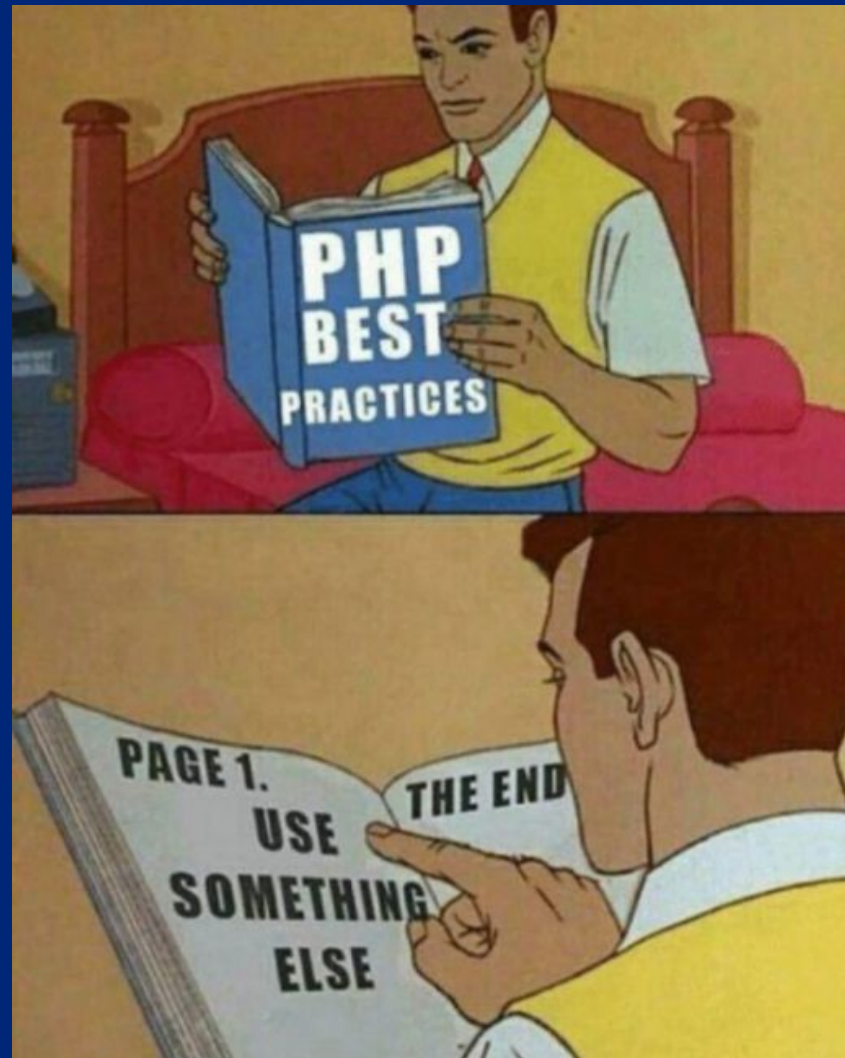


PHP (1)



