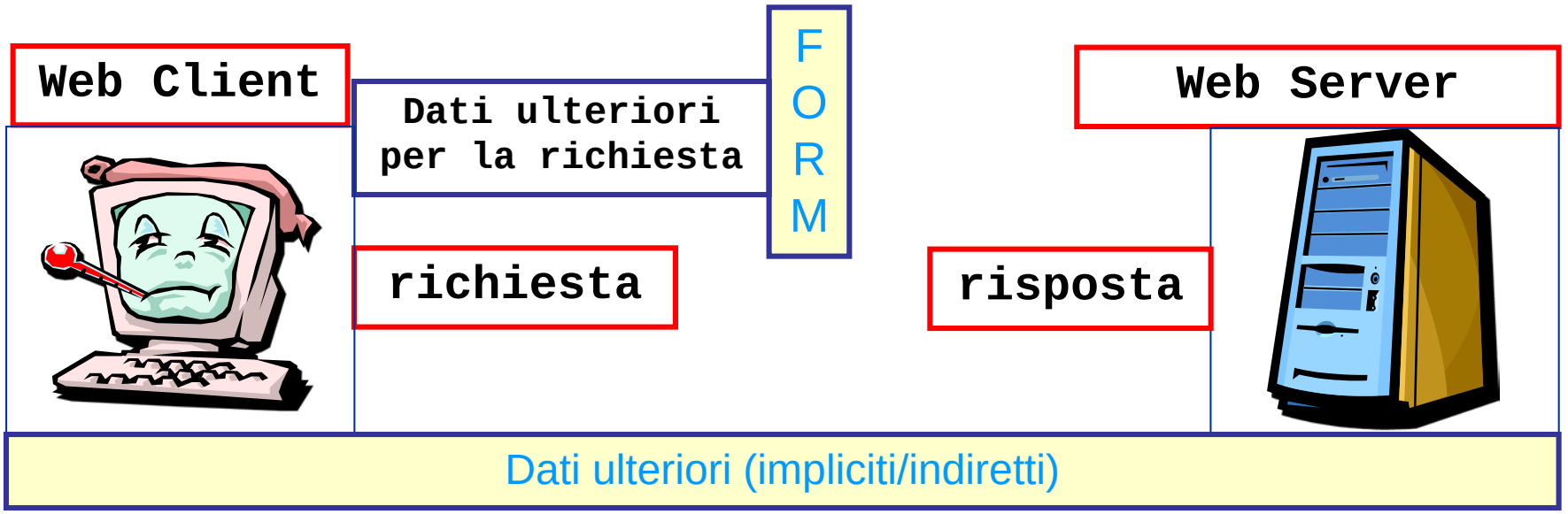
Programmazione in PHP – riepilogo preventivo



Client-side programming

CGI programming

PHP Programming

server-side

Ci occupiamo delle cose in giallino …

PHP?

Linguaggio di scripting

script = lista istruzioni da "interpretare" (+/- in opposizione a "compilare ed eseguire")

script php

file con istruzioni php (ed eventualmente codice HTML) che può essere oggetto di una **richiesta** http

1) Server side execution (mediato dal web server)
2) output = ... come un qualsiasi altro linguaggio ... ma nel nostro contesto: risorsa oggetto della **risposta** (primarily codice HTML contenuto nello script + codice prodotto dalle istruzioni)

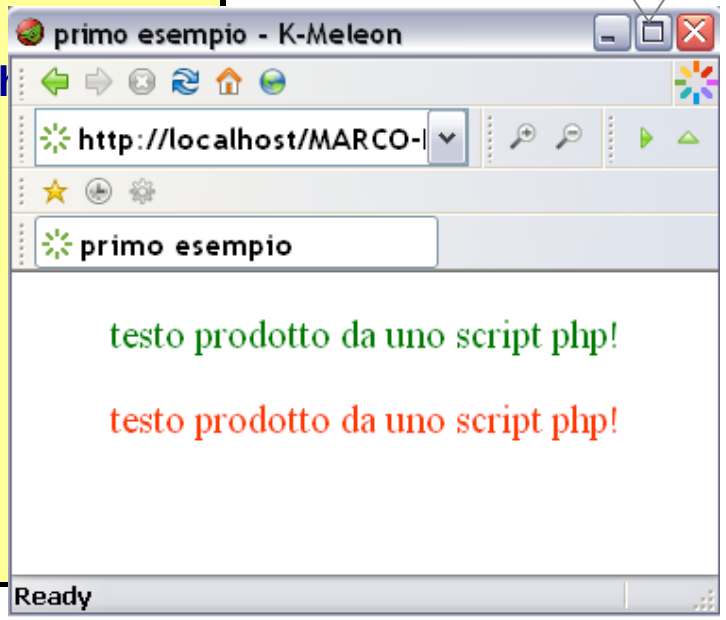
differenze con cgi programming

- lo script non deve necessariamente risiedere in directory particolari del server (dovunque in htdocs va bene, a meno di proibizioni stabilite da SO;
- codice HTML può essere presente nello script per conto suo (cioè non come prodotto di operazioni di output) ... anzi, tipicamente deve, dato che apparirà così com'è nel documento html prodotto.
- Viene creata la risorsa (il documento HTML), non il messaggio di risposta

PHP??

```
<?php echo '<?xml version=
8"?>'; ?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Stri
"http://www.w3.org/TR/xhtml1/DTD/x
<html xmlns="http://www.w3.org/199
xml:lang="en" lang="en">


<head> <title>primo esempi
<body>



<p style="color:green; text-align:center">
 <?php echo "testo prodotto da uno script ph
?>
</p>

<?php
echo "<p style=\"color:red; text-align:center\">";
  echo "testo prodotto da uno script php!";
  echo "</p>";
?>
</body></html>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head> <title>primo esempio</title></head>
<body>
<p style="color:green; text-align:center">
        testo prodotto da uno script php!
</p>

<p style="color:red; text-align:center">testo
prodotto da uno script php!</p>
</body></html>
```



PHP???

