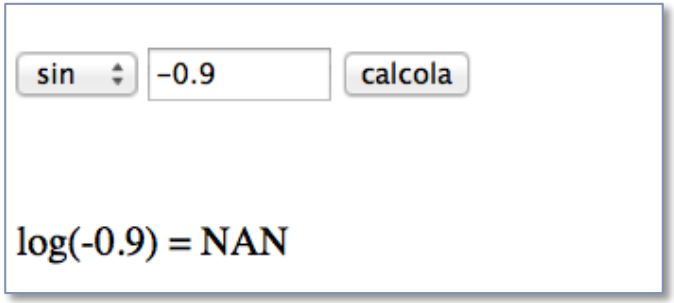
```
$c = array("a"=>1,"b"=>2,"c"=>3,"d"=>4,"e"=>5);
$m = mymin($c);
```

Il minimo del vettore ["a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5] è 1

Variabili funzione

- Sono supportate le variabili funzione
  - Una variabile `$f` contiene il nome di una funzione come stringa
  - La chiamata della funzione si può ottenere come `$f(`*`parametri`*`)`

```
<?php
  $functs = array("sin","cos","tan","exp","log");
  $f = $_REQUEST['functID'];
  $x = $_REQUEST['x'];
?>
<form action="15-functiondemo.php" >
  <select name="functID" >
  <?php
    foreach($functs as $fname)
      echo "<option>${fname}</option>";
  ?>
  </select>
  <input type="text" size="8" name="x" />
  <input type="submit" value="calcola"/>
</form><br /><br />
<?php
  if(!empty($f) && !empty($x)) {
    $result = $f($x);
    echo "${f}(${x}) = ${result}";
  }
?>
```



Classi e oggetti (cenni)

- E' supportata la definizione di classi
  - Una classe descrive un insieme di oggetti caratterizzati da proprietà (variabili e costanti) e metodi (funzioni)
  - Un oggetto di una classe viene creato con l'operatore **new** che produce un handle (riferimento) all'oggetto creato
  - Dato un handle di un oggetto si possono eseguire i metodi su tale oggetto o accedere alle sue proprietà (se public)

```
class lamp {
  public $power;
  private $state="off";

  function __construct($pwr)
     {$this->power=$pwr;}
  public function turnon(
    {$this->state="on";}
  public function turnoff()
   {$this->state="off";}
  public function getstate()
   {return $this->state;}
}
```

proprietà

metodi

```
$lamp75 = new lamp(75);
echo $lamp75->getstate()."<br />";
$lamp75->turnon();
echo $lamp75->getstate()."<br />";
echo $lamp75->power."<br />";
echo $lamp75->state."<br />";
```

