1 Pagine Web Statiche e Dinamiche

Le pagine Web possono essere sia *statiche* che *dinamiche*. Le prime, presentano gli stessi contenuti a tutti gli utenti che vi accedono (ad esempio, la pagina web del *Curriculum Vitae*); le seconde possono mostrare contenuti diversi in base alle azioni effettuate dagli utenti (ad esempio, il *login* di accesso all'area riservata di un sito).

Quando un *client* richiede al *server* una pagina web statica (.html o .htm), vengono eseguiti i seguenti passaggi:

- il *client* richiede una pagina al Web Server (effettua una *request*);
- il Web Server invia la pagina web al *client* sotto forma di documento HTML (*response*);
- il *browser* del *client* interpreta e visualizza il documento HTML ricevuto.

Quando, invece, un *client* richiede al *server* una pagina web dinamica (.php), vengono eseguiti i seguenti passaggi (Fig. 1):

- il *client* (**1**) richiede una pagina al Web Server (*request*);
- l'interprete del *php* (*php engine*[1]) elabora il codice *php* interno alla pagina (**2**);
- se previsto dal codice *php*, il Web Server effettua una richiesta al database (**3**), il quale restituisce le informazioni al *php engine* (**4**);
- il *php engine* prima converte in codice HTML il codice sorgente *php* e i dati ottenuti dal database, poi invia l'output al Web Server (**5**);
- il Web Server invia la pagina al *client* (**6**) sotto forma di documento HTML (*response*).

Tipiche applicazioni degli script lato server sono le interrogazioni ai database remoti (ad esempio: motori di ricerca e forum). In genere, le applicazioni internet fanno uso di tre tipologie di pagine secondo l'architettura a tre livelli (*Three-Tier Architecture*). In questa architettura, il livello più basso è l'output che riceve l'utente finale (*front end*); il livello intermedio (*middleware*) è in genere costituito dalle pagine *php*, mentre il livello più alto (*back end*) è costituito dai database.

2 Il Linguaggio PHP

La sintassi del *php* prevede che l'inizio dello script inizi col tag <?*php* e termini col tag ? >. Quando il *php engine* incontra tali tag, esegue tutto il codice al loro interno. Ciò permette di integrare codice *php* all'interno del codice HTML. As esempio:

```
<?php
echo "<b> Hello World!</b>";
?>
```

L'istruzione *echo* visualizza a schermo la stringa posta tra virgolette (" "); al suo interno, come visto, può essere incluso anche del codice HTML (il tag $< b >$). La sintassi del *php* è *case sensitive* e prevede che ogni istruzione termini col punto e virgola (;). I commenti, come in *C*, si scrivono col doppio slash (//).

[1] Lo *Zend Engine*, o *php engine* è un codice open source usato come interprete del codice *php* (link).