Personal Home Page tools

PHP/FI (Form Interpreter) ’95, Lerdorf

PHP/FI 2.0 ‘97

```
<!– codice -->
```

Php Hypertext Processor

PHP 3.0 ‘98 (A.Gutmans, Z.Suraski, R.Lerdorf)

PHP 4 2000-2002-...2008 (zend engine)

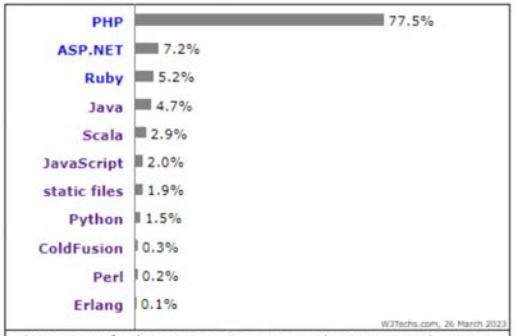
PHP 5 ~2005 …(zend engine 2.0)

implementazione "reale" della programmazione a oggetti, Pub/Pro/Pri members, Exception: Try/Catch/Throw, passing by ref., XSL, DOM/simpleXML, altre aggiunte o migliore efficienza

PHP 6 (?!) … 7 … 8

This diagram shows the percentages of websites using various server-side programming languages. See technologies overview for explanations on the methodologies used in the surveys. Our reports are updated daily.

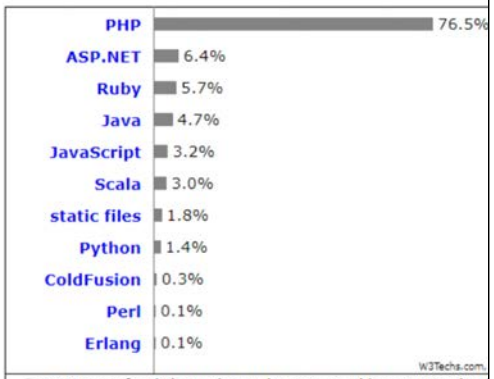
How to read the diagram:
PHP is used by 77.5% of all the websites whose server-side programming language we know.



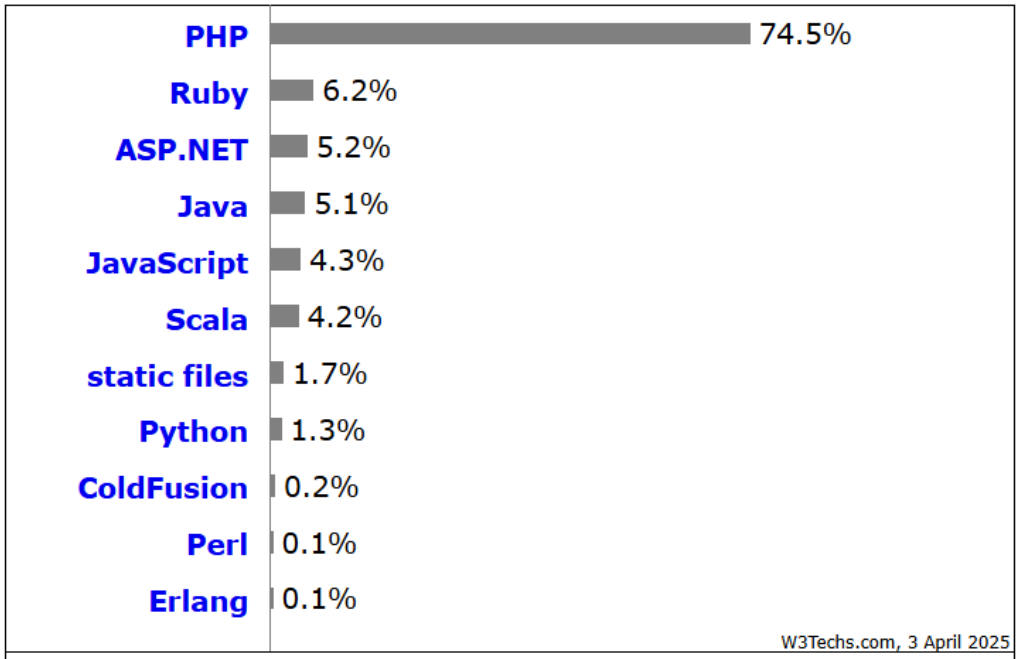
Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

The following server-side programming languages are used by less than 0.1% of the websites

- Miva Script
- C
- Lasso
- Smalltalk
- C++
- Go
- Tcl
- Haskell
- Lisp
- Ada
- Lua



How to read the diagram:
PHP is used by 74.5% of all the websites whose server-side programming language we know.



Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

The following server-side programming languages are used by less than 0.1% of the websites

- Miva Script
- C
- Smalltalk
- C++

6 (?!) ... 7 ... 8

firstPhp.php

istruzioni php sparse nel codice html

```
<?php echo '<?xml version="1.0" encoding="UTF-8"?>';
?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en">

<head><title>primo esempio</title></head>
<body>

<p style="color:green; text-align:center">
 <?php echo "testo prodotto da uno script php!\n"; ?>
</p>


     <?php
       echo "<p style=\"color:red; text-align:center
       echo "testo prodotto da uno script php!";
       echo "\n</p>\n";
?>

</body>
</html>
```

Nella pagina risultante, il contenuto dell'element p è l'output dell'istruzione echo

Nella pagina risultante, ci sarà la sequenza continua di caratteri prodotti dalle tre echo

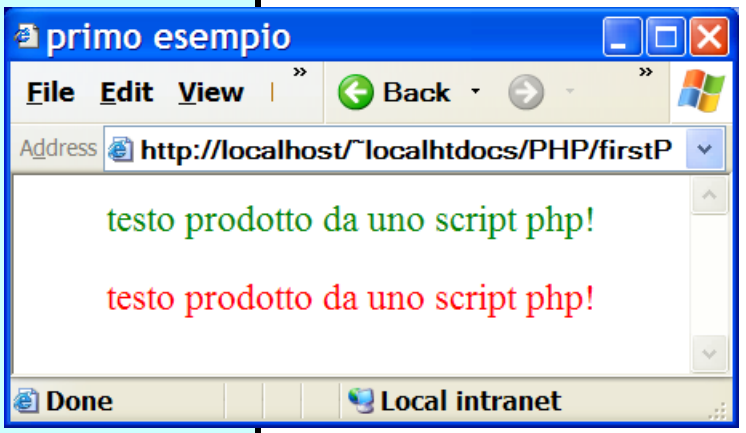
echo

- void, variable number of param (separati da ',');
- language construct (non parent.)
- escape char '\' - si usa per riuscire a stampare car. speciali
- operatore di concatenazione = '.' ("stringa1"."stringa2" = ?)

