

UNIT 1

1.1 Concetti fondamentali sulle applicazioni Web

Il concetto base da assimilare prima di intraprendere qualsiasi sviluppo legato al Web è individuabile nella distinzione tra script server-side e script client-side.

A prescindere dai protocolli di ricerca del server (DNS) e di trasmissione dei dati (HTTP basato su TCP/IP) che verranno utilizzati per trasferire una pagina dal web server al client dell'utente è fondamentale capire come il contenuto della pagina venga generato e interpretato.

La programmazione lato server è dedicata proprio alla generazione del contenuto, sia esso HTML, XML, JPEG/GIF o qualsiasi altro formato che sarà successivamente inviato al client (sia esso un browser grafico come Internet Explorer o Mozilla oppure un client testuale come Linx o addirittura un comando che cattura l'output di una chiamata HTTP come wget).

Il flusso legato ad una richiesta del browser può essere delineato, semplificando, in questi passaggi:

- 1- il web server che riceve la richiesta cerca (fetch) all'interno del file system dell'host su cui gira, il file sorgente associato
- 2- in base alla configurazione del web server questo file può venire o meno processato da un interprete che ne effettuerà un parsing eseguendo ciò che riconoscerà essere un'istruzione
- 3- il contenuto ottenuto, assieme ad indicazioni sulla tipologia dello stesso, viene ritornato al browser
- 4- il browser legge il MIME type del file predisponendosi ad interpretarlo, eventualmente utilizzando plug-in associati al particolare tipo
- 5- il contenuto viene finalmente mostrato all'utente

PHP è un linguaggio che viene interpretato sul server e, nel caso sia stato installato come modulo, dal web server. Questo significa che l'utente non vedrà mai, se il file è stato correttamente interpretato, il codice sorgente ma piuttosto il risultato delle elaborazioni che sono state effettuate sul web server remoto, in generale mostrate tramite HTML.

D'altra parte linguaggi come Javascript sono interpretati direttamente dal client. Cioè il browser quando incontra dei tag (in genere del tipo `<script>`) si attiva per eseguire le istruzioni che vi sono incluse.

Può essere banale ma deve essere ben chiaro che le informazioni legate all'esecuzione di script client-side non sono utilizzabili dagli script server-side a meno che esse vengano esplicitamente inviate al server tramite una ulteriore chiamata HTTP.

1.2 Storia di PHP

Nel giugno 1995 Rasmus Lerdof annuncia su `comp.infosystem.www.authoring.cgi` il rilascio di un piccolo insieme di file binari scritti in C con il fine di registrare le visite alla pagina web contenente il suo curriculum...

A settembre viene rilasciato FI (Form Interpreter) un parser di pagine HTML con la possibilità di interagire con mSQL a fine 1995 PHP/FI è già utilizzato da diversi beta-tester e sviluppatori.

Nel Novembre 1997 esce la PHP/FI 2.0 che supporta mSQL, Postgres95 e MySQL, pare che fosse già usato da 50mila domini.

Il salto di qualità avviene nel giugno 1998 con PHP3 grazie a due israeliani Zeev Suraski e Andi Gutmans che hanno creato un parser chiamato Zend Engine (la versione 2 è inclusa in PHP 5). I punti di successo di PHP 3 furono il nuovo parser, il supporto di altri database, la compatibilità con Windows e soprattutto il numero crescente di sviluppatori che garantivano una continuità e una larga base di testing. PHP3 pare fosse installato sul 10% dei domini dell'epoca.

Nel maggio 2000 esce PHP 4, non più con licenza GPL ma PHP license (più restrittiva ma sempre open source) che supporta nativamente le sessioni e offre una modularità avanzata.

Ben 4 anni dopo viene rilasciato PHP 5 che tramite Zend Engine 2 introduce un nuovo modello di programmazione ad oggetti, interazione con file XML e il supporto di Web Services.

1.3 A cosa serve PHP

L'utilizzo più immediato di PHP rimane quello per cui è nato: la gestione dell'iterazione tra l'utente e un database.

Anche se ad un primo approccio questo può sembrare riduttivo, a detta dell'autore può rientrare in questa categoria almeno l'80% delle applicazioni web. Possono infatti rientrare in questa categoria sia un semplice forum o guestbook che un'applicazione avanzata ad esempio di internet banking o un motore di ricerca.

PHP offre sia gli strumenti necessari a manipolare nei modi più disparati l'input ricevuto, nel caso più classico da una form HTML sia funzioni native per il mantenimento dei dati in un database MySQL che la possibilità di utilizzare invece altri database OpenSource come PostgreSQL o mSQL oppure proprietari come Oracle, Sybase, MS SQL o altri anche tramite ODBC.