Marina Ribaud, marina.ribaud@unige.it

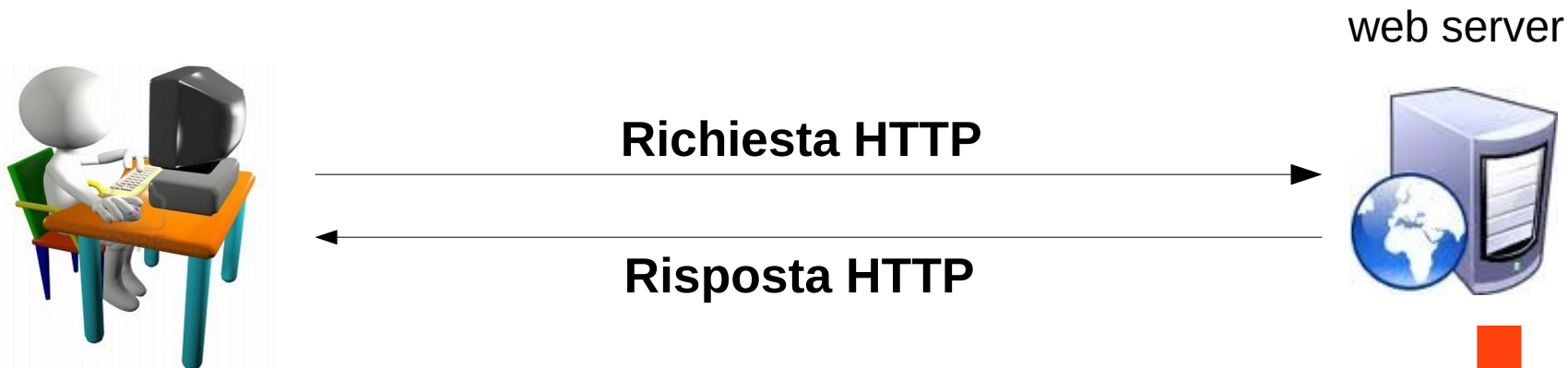
Programmazione lato server

2



Programmazione lato server

3



Sul server

- **Apache** per la comunicazione
- Interprete **PHP** per la logica applicativa
- **MySQL** o **MariaDB** per l'accesso ai dati

PHP

4

- Linguaggio di scripting, open source, server side, sviluppato per il web
- Il progetto inizia nel 1994 ad opera del danese Rasmus Lerdorf



I'm not a real programmer. I throw together things until it works then I move on. The real programmers will say Yeah it works but you're leaking memory everywhere. Perhaps we should fix that. I'll just restart Apache every 10 requests.

— Rasmus Lerdorf —

AZ QUOTES

PHP: struttura di un file

5



Un file con codice PHP deve essere salvato in formato **testo** ed estensione **.php**

PHP: struttura di un file

6

- PHP è **HTML-embedded**, cioè gli script possono essere inseriti nelle pagine HTML in cui devono produrre i loro output
- Il **web server riconosce le pagine PHP**, distinguendole da quelle statiche, sulla base dell'estensione .php e richiama l'interprete
- Il browser riceve solo il codice HTML generato: **quindi vede cosa produce lo script ma non come arriva al risultato**

Vedi: http://www.w3schools.com/php/php_intro.asp

PHP: come si inserisce codice?

7

Short style // deprecated

<?

PHP code here

?>

XML style

<?php

PHP code here

?>

SCRIPT style // deprecated

<script language="php">

PHP code here

</script>

PHP: esempio

8

Sul server: script hello.php

```
<html>
<head><title>Hello World</title></head>
<body>
<?php
    echo "<h1>Hello World</h1>";

?>
</body>
</html>
```



Sul browser

```
<html>
<head><title>Hello World</title></head>
<body>
<b><h1>Hello World</h1>
</body>
</html>
```

NB: i file PHP vanno sempre richiamati passando tramite il web server, sul computer di casa **[http://localhost/...](http://localhost/)**

PHP: struttura lessicale

9

Le istruzioni devono essere **separate** tra loro dal ;

Commenti

es. *// questo è un commento*

es. */* questo è un commento */*

es. *# questo è un commento*

Vedi: http://www.w3schools.com/php/php_syntax.asp

PHP: struttura lessicale

10

Gli **identificatori** devono iniziare con il simbolo **\$**

✓ es. \$firstname
 \$lastname
 \$email

PHP è **case sensitive**, \$a != \$A

Per gli identificatori **non** si possono usare le keyword del linguaggio

PHP: keyword

11

and
break
case
class
continue
default
do
else
elseif
extends
false
for
foreach
function
global
if

include
list
new
not
or
require
return
static
switch
this
true
var
virtual
xor
while

PHP: variabili e scope

12

- Le variabili in PHP **non** devono essere dichiarate, basta assegnare dei valori
- Le variabili PHP hanno quattro scope:
 - local
 - global
 - static
 - parametri delle funzioni

<https://www.phptutorial.net/php-tutorial/php-variable-scopes/>

PHP: local

13

```
<?php
    $a = 1;           // global scope

    function localscope()
    {
        return $a; // reference to local scope variable
    }

    echo localscope();
?>
```

PHP: global

14

```
<?php
    $a = 1;           // global scope
    $b = 2;

    function sum()
    {
        global $a, $b; // global scope
        return $a + $b;
    }

    echo sum();
?>
```