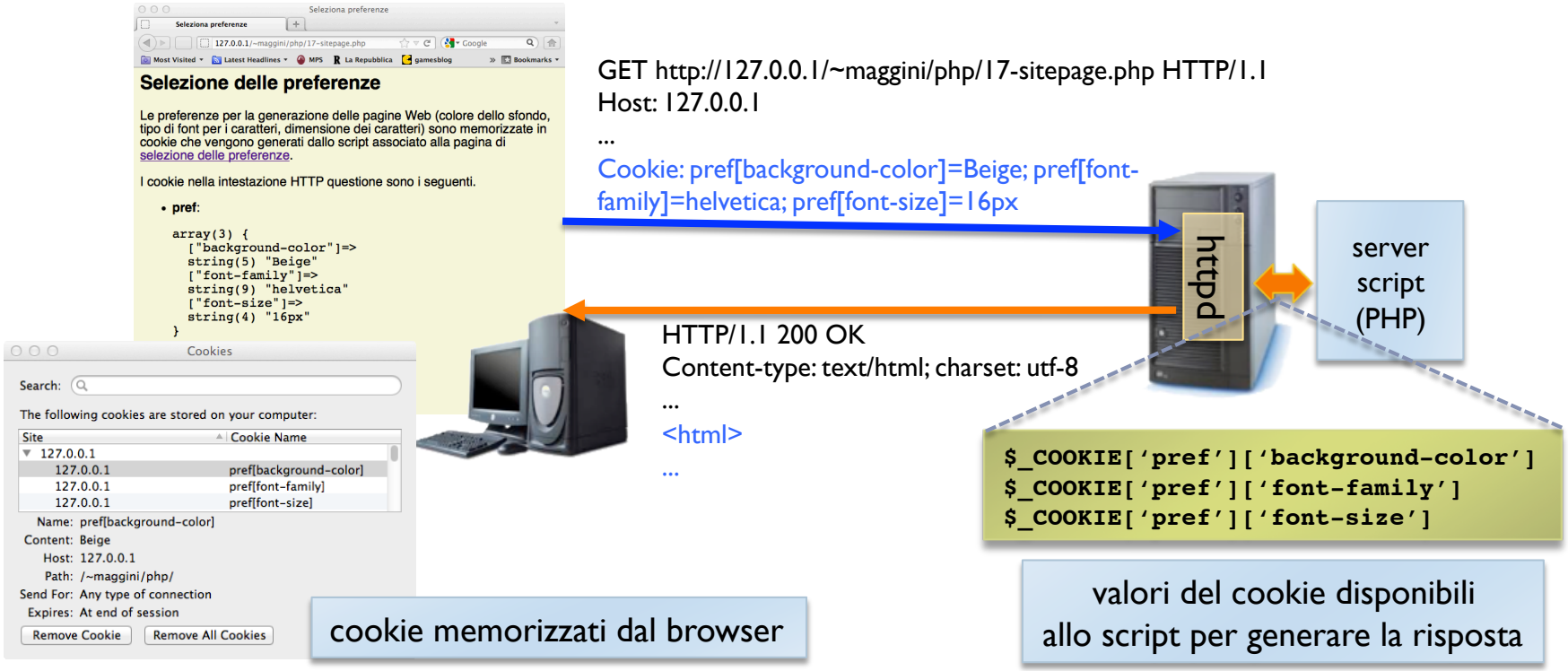
```
off
on
75
Fatal error: Cannot access private property
```

Cookies

- I cookie sono un meccanismo per memorizzare dati su un browser da parte di un server
  - Sono un supporto per creare uno stato nelle transazioni HTTP, ovvero permettono di creare una storia delle azioni effettuate da un browser su un certo server Web
  - Devono essere abilitati e supportati dal browser
  - Un cookie è una variabile con associati dei dati che viene inizializzata e manipolata dal server Web e memorizzata sul browser in seguito a una transazione HTTP
    - Alla prima connessione il server Web invia la variabile nell'header HTTP creando il cookie sul browser (se i cookie sono abilitati)
    - Il browser memorizza la variabile in una tabella interna con il riferimento al server Web che l'ha inviata
    - Tutte le volte che il browser contatta il server Web invia nell'header della richiesta il cookie con la variabile e il suo valore attuale
    - Il server Web accede alla variabile ed eventualmente invia un aggiornamento nella risposta HTTP

Uso dei cookie

- I cookie sono dati e non codice eseguito
  - Non possono essere utilizzati per creare virus o malware
  - Possono essere utilizzati per tracciare le azioni degli utenti con possibili minacce alla privacy (tracking cookies - third-party tracking cookies)
  - L'ultima specifica è nell'RFC RFC 6265 (2011) che aggiorna i precedenti