Una modalità di utilizzo più di nicchia ma che sta comunque prendendo piede anche grazie alla versatilità e alle funzioni native offerte dal linguaggio è lo scripting da console, cioè il realizzare script che verranno lanciati direttamente da riga di comando o tramite schedulatori come possono essere cron per ambienti UNIX-like oppure task scheduler o winat per ambienti Microsoft.

Infine esiste la possibilità, a dire il vero non molto sfruttata, di implementare applicazioni con interfacce grafiche (GUI) tramite il supporto fornito a GTK.

Come già accennato l'interprete PHP è disponibile sia per molti flavour UNIX che per Windows, ma anche per MacOS X. E' installabile come modulo su Apache e come CGI sui web server più diffusi da IIS a Netscape a iPlanet fino al piccolo Xitami.

Il linguaggio PHP offre funzioni per creare on-fly e modificare direttamente non solo file di tipo HTML o XHTML o XML ma anche immagini, file Acrobat (PDF) e addirittura filmati Flash. Sono inoltre presenti funzioni per interagire con i protocolli più comuni come LDAP, IMAP, NNTP, POP3, HTTP, FTP e COM Windows oppure per utilizzare classi Java esterne o ancora per utilizzare Espressioni Regolari o archivi (gzip).

1.4 Risorse online su PHP

Esistono sul web numerosi siti che trattano PHP e forniscono risorse, documentazione, informazioni, notizie.

www.php.net è il sito ufficiale del linguaggio PHP.

In <http://www.php.net/downloads.php> è possibile scaricare le versioni stabili e le nuove release del linguaggio oltre a Fix e Patch di sicurezza per le versioni più datate. Il sito fornisce documentazione dettagliata, completa e commentata dalla community. In <http://www.php.net/manual/it/> si trova un ottimo manuale online (in italiano).

Zend Technologies (www.zend.com) è la società, fondata dai principali autori di PHP, che ha sviluppato l'engine del PHP e propone vari prodotti commerciali per lo sviluppo, la gestione, l'ottimizzazione e l'encoding di codice PHP.

Esistono inoltre numerosi siti sul mondo php, i migliori sono linkati e divisi per categoria in <http://www.php.net/links.php>.

1.5 Introduzione alle variabili in PHP

Il linguaggio PHP ha una libertà di utilizzo delle variabili non comune nella maggior parte linguaggi di programmazione. Infatti non è necessaria al fine del corretto funzionamento dello script la dichiarazione della variabile stessa o la definizione del tipo di variabile.

E' però necessario che ogni variabile venga sempre preceduta dal carattere \$.

E' corretto scrivere in un qualsiasi punto dello script:

```
$pippoPluto = 20;
```

Tuttavia è permessa una dichiarazione tramite la sintassi:

```
var $pippoPluto;
```

anche con un assegnamento immediato al momento della dichiarazione (diretto, indiretto o come risultato di una funzione/espressione):

```
var $pippoPluto = 0;
```

```
var $pippoPluto2 = true;
```

```
var $pippoPluto3 = "hello, how are you?";
```

```
var $pippoPluto4 = explode($pippoPluto3,",");
```

```
var $pippoPluto5 = sqrt(2);
```

Per motivi di leggibilità è utile utilizzare la dichiarazione come in linguaggi più restrittivi.

Ogni variabile dichiarata o meno può in ogni momento cambiare tipo a seconda dell'assegnamento che viene effettuato, quindi se si era dichiarata la variabile pippoPluto come in precedenza:

```
var $pippoPluto = 0;
```

In una qualsiasi parte dello script si può effettuare un cambio di tipo semplicemente assegnando il nuovo dato (descritto in modo diretto, indiretto o come risultato di una funzione/espressione) alla variabile. Quindi sarà corretto assegnare alla variabile pippoPluto dichiarata in precedenza una stringa, un oggetto, un numero razionale, etc. anche come risultato di un operazione che utilizza il precedente valore della variabile:

```
var $pippoPluto = $pippoPluto + "ciao";
```

```
var $pippoPluto = new TObject;
```

```
var $pippoPluto = 1,34567;
```

```
var $pippoPluto = log(3)*sin(5);
```

1.6 Le costanti in PHP

In PHP è possibile definire delle costanti che non possono variare il valore.

Il motivo per cui si definisce una costante è di fatto legato a due motivi:

1. Se un valore assegnato non deve cambiare è corretto utilizzare una costante che agisce quindi su un'area di memoria dedicata a questo scopo
2. Una costante non potendo cambiare di valore durante l'esecuzione di uno script permette di essere certi di evitare che per sbaglio avvenga una variazione di valore che possa comportare errori e/o problemi

Le costanti in PHP sono case-sensitive.

Una costante si definisce come segue:

```
bool define ( string name, mixed value [, bool case_insensitive] )
```

Il che tradotto significa che la funzione define ritorna un valore true o false a seconda della buona riuscita della definizione della costante.