PHP: static

15

```
<?php
    $a = 1;    // global scope
    $b = 2;

    function sum()
    {
        global $a, $b; // global scope
        // local variable, that keeps values among successive calls
        static $sum;

        $sum += $a + $b;
        return $sum;
    }

    echo sum();
    echo sum();
    echo sum();
?>
```

PHP: parametri

16

```
<?php
    $a = 1;           // global scope
    $b = 2;

    function sum($a,$b)
    {
        return $a + $b;
    }

    echo sum($a,$b);
?>
```


PHP: variabili superglobali

17

- **Array** di variabili automaticamente globali, disponibili in ogni script e in ogni scope
- Esistono diversi array superglobali, ognuno con una funzione diversa
 - **\$_GLOBALS** contiene un riferimento a tutte le variabili disponibili nello scope globale
 - **\$_SERVER** insieme di variabili relative al server web o legate all'ambiente di esecuzione dello script corrente
 - **\$_GET**, **\$_POST**, **\$_COOKIE**, **\$_SESSION**

PHP: tipi di dati

18

Boolean

Integer

Float

String

Array

Object

PHP è **debolmente tipato**

Il **tipo** di una variabile può **cambiare dinamicamente**,
a seconda del valore che viene assegnato ad essa

PHP: costanti

19

```
define("NOMECOSTANTE", valore);
```

Per convenzione per i nomi delle costanti si usano i caratteri maiuscoli

Per vedere i valori delle costanti built-in di PHP (e molte altre informazioni) si può usare il seguente file **test.php**

```
<?php  
    phpinfo();  
?>
```

PHP: istruzioni

20

- + o - come il C (e JavaScript)
- Concatenazione tra stringhe
 - `$msg = 'Hello ' . $nome;`
 - `$msg = "Hello " . $nome;`
 - `$msg = "Hello $nome";` // solo con "..."
- Per l'output
 - echo** "Hello world\n";
 - print** ("Hello world\n");
 - print_r** (\$_POST);

PHP: istruzioni

21

- Output multilinea <<< (here-document, heredoc)
<?php

echo <<<**STAMPA**

stringa di output su

più linee che verrà stampata

come scritta, preservando gli spazi.

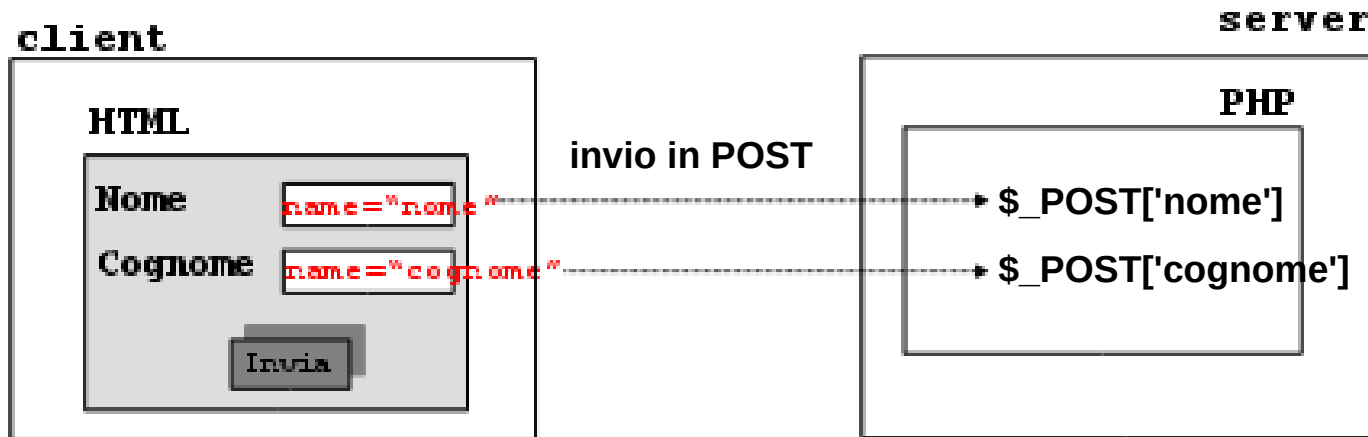
STAMPA; // la chiusura del comando deve essere a inizio riga