output di firstPhp.php

il browser ha chiesto
http://localhost/.../PHP.../firstPhp.php
la risposta contiene il codice xhtml mostrato

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en">

<head><title>primo esempio</title></head>
<body>

<p style="color:green; text-align:center">
testo prodotto da uno script php!
</p>


     <p style="color:red; text-align:center">
testo prodotto da uno script php!
</p>

</body>
</html>
```



che succede se si apre direttamente da file system con il browser
.../PHP.../firstPhp.php

PHP - sintassi di base

02-escaping.html.php
(https://www.php.net/manual/en/)

Ha molto di simile a C … C++, java, javascript, perl

- blocchi di codice { }
- terminatore di istruzione ;
- operatori ... **||**, **&&**, !, **==**, **<>**, **!=**, **?:** (espr. Cond.), … **===**, **!==**
- commenti al codice

```
# commento stile unix
// commento monolinea stile c++
/* commento multilinea stile c */
```

- ***html escaping*** cioè come far capire dov'è il codice php rispetto a quello html

<?php … ?> (in questo ambiente si è in "modalità php")

<? … ?> ← **short_open_tag On**

<% … %> ← **asp_tags On**

php.ini

PHP - “advanced” escaping

interspersed.php

(bah) perché disperdere codice php così finemente nel file?

```
<?php
if ($expression) {
    ?>
    <strong>This is true.</strong>
    <?php
} else {
    ?>
    <strong>This is false.</strong>
    <?php
}
?>
```

perché scrivere tanto codice html all'interno di costrutti echo o print è costoso in termini di tempo di esecuzione

invece il codice html esterno a script non viene considerato dal parser php e quindi non fa sprecare tempo di elaborazione.

```
<?php
if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIè) !== false) {
?>
<h3>strpos must have returned non-false</h3>
<center><b>You are using Internet Explorer</b></center>
<?php
} else {
?>
<h3>strpos must have returned false</h3>
<center><b>You are not using Internet Explorer</b></center>
<?php
}
?>
```

PHP - variabili

curiosità

Identificatori (lettera, oppure '_', all'inizio) preceduti da un '$'

```
$great = 'fantastic'
```

- quattro tipi di base (integer, float/double, boolean, string)

- nessuna dichiarazione indispensabile

- *type juggling* (tipo della variabile determinato in base a contenuto/contesto)

Valori booleani

`TRUE` `(!= 0)`

`FALSE` `(== 0)`

case insensitive

```
$a = 10;                          # $a integer
$c = "fine"                       // $c è stringa
$b = 6.3;
$c = $a + $b;                     /* $c è float */
$d = (int)$c;          // type casting ($d intero)
$e = settype($d, double);         # $d ora double
print(gettype($e));               // stampa boolean
if (is_int($d)) {      // is_type per type check
  $d += 4;  }
if (isset($d)) {       // la variabile "esiste" … vedi manuale
...}
```

vedere page.php, ipotizzare cosa dovrebbe mostrare se short_open_tags fosse On, e poi verificare

PHP - stringhe

Classiche stringhe.

Le cose interessante\i riguardano le modalità con cui il contenuto di una stringa può essere stampato direttamente così com'è, oppure "interpretando" il contenuto:

che effetto ha il nome di una variabile, o un carattere speciale, nella stringa?

Tre modi di scrivere stringhe (assumi **`$total=10000`**)

- single quoted:

- double quoted

- here doc / nowdoc

PHP - stringhe

Classiche stringhe. Le cose interessante\i riguardano le modalità con cui il contenuto di una stringa può essere stampato direttamente così com'è, oppure "interpretando" il contenuto: che effetto ha il nome di una variabile, o un carattere speciale, nella stringa? Tre modi di scrivere stringhe (assumi **`$total=10000`**)

- single quoted:

- double quoted

i nomi di var vengono *espansi;*
i car speciali sono dati con `'\'`:
`\n \r \t \" \\ \$`

```
echo "spendiamo $total \"euro\" all'anno\n";
```

produce

```
spendiamo 10000 "euro" all'anno
```

(e poi a capo)

- here doc

- nowdoc

PHP - stringhe

interpr.stringhe.html/php

Classiche stringhe. Le cose interessante\i riguardano le modalità con cui il contenuto di una stringa può essere stampato direttamente così com'è, oppure "interpretando" il contenuto: che effetto ha il nome di una variabile, o un carattere speciale, nella stringa? Tre modi di scrivere stringhe (assumi **`$total=10000`**)

- single quoted:

```
echo 'spendiamo $total euro all\'anno\n';   produce
        spendiamo $total euro all'anno\n
```

nulla viene interpretato; per scrivere `'` si scrive `\'`

- double quoted

i nomi di var vengono *espansi;*
i car speciali sono resi con `'\'`:
`\n \r \t \" \\ \$`

```
echo "spendiamo $total \"euro\" all'anno\n";        produce
                spendiamo 10000 "euro" all'anno     (e poi a capo)
```

- here doc

- nowdoc

PHP - stringhe

Classiche stringhe. Le cose interessante\i riguardano le modalità con cui il contenuto di una stringa può essere stampato direttamente così com'è, oppure "interpretando" il contenuto: che effetto ha il nome di una variabile, o un carattere speciale, nella stringa? Tre modi di scrivere stringhe (assumi **`$total=10000`**)

- single quoted:

- double quoted

- here doc

Tutto come per double quoted; stringa racchiusa tra <<<ID e ID; (a inizio riga) - no escaping

```
<?php  $lui="Ettore"; $lei="Andromaca"; $town = "Ilio"; $vaff="Achille";
echo <<<EOT
Hi there, my name is "$lei". Hi, my name is $lui. We were so happy in $town.
Then that hoodlum $vaff came.
All this in a heredoc string. BTW This should print a capital a: \x41
EOT;
```

Hi there, my name is "Andromaca". Hi, my name is Ettore. We were so happy in Ilio. Then that hoodlum Achille came. All this in a heredoc string. BTW This should print a capital a: A

PHP - stringhe

heredoc.php

Classiche stringhe. Le cose interessante\i riguardano le modalità con cui il contenuto di una stringa può essere stampato direttamente così com'è, oppure "interpretando" il contenuto: che effetto ha il nome di una variabile, o un carattere speciale, nella stringa? Tre modi di scrivere stringhe (assumi **`$total=10000`**)

- single quoted:

- double quoted

Hi there, my name is "$lei". Hi, my name is $lui. We were so happy in $town. Then that hoodlum $vaff came. All this in a heredoc string. BTW This should print a capital a: \x41

no escaping

- nowdoc

Tutto come per single quoted; stringa racchiusa tra <<<'ID' e ID; (a inizio riga) no escaping