Uso dei cookie

- Gestione di sessioni
  - Il cookie può essere utilizzato per memorizzare la storia di più richieste allo stesso sito fra loro correlate
  - I cookie sono stati introdotti (1994) come meccanismo per poter gestire un carrello della spesa (shopping cart)
  - Con le soluzioni attuali il cookie non contiene direttamente tutte le informazioni di stato ma solo un ID univoco di sessione che identifica un'aria di memoria sul server ad ogni richiesta
- Personalizzazione
  - Un cookie può essere utilizzato per memorizzare delle preferenze dell'utente per l'accesso ad un dato sito
- Tracking
  - Possono essere utilizzati per tracciare le abitudini di navigazione degli utenti
  - un cookie contenente un ID univoco viene inviato all'utente alla prima connessione al sito e da quel momento è possibile creare un log delle pagine visitate
  - I third-party cookies sono cookie generati da siti diversi da quello visualizzato nella barra del browser per il fatto che la pagina caricata contiene un elemento proveniente dal sito del cookie (es. un banner pubblicitario)
    - Il third-party può di fatto tracciare tutti i siti visitati dall'utente che contengono i propri banner
    - Si può configurare il browser in modo che non li accetti

Cookie e HTTP

- I cookie sono inviati nelle intestazioni HTTP
  - Un server può creare o modificare un cookie su un browser inviando l'intestazione **`Set-cookie:`** nella quale si possono specificare (utilizzando il carattere ';' come separatore)
    - Più coppie **`variabile=valore`**
    - Il campo di visibilità del cookie sia come path (**`path=/mydir/;`**) che come dominio rispetto al server (**`domain=.unisi.it;`** per un cookie da www.dii.unisi.it)
    - Il tempo di vita (**`expires=Sun, 22-Apr-2012 12:07:22 GMT;`**) che indica il timestamp dopo il quale il browser deve eliminare il cookie
    - Le opzioni **`Secure`** (il cookie è inviato solo su connessioni crittate) e **`HttpOnly`** (il cookie è manipolabile solo via HTTP e non javascript)
  - Il client invia il cookie al server oringine ad ogni richiesta con l'intestazione **`Cookie:`**
    - L'intestazione contiene i cookie come lista **`variabile=valore;`**

Cookie in PHP

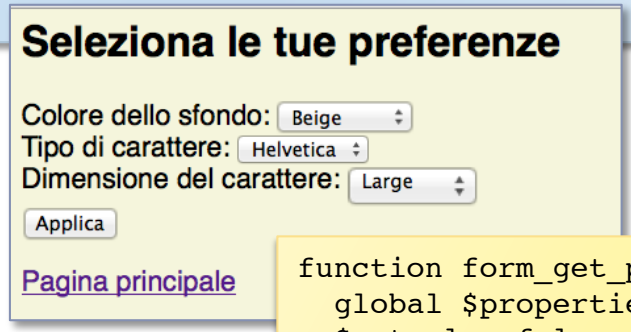
- Si possono definire cookie con l'istruzione

```
setcookie($name,$val,$expire,$path,$domain,$secure,$httponly)
```

  - dato che deve essere generata l'intestazione HTTP `Set-Cookie` l'istruzione deve essere eseguita prima di mandare in output il corpo della risposta
    - eventulmente si possono usare le funzioni di libreria per il buffering della risposta (`ob_start()`, `ob_flush()`)
  - `$name` è il nome della variabile associata al cookie e può essere usata anche per specificare un elemento di array
  - `$value` è il valore associato e viene inviato in forma *urlencoded*
  - Gli altri parametri sono opzionali e si riferiscono alle opzioni possibili per la definizione di un cookie
- Uno script PHP può accedere ai cookie usando l'array associativo **`$_COOKIE`**
  - Se il browser ha il cookie **`'mycookie'`** memorizzato la variabile **`$_COOKIE['mycookie']`** è inizializzata al valore inviato dal browser

Esempio di uso dei cookie (set)

```
<form action="17-pref.php" method="post”>
  Colore dello sfondo: <select name="background-color">
    <?php get_opts("background-color") ?>
  </select></br />
  Tipo di carattere: <select name="font-family">
    <?php get_opts("font-family") ?>
  </select></br />
  Dimensione del carattere: <select name="font-size">
    <?php get_opts("font-size") ?>
  </select><br />
  <input type="submit" value="Applica" /><br />
</form>
```




```
function form_get_prefs() {
  global $properties;
  $retval = false;
  foreach($properties as $prop=>$values) {
    $setprop = $_REQUEST[$prop];
      continue;
    setcookie("pref[$prop]","$setprop”);
    $retval = true;
  }
  return $retval;
}
```

```
HTTP/1.1 200 OK
...
Set-Cookie: pref[background-color]=Beige; pref[font-family]=helvetica; pref[font-size]=16px
....
```