?>

PHP: dati in arrivo dal client

22

Ogni dato scritto dall'utente in un **form**, una volta spedito, viene memorizzato in una **variabile PHP**



PHP: dati in arrivo dal client

23

- I dati inviati con il metodo POST sono memorizzati nell'array **`$_POST[]`**
- I dati inviati con il metodo GET sono memorizzati nell'array **`$_GET[]`**
- Si può vedere il contenuto di questi array (e di tutti gli array associativi) con **`print_r()`**

PHP: dati in arrivo dal client

24

In passato la direttiva **register_globals = On** nel file **php.ini** permetteva di usare lo short style (\$nome, \$cognome) oggi deprecato

... compromise the security of your scripts. **With form variables automatically turned into global variables there is no obvious separation between variables that you have created, and untrusted variables that have come directly from the user. ...**

PHP: dati in arrivo dal client

25

`$nomevar`

`// deprecated`

`$_POST['nomevar']`

`// ok`

`$_GET['nomevar']`

`// ok`

`$_REQUEST['nomevar']`

`// medium style`

`$HTTP_POST_VARS['nomevar']`

`// long style deprecated`

`$HTTP_GET_VARS['nomevar']`

PHP: salvare dati sul server

26

Esistono due metodi per salvare i dati in arrivo dal client

- 1) **file**
- 2) **database** // lo vedremo più avanti

PHP: gestione file

27

- Molto simile al C
 - fopen() <http://php.net/manual/en/ref.filesystem.php> ecc.
 - fclose() - Closes an open file pointer
 - fgets() - Gets line from file pointer
 - fread() - Binary-safe file read
 - fwrite() - Binary-safe file write
 - fsockopen() - Open Internet or Unix domain socket connection
 - file() - Reads entire file into an array
 - file_exists() - Checks whether a file or directory exists
 - is_readable() - Tells whether a file exists and is readable
 - ...

PHP: gestione file

28

- Per ragioni di sicurezza se si scrivono dati sensibili in un file sul server, questo deve essere salvato **fuori** dalla **DocumentRoot**

```
$filename = $_SERVER['DOCUMENT_ROOT'] . "../../../private/mydata.txt";  
$fp = fopen($filename,"a");
```

NOTA: il nome del file non viene **MAI** scelto dall'utente

- Se non si specifica alcun pathname, il file viene creato nella stessa directory dello script PHP che apre il file stesso

DocumentRoot

29

- Under **XAMPP** root directory there is a folder called **htdocs**. That's where **you should put your web site related stuff**. For each web site you create, it's better to **create a folder inside htdocs folder** and then put content inside that to avoid conflicts.
- The **DocumentRoot** is the **top-level directory in the document tree visible from the web** and this directive sets the directory in the configuration from which **Apache2** or HTTPD looks for and serves web files from the requested URL to the document root. For example:
DocumentRoot
"/var/www/html"

PHP: occhio ai permessi!

30

Se si ottiene un errore come quello seguente si deve verificare che lo script abbia i permessi per scrivere sul file

Warning: fopen(users.txt): failed to open stream: Permission denied in /var/www/php/register.php on line 14

Nella maggior parte dei sistemi **gli script girano con l'utente del web server** (di solito www-data) oppure con l'utente proprietario della directory che contiene lo script

PHP: occhio alla concorrenza!

31

```
$fp=fopen("users.txt","a");  
$line = ".....";  
flock($fp,LOCK_EX);  
fwrite($fp,$line);  
flock($fp,LOCK_UN);  
fclose($fp);
```

LOCK_SH to acquire a shared lock (reader)
LOCK_EX to acquire an exclusive lock (writer)
LOCK_UN to release a lock (shared or exclusive)

Vedi: <http://php.net/manual/en/function.flock.php>

PHP: include/require

32

Nel caso di documenti in cui si ripetono più volte le stesse parti si possono usare le istruzioni **include()**, **require()**, **require_once()**

 **warning** se non
trova il file

 **fatal error** se non
trova il file

```
<?php
```

```
include("comuni/header.php");  
include("comuni/navbar.php");  
include("comuni/leftmenu.php");
```

```
page content here  
page content here
```

```
include("comuni/footer.php");
```

```
?>
```