```
<?php  $lui="Ettore"; $lei="Andromaca"; $town = "Ilio"; $vaff="Achille";
echo <<<'EOT'
Hi there, my name is "$lei". Hi, my name is $lui. We were so happy in $town.
Then that hoodlum $vaff came.
All this in a heredoc string. BTW This should print a capital a: \x41
EOT;
```

PHP - stringhe - 2 – esperimenti ...

- `$risultato = '$total= ' . $total;` $risultato è assegnato con la stringa **`$total= 10000`**

(remember, l'operatore **'.'** è quello di concatenazione di stringhe)

- `print('<br />il tipo di $total e\' '.gettype($total).": $total");`
  stampa **`<br />il tipo di $total è integer: 10000`**

- `printf (" la somma è %.2f <br />",  $total);`
  stampa **`la somma è = 10000.00 <br />`**

- `$risultato = sprintf ("la somma è %.2f <br />",  $total);`
(come prima, ma l'output è diretto nella stringa `$risultato` e non in … output)

PHP - variabili d'ambiente

https://www.php.net/manual/en/reserved.variables.server.php

Le variabili scambiabili tra client e server sono contenute in una serie di array

$_ENV (server environment)

```
$_ENV['HOME']; $_ENV['PATH']; $_ENV['USER'];
```

$_SERVER

```
$_SERVER['SERVER_NAME] - $_SERVER['PHP_SELF'] -
$_SERVER['HTTP_USER_AGENT'] - $_SERVER['REMOTE_ADDR']
```

$_GET (le var ricevute tramite richiesta con metodo *get "URL query parameters*)

$_POST (… *post*)

$_FILES (gestione file uploaded)

```
$_FILES['file']['name'];
```

$_COOKIE, $_SESSION, variabili cookie (sul client) e variabili session (sul server)

$_REQUEST, *get, post, cookie*

```
foreach ($_XXX as $key => $value) {
    echo "$key => $value\n";
}
```

si tratta di variabili (array) **superglobals** o autoglobals, cioè automaticamente visibili da qualunque funzione

PHP - variabili get e post

Supponendo di avere una form con il contenuto seguente

```
<form action="firstt.php" method="get">
<p>Nome: <input type="text" name="userName" value="Susy" size="30"></p>
<p>Cognome: <input type="password" name="userPassword" value="abby" size="30">
```

per accedere ai dati inviati dalla form

```
<h3>$_GET['userName']:              <?php echo $_GET['userName'] ?></h3>
<h3>$_REQUEST['userName']:          <?php echo $_REQUEST['userNamè] ?></h3>

<h3>$_GET['userPassword']:          <?php echo $_GET['userName'] ?></h3>
<h3>$_REQUEST['userPassword']:      <?php echo $_REQUEST['userName'] ?></h3>
```

prima vedi firstt1.php e prevedi quel che fa
poi usa first1.html

poi crea un nuovo file second.html, in cui ci siano due bottoni per la form. Dovrai fare anche una nuova action second.php, per questo file.
Il primo bottone funziona come quello di firstt1.html; il secondo fa sì che nella pagina prodotta, in fondo, ci sia una div con un piccolo racconto dell'orrore. Sol in firstt2

imgButton.html/php fa vedere come, oltre ai valori dei campi della form, possano viaggiare anche le coordinate del clic su un'immagine (che appunto qui è usata come bottone alternativo per l'invio della form).

PHP - costanti

constants.php

Identificatori associati stabilmente ad un valore.

- per definire una costante si usa `define()`
- il nome **non** è preceduto da **$** e si usa direttamente
- valori di soli tipi scalari (integer, boolean, float, string)

- in sostanza: globali

- non ridefinibili

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.
?>
```

- per convenzione: si usano solo maiuscole per gli identificatori di costante

- costanti predefinite: `PHP_VERSION, E_ALL, E_WARNING, M_SQRT2, M_SQRTPI …`

- costanti *magical*
  - **__LINE__** **__FILE__** **__FUNCTION__**
  - **__DIR__** (5.3) **__CLASS__** (4.3) **__METHOD__** (5.0)

PHP - array

Variabili definite con Il costrutto **array** (array indicizzati e *associativi* al contempo)

```
array (
    lista di valori, opzionalmente
    associati ad una chiave
    [key1 =>] value1,
    [key2 =>] value2,
    …
)
```

```
$a = array(75, 121, 10, "pistacchio", 22.99);
```

- gli elementi possono essere di qualunque tipo
- gli indici, interi o stringhe …

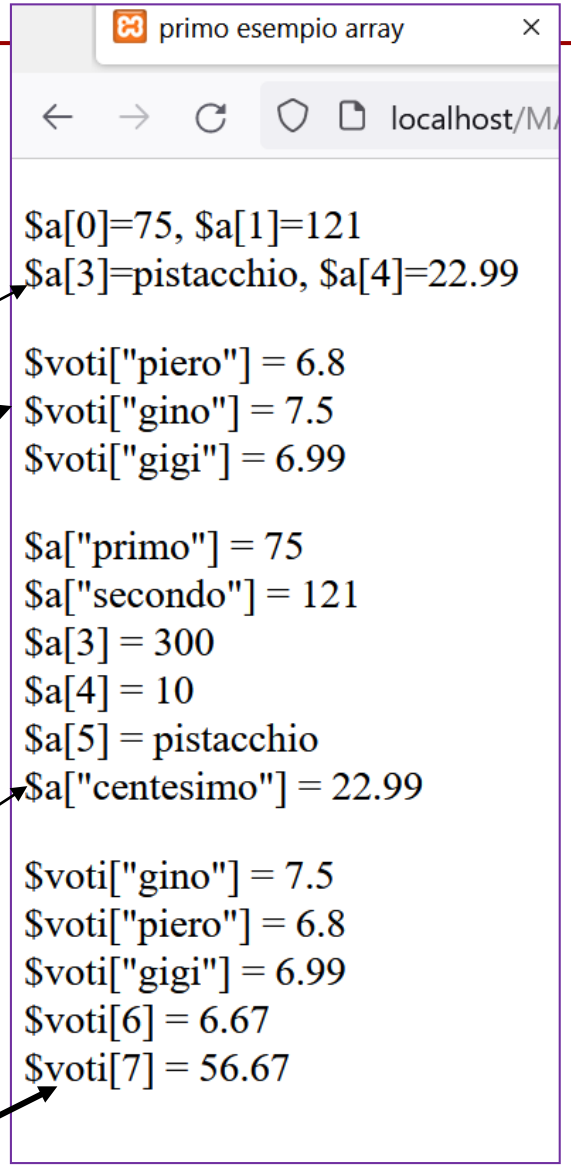
```
$voti = array("gino"=>7.5, "piero"=>6.8, "gigi"=>6.99);
```