Il primo argomento della funzione è il nome assegnato alla costante, il secondo è il valore assegnato e il terzo (opzionale) è la forzatura della costante nell'essere case-insensitive. Se non assegnato il terzo argomento la costante sarà come di default, case-sensitive.

Ad esempio se definiamo la costante:

```
define ("test", "valore1");
```

La costante sarà richiamabile esclusivamente come test scritto interamente in minuscolo.

Notiamo subito che per le costanti non è definito il simbolo del dollaro \$ che indica la condizione di variabile.

Per maggiori informazioni:

<http://it.php.net/manual/it/function.define.php>

Approfondimenti:

<http://it.php.net/manual/it/language.constants.php>

<http://it.php.net/manual/it/function.defined.php>

<http://it.php.net/manual/it/function.constant.php>

1.7 Operatori nel linguaggio PHP

Di seguito vedremo i diversi operatori utilizzabili nel linguaggio PHP.

ARITMETICA BASE

`$a + $b` esegue la somma di `$a` e `$b`.

`$a - $b` esegue la differenza di `$a` e `$b`.

`$a * $b` esegue il prodotto di `$a` e `$b`.

`$a / $b` esegue il quoziente di `$a` e `$b`.

`$a % $b` esegue `$a` diviso da `$b` e restituisce il resto.

CONFRONTARE DUE VARIABILI / VALORI

`$a == $b` Uguale. Restituisce TRUE (vero) se `$a` è uguale a `$b`.

`$a === $b` Identico. Restituisce TRUE se `$a` è uguale a `$b` e se sono dello stesso tipo.

`$a != $b` Diversi. Restituisce TRUE se `$a` è diverso da `$b`.

`$a $b` Diversi. Restituisce TRUE se `$a` è diverso da `$b`.

`$a !== $b` Non identici. Restituisce TRUE se `$a` è diverso da `$b`, o se non sono dello stesso tipo.

`$a < $b` Minore. Restituisce TRUE se `$a` è strettamente minore di `$b`.

`$a > $b` Maggiore. Restituisce TRUE se `$a` è strettamente maggiore di `$b`.

`$a <= $b` Minore o uguale. Restituisce TRUE se `$a` è minore o uguale a `$b`.

`$a >= $b` Maggiore o uguale. Restituisce TRUE se `$a` è maggiore o uguale a `$b`

OPERATORI D'ESECUZIONE

PHP supporta un operatore di esecuzione: backticks (```). PHP cercherà di eseguire il contenuto dei backticks come comando di shell; sarà restituito l'output, che potrà anche essere assegnato ad una variabile.

```
$output = `ps xa`;
```

esegue il comando `ps xa` e assegna l'output alla variabile `$output`

```
echo "<.pre>$output";
```

visualizza il contenuto formattato (con spazi e a capo) della variabile `$output`, utilizzando il tag PRE

ATTENZIONE: i backticks non sono apostrofi!

OPERATORI D'INCREMENTO

Operatori di pre e post incremento, come nel linguaggio C.

`++$a` Pre-incremento. Prima incrementa `$a` di una unità e poi restituisce `$a`.

`$a++` Post-incremento. Prima restituisce `$a` e poi incrementa `$a` di una unità.

`--$a` Pre-decremento. Prima decrementa `$a` di una unità e poi restituisce `$a`.

`$a--` Post-decremento. Prima restituisce `$a` e poi decrementa `$a` di una unità.

OPERATORI LOGICI

`$a and $b` And. Restituisce TRUE se sia `$a` che `$b` sono TRUE. Al posto di `and` si può anche usare `&&`.

`$a or $b` Or. Restituisce TRUE se uno dei due operatori è TRUE. Al posto di `or` si può anche usare `||`.

`$a xor $b` Xor. Restituisce TRUE se uno dei due operatori è TRUE, ma non entrambi.

`!$a` Not. Restituisce TRUE se `$a` non è TRUE.

OPERATORI DI STRINGA

Ci sono due operatori di stringa: l'operatore di concatenazione e quello di assegnazione concatenata.

L'operatore di concatenazione ('.') restituisce la concatenazione dei suoi argomenti a destra e a sinistra.

```
$a = "Ciao ";
```

```
$b = $a . "a tutti!";
```

Ora la variabile \$b contiene il valore "Ciao a tutti"

L'operatore di assegnazione concatenata ('.=') aggiunge alla fine dell'argomento sul lato destro l'argomento sul lato sinistro.

```
$a = "Ciao ";
```

```
$a .= "a tutti";
```

Ora la variabile \$a contiene il valore "Ciao a tutti"

OPERATORI DI CONTROLLO ERRORE

Il carattere @ (at) anteposto ad un espressione permette di ignorare l'errore che potrebbe essere generato nel caso in cui l'espressione non sia corretta.