Esempio di uso dei cookie (get)

```
GET http://127.0.0.1/~maggini/php/17-sitepage.php HTTP/1.1
Host: 127.0.0.1
...
Cookie: pref[background-color]=Beige; pref[font-family]=helvetica; pref[font-size]=16px
....
```

```
function get_style_pref() {
  global $properties;
  echo "<style>\n";
  foreach($properties as $prop=>$values) {
    $val = $_COOKIE['pref'][$prop];
    if(!empty($val))
      echo "body { $prop: $val }\n";
  }
  echo "</style>";
}
```

```
<head>
 <title>Seleziona preferenze</title>
 <style>
  body { background-color: Beige }
  body { font-family: helvetica }
  body { font-size: 16px }
 </style>
</head>
```

```
foreach($_COOKIE as $name=>$val) {
 echo "<li><b>$name</b>: <pre>";
 var_dump($val);
 echo "</pre></li>";
}
```

Selezione delle preferenze

Le preferenze per la generazione delle pagine Web (colore dello sfondo, tipo di font per i caratteri, dimensione dei caratteri) sono memorizzate in cookie che vengono generati dallo script associato alla pagina di selezione delle preferenze.

I cookie nella intestazione HTTP questione sono i seguenti.

- **pref**:

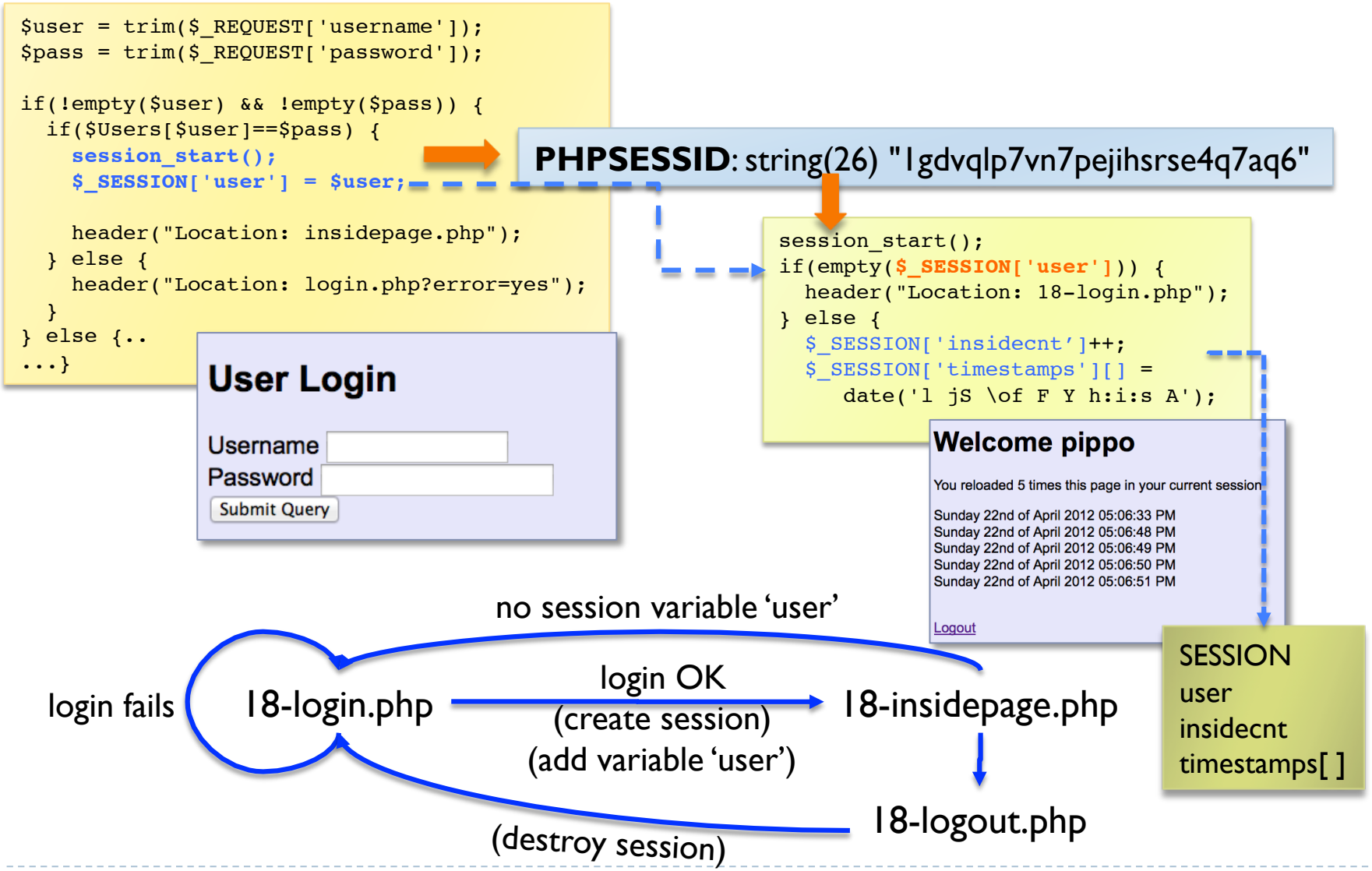
```
array(3) {
  ["background-color"]=>
  string(5) "Beige"
  ["font-family"]=>
  string(9) "helvetica"
  ["font-size"]=>
  string(4) "16px"
}
```