- anche tutti e due i modi nel medesimo array

```
$a = array("primo"=>75, "secondo"=>121, 3=>300,
     10, "pistacchio", "centesimo"=>22.99);
```

- assegnazione diretta di nuovi elementi, possibile specificando un indice o anche no (vedi regole per l'indicizzazione automatica nella documentazione)

```
$voti[6]=6.67;
$voti[]=56.67;
```



Vedi esempioArray.php

PHP – strutture di controllo

```
if (condizione) {
 ...
}
else {
...
}
```

```
if (condizione) {
 ...
} elsif (cond) {
...
} elsif (cond) {
...
} else {
...
}
```

```
switch ($_GET[“submit”]) {
 case “invio1”: // cosa
        ...
    break;
 case “invio2”: // cosa
        ...
    break;
 ...
 default:       // almeno
}
```

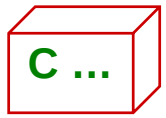
```
continue;
```

```
while ($total<$much) {
     // corpo ciclo
        ...
}
```

```
do {
       // corpo ciclo
...
} while (cond)
```

```
break;
```

```
for (exp1; exp2; exp3) {
       // corpo ciclo
...
}
```



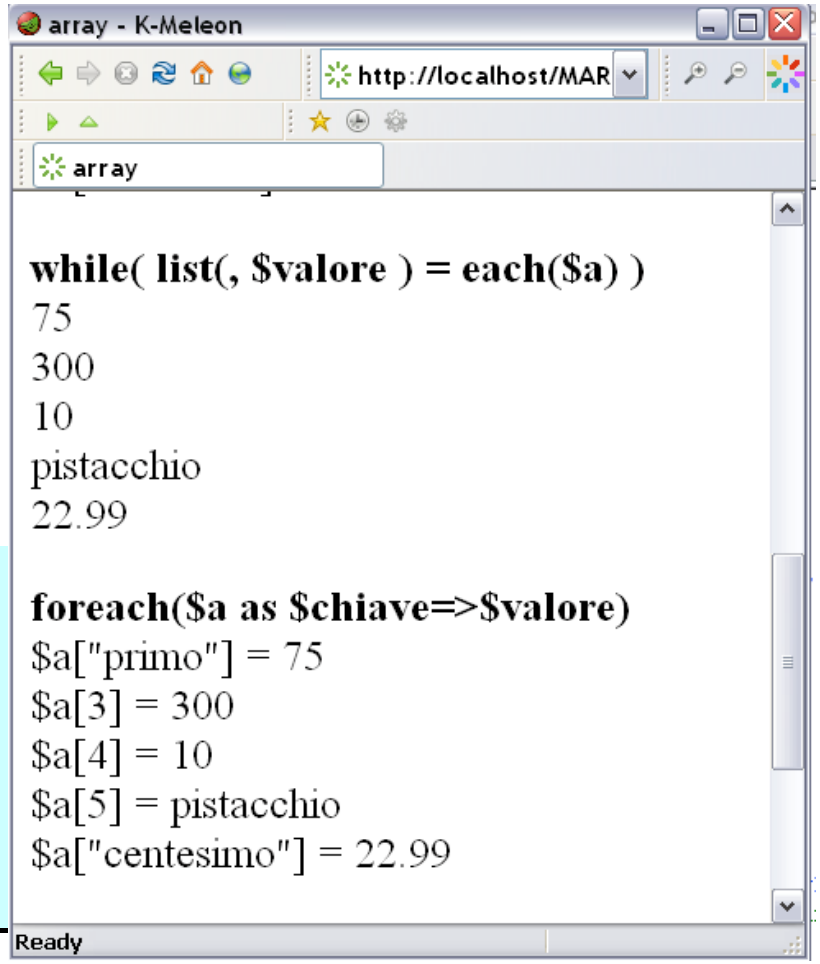
PHP – foreach –

Vedi esempioArray.2.php

foreach ($a as $key=>$value) scorre gli elementi di $a e assegna ogni volta una delle coppie chiave/valore alla coppia $key/$value.

Il reset è automatico all'inizio di foreach()

```
<?php
foreach($a as $chiave=>$valore)
  if (is_int($chiave))
    echo "\$a[$chiave] = $valore<br />";
  else
    echo "\$a[\"$chiave\"] = $valore<br />";
 ?>
</p>
```



NB $chiave e $valore sono esempi di identificatori decisi dal programmatore, in base al significato dei dati contenuti nell'array ... andrebbero ugualmente bene ($primo, $secondo), oppure ($coordinata, $valore_assegnato), o altro che abbia senso rispetto ai dati da gestire ...

PHP - error reporting

Vedi errors.display.php

poi in errors-display-corr c'e` la discussione del contenuto del primo script

Vari tipi di errore: i piu` frequenti

- NOTICE (forse non è niente ... es. si usa una variabile vuota)
- WARNING (probabilmente ci saranno impicci: header information, tipi parametri)
- PARSE (parentesi, **;**, stringhe mal fatte ... errori sintattici nello script: *FATAL*)
- ERROR (memoria? ...: FATAL)

```
<?php
error_reporting (0);                    // non voglio vedere
error_reporting (E_ALL);                // voglio vedere tutti i messaggi d'errore
error_reporting (E_ALL & ~E_NOTICE);        // tutti meno le NOTICE
...
```

Queste scelte sono possibili se nel file php.ini c'è `display_errors On`
ma invece il default consigliato se il sistema è “pubblicato” è `display_errors Off`
(la stampa di errori php è sconsigliabile in un sito web in funzione).

