Verso l'architettura MVC-2 PHP: Hypertext Preprocessor

ALBERTO BELUSSI

ANNO ACCADEMICO 2012/2013

PHP per generare pagine web dinamiche

Anche uno *script PHP* può essere visto come uno "schema di pagina Web" dove:

- le parti statiche sono scritte in HTML e
- le parti dinamiche sono generate attraverso porzioni di codice PHP.

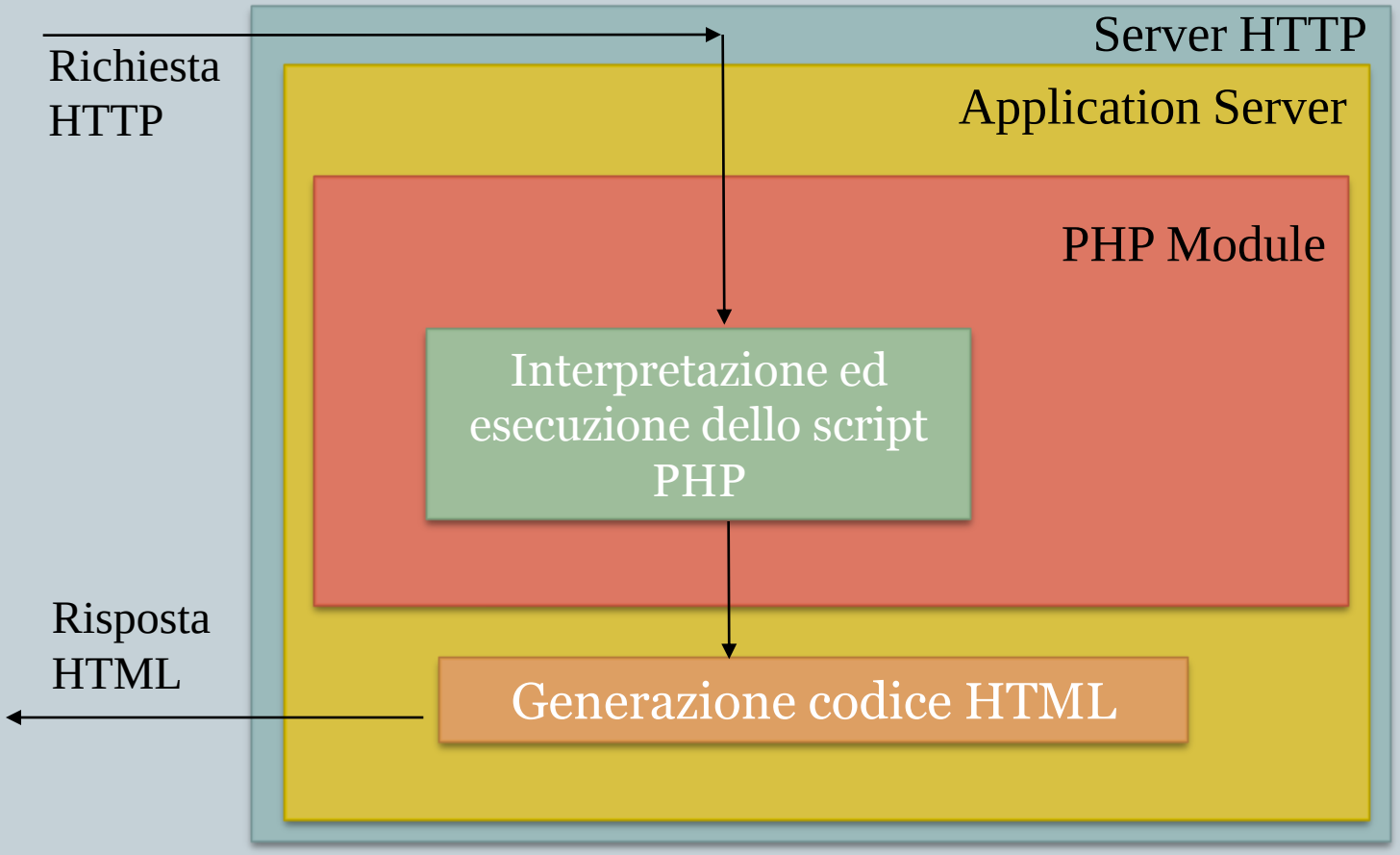
Gli script PHP vengono "gestiti" da un componente operante sul web server (*PHP module)*. Questo componente interpreta ed esegue gli script PHP.

Riferimento:
http://www.php.net/

Funzionamento PHP (1)

- Codice sorgente PHP: è scritto dal programmatore dell'applicazione web, con la collaborazione di altri esperti per la parte di presentazione (grafici, …) .
- Si tratta di un file con estensione .php contenente:
  - **Codice HTML**: parti statiche e grafica
  - **Scripting PHP**: parti dinamiche

Funzionamento PHP (2)



Sintassi PHP

Uno script PHP può essere visto come un documento HTML esteso con alcuni marcatori speciali per "immergere" codice nell'HTML.