Selezione preferenze

Sessioni

- Il supporto PHP alle sessioni permette di mantenere dati sul server in una sequenza di richieste HTTP dallo stesso browser
  - Ad ogni utente (browser) è associato un identificatore univoco (SESSION ID)
  - Il session ID è scambiato in un cookie o nell'URL (variabile GET)
  - La sessione permette di memorizzare in modo persistente variabili nell'array `$_SESSION[]`
  - Quando un browser accede al sito il PHP controlla automaticamente (se attivo) o su richiesta (`session_start()`) se è presente un session ID valido nella richiesta e in tal caso rispristina l'ambiente salvato in precedenza
  - Per default l'ambiente viene salvato in file in una directory sul server specificata nella configurazione

Creazione di una sessione



Distruzione della sessione

- Per eliminare la sessione occorre
  - fare l'unset delle variabili di sessione
  - eliminare il cookie utilizzato per memorizzare il session ID
    - il nome predefinito del cookie si può ottenere con `session_name()`
    - il cookie si può eliminare specificando un timestamp di expire antecedente il timestamp attuale
  - liberare lo spazio sul server allocato per serializzare le variabili memorizzate nella sessione (session_destroy())

```
 session_start();
 $_SESSION = array();
 if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time()-42000, '/');
 }
session_destroy();
```

18-logout.php

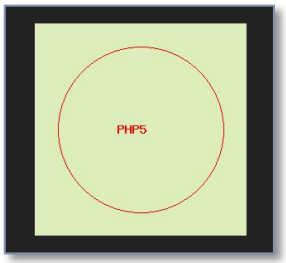
header()

- L'istruzione `header($head)` permette di inserire un campo di intestazione nella risposta HTTP
  - Deve essere eseguita prima che sia effettuato un qualsiasi output (anche uno spazio o un newline) in modo che **`$head`** sia correttamente nell'intestazione
  - Può essere usato per specificare varie informazioni come il tipo MIME della risposta (`Content-type:`), l'indirizzo di una redirezione (`Location:`), il controllo della cache (`Cache-control:`)