/*in questo caso se non avessimo messo "@" prima del comando, sarebbe stato segnalato l'errore.*/

```
$testo = "testo della query";
```

```
$query = @mysql_query($testo);
```

Se nel file di configurazione "php.ini" la caratteristica track_errors è abilitata, quindi a 1, ogni messaggio di errore generato sarà salvato nella variabile globale \$php_errormsg, sovrascrivendosi ogni volta.

1.8 Interazione tra PHP e HTML

Il linguaggio di programmazione PHP interagisce in modo diretto con l'HTML.

Per capire meglio vediamo il seguiamo i seguenti passaggi.

1. Il client del navigatore, ovvero il web browser richiede una pagina PHP test.php
2. Il server web Apache (solitamente) in ascolto riceve la richiesta e inizia a processarla
3. Nella pagina è presente codice PHP che viene eseguito e processato da Apache mediante il modulo di PHP installato su di esso.
4. Il codice che viene prodotto e inviato al web browser è codice HTML (o eventualmente altri linguaggi interpretati lato client come XML, javascript etc...)
5. Non una riga di codice PHP viene inviata al browser
6. Il browser processa in codice ricevuto e lo visualizza al navigatore
7. Il tutto avviene mediante il protocollo HTTP

Approfondiremo in seguito il concetto di client/server e di linguaggio di scripting server-side, in ogni caso prendiamo atto che PHP normalmente interagisce con il linguaggio HTML per la visualizzazione dei contenuti e per ricevere i dati dal browser dell'utente.

Tale situazione è tipica e assolutamente comune per tutte le web application.

1.9 Interazione tra PHP e Database

Allo stesso modo con cui PHP interagisce con il linguaggio HTML lo fa anche con i database.

In pratica come si può ben notare dal file di configurazione di php, php.ini, il linguaggio PHP interagisce con i più svariati database per archiviare, memorizzare e prelevare i dati in esso archiviati.

Esistono per ogni database server delle specifiche funzioni di connessione, invio query, recupero dati etc...

Il linguaggio PHP nasce per lavorare prevalentemente con il database server MySQL ma è in grado di lavorare egregiamente con una serie di database di cui vediamo una rapida lista:

- Interbase
- SQL Server di Microsoft
- PostgreSQL
- Oracle
- Sqlite
- Sybase
- DB2
- etc...

Questo ci permette di poter utilizzare il linguaggio PHP con qualunque database sul mercato (quasi) e di poter costruire quindi applicazioni dinamiche basate oltre che sui dati inviati dall'utente anche su dati memorizzati nei più disparati database e

aggiornare gli stessi.

1.10 Creazione di una semplice pagina PHP

La creazione di una pagina PHP avviene in modo molto semplice.
Per prima cosa creare un nuovo file all'interno della document root (la radice) del server web apache.

Ad esempio il file esempio.php

Il codice PHP perché venga interpretato dal modulo PHP installato sul server web è necessario che sia racchiuso tra i tag:

```
<?php  
?>
```

E' altresì possibile che l'apertura del tag avvenga in forma breve: <?

Tale soluzione è sconsigliata in quanto prevede che siano accettati a livello di configurazione di php.ini la possibilità di avere tag abbreviati.

Maggiori informazioni su php.ini:

<http://it.php.net/manual/it/ini.core.php>

Nel caso si lavori su server dedicato o di propria gestione non vi è alcun problema, ma in caso di hosting shared è meglio evitare qualunque problematica inutile direttamente legata alla configurazione del server web e di php.

All'interno di questa pagina andiamo ad esempio ad assegnare un valore ad una variabile e a stamparne a video il suo contenuto:

```
<?php  
$a = "Questo è il mio primo script";  
echo $a;  
?>
```

Come possiamo notare, oltre alla funzione echo (che stampa a video) è presente un ";" come ogni terminazione di riga.

Ovviamente in questo caso si è deciso di non dichiarare la variabile \$a ma di utilizzarla direttamente nel codice.

In caso di script o progetti complessi è tuttavia suggerita la dichiarazione delle variabili con adeguati commenti sull'utilizzo che si andrà a fare di tale variabile.

Richiamando dal proprio browser ad esempio:

<http://localhost/esempio.php>

Otterremo il seguente output video:

Questo è il mio primo script

Nota sull'estensione dei files

Gli script PHP normalmente per uso e consuetudine hanno estensione .php, è tuttavia possibile che vi siano altre possibilità di nominare i files.

Le estensioni più tipiche sono: .php3, .php4, .phtml, .inc etc...

L'estensione è totalmente trasparente, purché il server web sia configurato per usare

il modulo di PHP per quella specifica estensione utilizzata.