Si può assegnare display_errors opportunamente nella fase di apprendimento di php, o di costruzione del sito, per poi assegnarlo ad Off (in modo che gli utenti del sito non vedano gli errori – possibile minaccia alla sicurezza, piu` che brutta figura).

Si può assegnare Off/0 oppure On/1) solo per la durata dello script, con

```
ini_set('display_errors', 0)
```

PHP - funzioni

Caratteristiche salienti :

- passaggio per valore
- (scope) variabili locali / variabili globali
- nesting
- variabili "funzione" (`$f = fun;  $f(a,b,v);`)
- return (!)
- passaggio per (tramite) indirizzo
- variabili static
- ricorsione
- numero variabile di argomenti

PHP - funzioni - variabili static

Una variabile static mantiene il proprio valore tra una chiamata e l'altra della funzione in cui è definita.

Solo durante la prima chiamata, l'inizializzazione prevista (solo con un valore costante, niente espressioni) viene eseguita.

```
<?php
function Test() {
    static $a = 0;

    echo $a;
    $a++;
}
?>
```

Vedi (costanti e funzioni) static.php

PHP - funzioni - scope locale e globale

Le variabili usate (visto che non sono dichiarate) in una funzione sono "locali a quella funzione": ogni riferimento ad una variabile locale è risolto solo cercando nella funzione:

```
<?php
$a = 1;

function Test() {
  echo $a;  /* ref. local scope var */
}

Test();
?>
```

```
non stampa nulla:
$a nella funzione è vuota
```

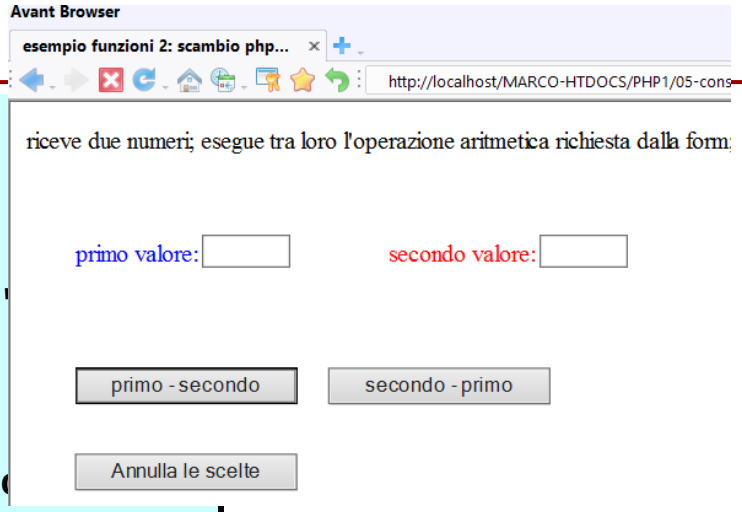
```
<?php
$a = 1;

function Test1() {
 global $a;
  echo $a;
}

function Test2() {
  echo $GLOBALS["a"];
}
?>
```

```
Dichiarazione global nella funzione: $a, nella funzione, sarà
    gestita con scope globale (cercata a top level);

array $GLOBALS: arr.associativo, contenente tutte le variabili
    globali disponibili ($GLOBALS, come $_GET, $_POST … è
    superglobal, cioè disponibile in qualunque funzione)
```

```
Apropos di questi array globali, vedi
funzioni1.php (c'è un esempio di uso di DATE,
e alla fine la stampa di GLOBALS
```

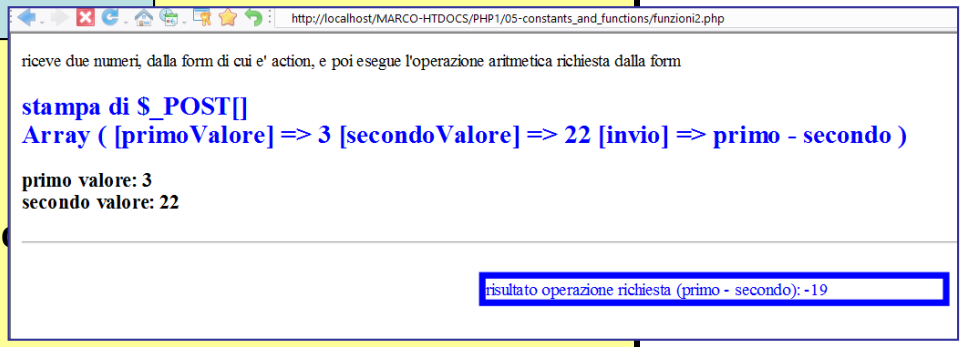
funzioni2.html/.php

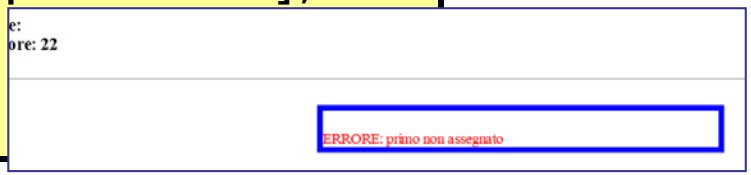
```
<?xml version="1.0" encoding="UTF-8"?>
...
<form action="funzioni2.php" method="post">

<p>...<input type="text" name="primoValore" size="5"
...
<input type="text" name="secondoValore" size="5">
...
<input type="submit" name="invio" value="primo - se
<input type="submit" name="invio" value="secondo - primo">
...
```



Una form e lo script che ne gestisce i dati

```
...
<?php print_r($_POST) ?>
...   $msg="vuota!";

 if ($_POST["primoValore"]=="")
   $msg="<br />ERRORE: primo non assegnat
// idem su secondoValore
 ...
 if ($msg=="vuota!") {.........
   if ($_POST["invio"]=="primo - secondo")
     $msg .= $_POST["primoValore"] - $_POST["secondoValore"];
   else $msg .= $_POST["secondoValore"] - $_POST["primoValore"];
 }
...
 echo $msg;
```





PHP - variabili a valore multiplo nelle form

Le variabili php possono essere associate ad un solo valore; quindi per ricevere le selezioni di una `<select>` o di una `<input ...checkbox>` si usa un array

```
<input type="checkbox" name="generi[]" value="Gialli" > Gialli<br />

<select name="fiori[]"  size="5" multiple="multiple">
```

formex.php.html

```
<h3>Hai scelto i seguenti generi letterari:<br />
<?php
  if (!isset($_POST['generi']))
    echo "<p style='text-align: center'>nessuno</p>\n";
  else {
    echo "<ul>\n";
    foreach($_POST['generi'] as $c=>$v)
       echo "<li>".$c."-".$v."</li>";
    echo "</ul>\n";
  }
?>
</h3>
```

Hai scelto i seguenti generi letterari:

- **0-Fantascienza**
- **1-Horror**

vedi formex.html. Poi formex.Ver1.php
Poi scrivi formex.Ver2.php, per gestire diversamente da Ver1 i dati della form e fallo eseguire come action della form.