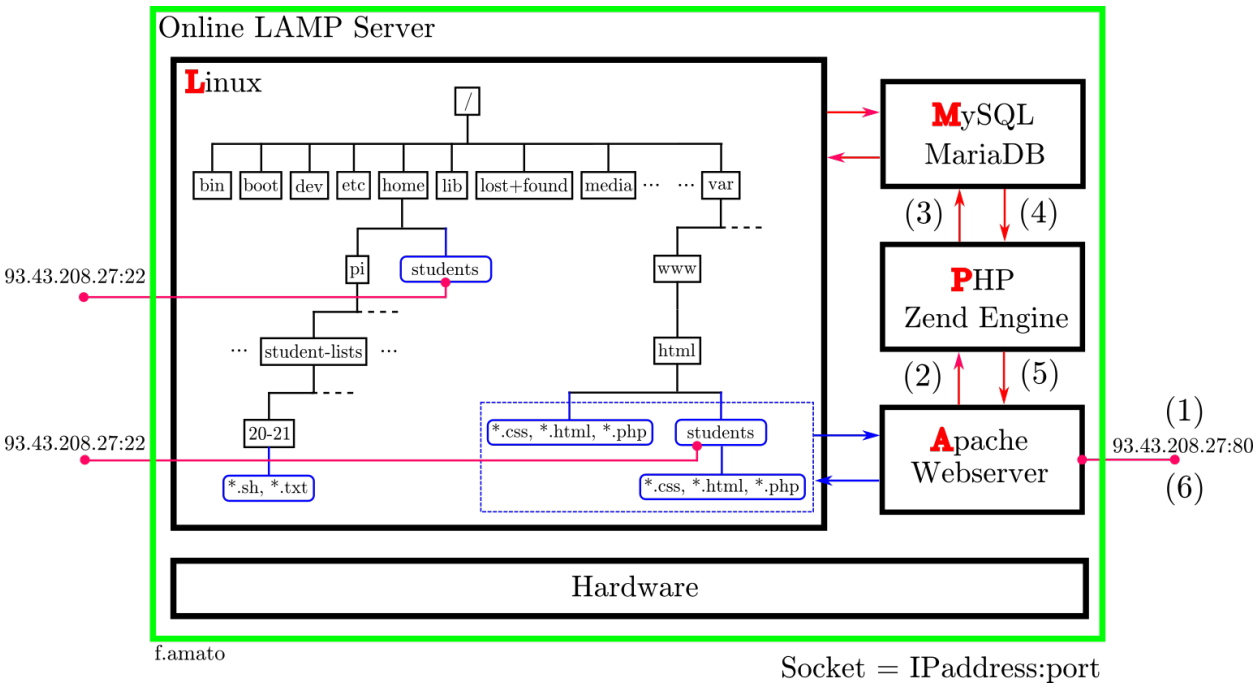

Figura 1: *Three-Tier Architecture* di un server LAMP costituito da **L**inux, **A**pache Web Server, **M**ySQL e **P**HP.

Funzione	Significato
is_int(espressione)	Verifica che la variabile sia di tipo intero
is_float(espressione)	Verifica che la variabile sia di tipo *float*
isset(espressione)	Verifica che la variabile sia inizializzata e non sia NULL
gettype(espressione)	Permette di conoscere il tipo di una variabile
settype(espressione, tipo)	Forza la variabile ad assumere il tipo indicato

Tabella 1: Alcuni esempi di funzioni disponibili in *php*.

2.1 Variabili

In *php*, le variabili hanno un nome che deve iniziare obbligatoriamente con il simbolo del dollaro ($) e sono *case-sensitive*. Le variabili non possiedono un tipo vero e proprio, il tipo viene definito in modo implicito in base al primo dato ad esse assegnato. Il tipo di una variabile, una volta assegnato, può essere convertito mediante *casting esplicito*. Di seguito, un esempio di assegnazione di variabile e *casting esplicito*:

```
$n = 25.54;
$numero = (int) $n; //casting esplicito, $n diventa un intero: 25
$n = 10;
$numero = (float) $n; //$n da intero diventa float: 10.0
```

2.2 Funzioni

Php permette di definire nuove funzioni o di utilizzarne di già esistenti. Alcune funzioni permettono di testare il tipo di una variabile (molto utile per l'inserimento dei dati in un form) come elencato in Tab. 1.

Funzione	Significato
$_GET	Contiene i campi HTTP ricevuti in GET
$_POST	Contiene i campi HTTP ricevuti in POST
$_FILES	Consente di effettuare l'upload di variabili
$_SERVER	Contiene informazioni riguardanti il server
$_COOKIE	Rappresenta i cookies HTTP
$_SESSION	Rappresenta le variabili di sessione

Tabella 2: Variabili d'ambiente in *php*.

2.3 Variabili d'ambiente

Le strutture dati predefinite disponibili in *php* sono chiamate variabili d'ambiente o *superglobals* che coinvolgono il server. In Tab. 2 ne sono riportate alcune. Un esempio di utilizzo delle variabili d'ambiente è rappresentato dal seguente codice:

```
<html>

    <head>
    </head>

    <body>
    Il tuo indirizzo IP: <?php echo $_SERVER['REMOTE_ADDR'] . "<br/>" ?>
    Il tuo browser web: <?php echo $_SERVER['HTTP_USER_AGENT'] . "<br/>" ?>
    Il tuo server web: <?php echo $_SERVER['SERVER_SOFTWARE'] . "<br/>" ?>
    Il tuo server <?php echo $_SERVER['SERVER_NAME'] . "<br/>" ?>
    </body>
</html>
```

L'output del codice è visionabile sul sito: 93.43.208.27/server.php.

2.4 Costanti

In *php* le costanti vanno sempre dichiarate con l'istruzione *define* e sono scritte in maiuscolo per differenziarle dalle variabili. Una costante è formata da un identificatore e da un valore:

```
define("PIGRECO", 3.14);
echo PIGRECO;
```

3 Quesiti di Approfondimento

- Spiega le differenze tra pagine web statiche e pagine web dinamiche.
- Cos'è una *response*? Da chi e a è viene inviata?
- Cos'è l'architettura a tre livelli (*Three-Tier Architecture*) ed in cosa consiste?
- A cosa serve il *casting*?
- Come si dichiara una variabile di tipo stringa?
- A cosa serve la variabile d'ambiente $_GET?
- Che valori restituisce la variabile d'ambiente $_SERVER['REMOTE_ADDR']?
- Che valori restituisce la variabile d'ambiente $_SERVER['REMOTE_NAME']?

4 Approfondimenti

- Camagni P., Nikolassy R., *"Database SQL & PHP"*, pagg. 270 - 282, Hoepli;