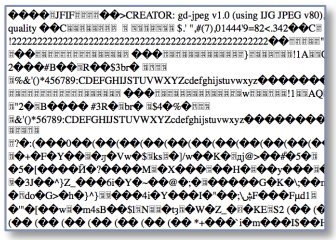
```
header("Content-type: "
      . image_type_to_mime_type(IMAGETYPE_JPEG));

$width = 256; $height = 256;
$txt = "PHP5";

$img = imagecreatetruecolor($width, $height);
$background = imagecolorallocate($img, 0xDD, 0xEE, 0xBB);
imagefill($img, 0, 0, $background);
$pencolor = imagecolorallocate($img, 0xFF, 0x00, 0x00);
imagearc($img,128,128,200,200,0,0,$pencolor);
imagestring($img,5,100,120,$txt,$pencolor);
imagejpeg($img);
imagedestroy($img);
```



Content-type: image/jpeg



Content-type: text/plain
(web server default)

PHP e database

- IL PHP ha librerie per la connessione a specifici DBMS (es. MySQLi) per la gestione di dati da uno script
  - Connessione con autenticazione al DBMS utilizzando la modalità prevista dallo specifico DBMS specificando un database

```
mysqli_connect(server,user,pwd,database)
```

  - Esecuzione di una query SQL

```
mysqli_query(query,connection)
```

  - Accesso alle righe del risultato della query

```
mysqli_fetch_row(data), mysqli_fetch_assoc(data), ....
```

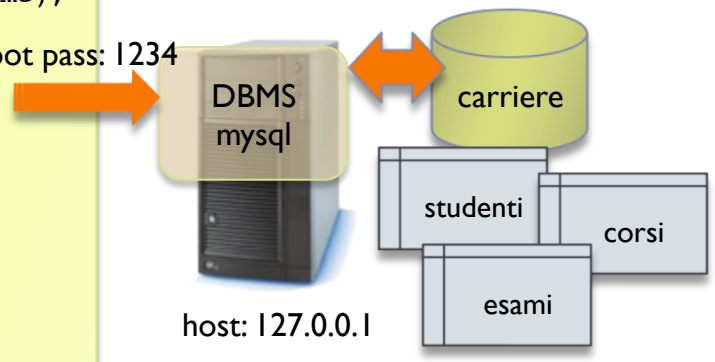
  - Deallocazione delle risorse

```
mysqli_free_result(data), mysqli_close(connection)
```

Esempio

```
$conn = mysqli_connect($dbmshost,$dbmsuser,$dbmspass,$dbname);
if(!$conn) {
  echo "<h1>Database connection problems</h1>"
      .mysqli_connect_error()."</body></html>";
  die();
}
$query = 'SELECT nome, cognome, titolo, voto
          FROM studenti, corsi, esami
          WHERE esami.matricola=studenti.matricola
                  AND esami.corso=corsi.codice
          ORDER BY cognome, nome, titolo';

$result = mysqli_query($conn,$query);
if(!$result) {
   echo "Database error: "
         .mysqli_error($conn)."</body></html>";
  mysqli_close($conn);
  die();
}
echo "<h2>Esami degli studenti</h2>";
echo "<table style=\"border: 2px solid gray;
            border-collapse: collapse\">\n";
while ($row = mysqli_fetch_row($result)) {
  echo "\t<tr>\n";
    foreach ($row as $col_value)
        echo "\t\t<td style=\"border: 1px solid black;\" >
            $col_value</td>\n";
  echo "\t</tr>\n";
}
echo "</table>\n";
mysqli_free_result($result);
mysqli_close($conn);
```



Esami degli studenti

giuseppe	mazzini	geografia	29
giuseppe	mazzini	storia	25
mario	rossi	geografia	25
mario	rossi	matematica	30
bianca	verdi	geografia	28
bianca	verdi	matematica	27