PHP - espressioni regolari (1/4)

Un'espressione regolare viene costruita mediante un linguaggio dedicato: descrive uno *schema di stringa, un modello, un pattern di caratteri*, costruito mediante delimitatori, metacaratteri, quantificatori, caratteri, cifre … una stringa può essere controllata, per determinare se verifica quello schema oppure no … preg_match()).

- atomi (^, $, …)

```
"/MSIE/"                          modello per le stringhe che contengono "MSIE"
"/^MSIE/"                         le stringhe che iniziano con MSIE
"/^MSIE$/"                        stringhe uguali a MSIE ('$'=finestringa)
preg_match("/MSIE\t/", $str)      true se $str contiene MSIE seguita da tab
preg_match("/MSIE|gecko/", $s)          $s contiene "MSIE" oppure "gecko"
preg_match("^a.", $str)           $str comincia con 'a', seguito da un car
```

- specificatori di quantità (quante volte l'atomo può apparire per andar bene)

```
preg_match("/^MSIE.*/", $str)
                          $str = MSIE seguito da qualsiasi numero di carat
preg_match("/^MSIE.+/", $str)
                          $str = MSIE seguito da almeno un carattere
preg_match("/^MSIE.?/", $str)
                          $str = MSIE seguito da un carattere opzionale
preg_match("/^a(...)+/", $str)
                          'a' seguita da almeno una "(sottostringa)" di 3 car
```

Vedi reg.exp.0/1.php

PHP - espressioni regolari (2/4)

- classi di carattere (predefinite o definite da utente)

```
"/^[z[:digit:]q]al/"   modello per le stringhe che iniziano con una z o
                       cifra o q, seguita da 'a' e poi da 'l'

"/^[[:alpha:]]a/"      ... le stringhe che iniziano con una lettera ,
                       seguita da a

preg_match("/[a-d]+/", $str)  $str contiene una sequenza di almeno una
                              lettera tra a,b,c,d,
```

```
preg_match("/^([[:alpha:]]+)/([[:digit:]\.]+)( .*)$|MSIE/", $str)
```

$str corrisponde al pattern se

contiene una sequenza di sottostringhe come segue: la prima composta da almeno una lettera, la seconda, preceduta da '/', è composta da una sequenza di almeno una cifra-o-**punto**; la terza è iniziata da uno spazio e termina la stringa con una sequenza qualsiasi di caratteri, **oppure** contiene "MSIE".

Schema di esempio (matcha tutte e due le possibiità)

?

```
[[:space:][:alnum:]]
Matches any character that is either a
white space character or alphanumeric.
```

Vedi reg.exp.0/1/2.php

PHP - espressioni regolari (2/4)

- classi di carattere (predefinite o definite da utente)

"/^[z[:digit:]q]al/" modello per le stringhe che iniziano con una z o cifra o q, seguita da 'à e poi da 'l'

"/^[[:alpha:]]a/" ... le stringhe che iniziano con una lettera , seguita da a

preg_match("/[a-d]+/", $str) $str contiene una sequenza di almeno una lettera tra a,b,c,d,

preg_match("/^([[:alpha:]]+)/([[:digit:]\.]+)(.*)$|MSIE/", $str)
$str corrisponde al pattern se
contiene una sequenza di sottostringhe come segue: la prima composta da almeno una lettera, la seconda, preceduta da '/', è composta da una sequenza di almeno una cifra-o-**punto**; la terza è iniziata da uno spazio e termina la stringa con una sequenza qualsiasi di caratteri, **oppure** contiene "MSIE".

Schema di esempio (matcha tutte e due le possibilità)
?

[[:space:][:alnum:]]
Matches any character that is either a white space character or alphanumeric.

Vedi reg.exp.0/1/2.php

PHP - espressioni regolari (2/4)

- classi di carattere (predefinite o definite da utente)

`"/^[z[:digit:]q]al/"` modello per le stringhe che iniziano con una z o cifra o q, seguita da ‘à e poi da ‘l’

`"/^[[:alpha:]]a/"` ... le stringhe che iniziano con una lettera , seguita da a

`preg_match("/[a-d]+/", $str)` $str contiene una sequenza di almeno una lettera tra a,b,c,d,

`preg_match("/^([[:alpha:]]+)/([[:digit:]\.]+)( .*)$|MSIE/", $str)`
$str corrisponde al pattern se
contiene una sequenza di sottostringhe come segue: la prima composta da almeno una lettera, la seconda, preceduta da '/', è composta da una sequenza di almeno una cifra-o-**punto**; la terza è iniziata da uno spazio e termina la stringa con una sequenza qualsiasi di caratteri, **oppure** contiene "MSIE".

Schema di esempio (matcha tutte e due le possibiità)

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

`[[:space:][:alnum:]]`
Matches any character that is either a white space character or alphanumeric.

Vedi reg.exp.0/1/2.php

PHP - espressioni regolari (3/4)

- "/modello/": check case sensitive
- "/modello/i": check case insensitive
- terzo parametro (opzionale) per preg_match: array con le corrispondenze tra sottoespressioni e sottostringhe

```
<?php
preg_match(    "/^([[:alpha:]]+)\/([[:digit:]\.]+)( .*)$/",
               $HTTP_USER_AGENT,
               $matches);

$browser=$matches[1];
$browserVer=$matches[2];
$browserDesc=$matches[3];
?>
```

```
matches[0] =   l'intera stringa soddisfacente il pattern;
matches[1] =   la prima sottostringa, corrispondente alla prima
               sottoespressione
matches[2] =   la seconda ...
matches[3] =   la terza
                                                    (massimo 10)
```

Vedi reg.exp.3.php; poi in reg.exp.3sol.php c'e` la soluzione

PHP - espressioni regolari (4/4)

Vedi reg.exp.form.php

```
if (isset($stringaFornita)) {
   if ($invio=="opzione1") {
     $opzioneUsata="opzione1";
     preg_match("/$stringaFornita/",
     $...HTTP_USER_AGENT, $riscontro);
   } else {
     $opzioneUsata="opzione2";
     preg_match("/$stringaFornita/i,
     $...HTTP_USER_AGENT, $riscontro);
   }
   $risultato="<p style=\"color:red;\" >risultato
dell'ultimo confronto (usando $opzioneUsata su
$stringaFornita): match= ".$riscontro[0]."</p>";
}
?>      <?xml version="1.0" encoding="UTF-8"?>
...
browser: <?php echo $browser ?><br />
versione del browser: <?php echo $browserVer ?><br />
descrizione del browser: <?php echo $browserDesc ?>
...
<?php echo "$risultato";
echo "<form action=\"$...PHP_SELF\" method=\"get\">"
... stringaFornita: <input type="text"
        name="stringaFornita" size="30">
...
<input type="submit"
       name="invio" value="opzione1">
```

risorse

php

- **`Pagina web corso…/PHP1/...`** *(esperimenti descritti a lezione)*

- *www.**php**.net* *documentazione (anche ricerca su funzioni e caratteristiche)*

- *www.php.net/download-docs.php* *(documentazione, direttamente dalla sorgente*

www.phpclasses.org *www.w3schools.com*

apache, php, mysql, per linux e windows e mac

- *http://www.apachefriends.org/*
- *http://www.easyphp.org/* *(devserver)*
- *http://www.wampserver.com/en* *https://en.wikipedia.org/wiki/WampServer*

Documentazione su mysql

- *www.**mysql**.com* *https://mariadb.org/*

Un'applicazione per gestire I db di mysql (ma è utile conoscere anche la "linea di comando" mysql)

- ***phpmyadmin**.sourceforge.net*

Attività in laboratorio / prodotti individuali (1/2)

PHP-0

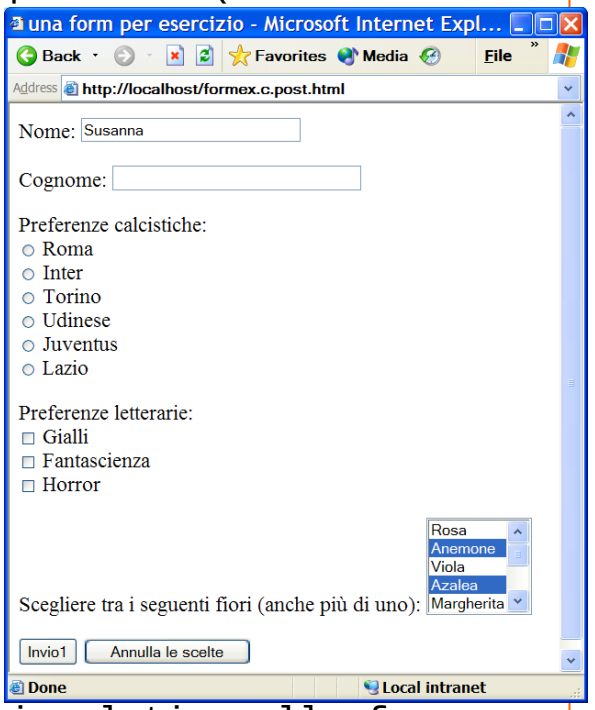
Ripercorrere le slides della lezione: sperimentare e modificare gli esempi visti. Su uno o piu` degli argomenti trattati in questa lezione, costruire una pagina html (o piu` pagine collegate) in cui siano spiegati I concetti appresi (elaborati autonomamnete anche con l'uso dei testi) e siano riportati esempi (mostrando/discutendo e permettendo di attivare script php adatti (scelti tra quelli qui visti, o creati per l'occasione.

PHP-1

Dopo la parte sulle variabili, risolvere questo quizzino: che fa lo script seguente?

```
<?php $x = "<ol>";
      $y = "<li>";
      $z = "</ol>";
      echo"$x$y aaa $y bbb $y ccc $z";?>
```

Se non già fatto, ripercorrere l'esempio in cui veniva attivata phpinfo() cercando di individuare quante piu` variabili d'ambiente e spiegandone il significato (andandone a cercare la definizione nel file di documentazione menzionato tra le risorse.



PHP-2

Ricordando e sfruttando quanto prodotto nell'esercizio cgi relativo alla form a fianco, costruire uno script php che sia action della form e che produca una pagina con un riassunto delle scelte eseguite nella form. La pagina prodotta dovrebbe contenere anche delle "conseguenze delle scelte" (come mostrare immagini o indirizzi di interesse o emettere giudizi sulle scelte, o valutarle dando un punteggio o dei giudizi in base a qualche criterio chiaramente esposto.

Attività in laboratorio / prodotti individuali (2/2)

PHP-3

Realizzare una pagina web, eser.php.3.html, e un relativo script di action, eser.php.3.php, in modo da fornire all'utente
- la possibile selezione di almeno una tra alcune voci (funzioni predefinite, costrutti e aspetti di php: ad. es. un menu` a tendina con foreach, while, list, array, scope, passaggio parametri, date(), ereg(), …)
- e produrre, in base alla selezione fatta dall'utente, una pagina in cui sono date spiegazioni su quanto richiesto.

Facoltativamente, rendere possibile la selezione di voci anche nella pagina prodotta in risposta ad una richiesta.

PHP-4

Realizzare una pagina web in cui venga fornita dall'utente una stringa di caratteri e venga prodotta una validazione di quella stringa come codice zip statunitense. Un codice zip, da quelle parti, è una sequenza di 5 cifre, seguita opzionalmente da una sequenza di al piu` 4 cifre separata dalla prima con un '-'.

PHP-5

Modificare l'esempio imgButton, in modo che lo script gestisca aree di una figura/bottone. Ampia libertà sugli scopi dello script (ad esempio, in base al punto di click, si forniscano informazioni su distinte funzioni predefinite php, oppure si indirizzi l'utente su pagine distinte).

Se non lo si è fatto in partenza, realizzare una versione dell'esercizio che faccia a meno del file .html ma contenga tutto nel file eser.php.5.php, avendo la form PHP_SELF come action.