Il macatore speciale per le porzioni di PHP è:

<?php ... ?>
<script languge="php"> .... </script>

forme brevi (solo se abilitate):
<? ... ?> per **<?php ... ?>**
<?= ... ?> per **<?php echo "..."; ?>**

Caratteristiche base del linguaggio

Variabili semplici

TIPI BASE: boolean (TRUE, FALSE), integer (intero con segno: 453, -22), float (numeri in virgola mobile, 1.234, 2E-10) e string (qualsiasi stringa di carattere: “hello”, ‘world’)

NOMI VARIABILI: iniziano sempre con il carattere $ e sono case sensitive

TIPO della VARIABILE: viene definito nel primo assegnamento; **non ci sono dichiarazioni di tipo**.

ASSEGNAMENTO: $a = TRUE; $B = “pippo”, $vInt = 3, $vFloat = 0.343

```
<?
$a_bool = TRUE;   // a boolean
$a_str  = "foo";  // a string
$a_str2 = 'foo';  // a string
$an_int = 12;     // an integer
?>
<p>
a_bool is of type <?=gettype($a_bool);  ?></br>
a_str is of type <?=gettype($a_str); ?>
</p>
```

```
// If this is an integer, increment it by four
<?
if (is_int($an_int)) {
    $an_int += 4; echo “Integer: $an_int”;
}
// If $a_bool is a string, print it out else does not print out anything
if (is_string($a_bool)) {
    echo "String: $a_bool";
}
?>
```

Caratteristiche base del linguaggio

Variabili di tipo array

ASSEGNAMENTO: **$arr = array();** // array vuoto

$arr = array(*key* => *value*, *key2* => *value2*, *key3* => *value3*, …);

$arr1 = array(1 => "a", “pippo" => "b“, “b" => “pippo“,
);

$arr2 = [“alberto" => 3, “roberto" => 1,
];

ACCESSO AI VALORI: **$arr [*KEY*]**

$v = $arr1[1]; // viene assegnata la stringa “a” alla variabile $v

$w = $arr2[“alberto”]; // viene assegnato l’intero 3 alla variabile $w

AGGIUNTA e ELIMINAZIONE di VALORI: **$arr [*KEY*] = *VALUE*, *unset*($arr[*KEY*])**

$arr2[“ugo”] = 10; // viene aggiunto un elemento all’array $arr2 “ugo” => 10

unset($arr2[“alberto”]) // viene cancellato l’elemento di chiave “alberto”

unset($arr2) // viene cancellato tutto l’array

Caratteristiche base del linguaggio

Controllo di flusso e cicli

IF .... ELSE ... :

if (*condizione*) *istruzione*;

```
if ($a > $b)  echo "a è maggiore di b";
```

if (*condizione*) { *istruzione1*; ... *istruzioneN*; }

```
if ($a > $b)  {
        echo "a è maggiore di b";
        $c = $a;
}
```

if (*condizione*) { *istr1*; ... *istrN*; } else { *istr1*; ... *istrM*; }

```
if ($a > $b) {
        echo "a è maggiore di b";
} else {
        echo "a non è maggiore di b";
}
```

if (*cond1*) { *istr1*; ... *istrN*; } elseif (cond2) { *istr1*; ... *istrM*; } else { ... }

Caratteristiche base del linguaggio

Controllo di flusso e cicli

WHILE … : le istruzioni del corpo {…} vengono ripetute mentre la condizione *cond* risulta vera. Quando *cond* diventa falsa il ciclo termina.

while(*cond*) { *istruzione1*; … *istruzioneN*; }

```
$i = 1;
while ($i <= 10) {
        echo $i++;
}
```

FOR … : le istruzioni del corpo {…} vengono ripetute mentre la condizione *cond* risulta vera. Quando *cond* diventa falsa il ciclo termina.

for(istr1; *cond; istr2*) { *istruzione1*; … *istruzioneN*; }

```
$i = 1;
for ($i = 1; $i <= 10; $i++) {
        echo $i;
}
```

Caratteristiche base del linguaggio

Controllo di flusso e cicli

FOREACH … : le istruzioni del corpo {…} vengono ripetute su tutti gli elementi dell'array *array1*.

foreach(*array1* as $value) { *istruzione1*; … *istruzioneN*; }

```
$arr = array(10, 20, 30, 40);
foreach ($arr as $value) {
        echo "valore: $value";
}
```

oppure

foreach(*array1* as *$key* => $value) { *istruzione1*; … *istruzioneN*; }

```
$arr = array("a"=>10, "b" => 20, "f" => 30, "g" => 40);
foreach ($arr as $key => $value) {
        echo "chiave: $key => valore: $value";
}
```

Caratteristiche base del linguaggio

Commenti

- // ...
- /* ... */
- # ...

Inclusioni

- **require(filepath)**: include il file indicato
- **require_one(filepath)**: include il file indicato se non è già stato incluso

Caratteristiche base del linguaggio

Funzioni

```
function Fname ($arg1, $arg2, /* ..., */ $argn) {
        echo "Esempio di funzione.\n";
        echo “Argomenti: $arg1, $arg2, ..., $argn”;
        $retval = $arg1 + $arg2 + ... + $argn;
        return $retval;
}
```

Senza return la funzione ritorna NULL.

Classi e oggetti in PHP

Classi e oggetti

Anche in PHP è possibile definire classi e istanziarle creando oggetti.

Definizione di una classe:

```
<?php
class SimpleClass {
        // property declaration
        public $var = 'a default value';
        // method declaration
        public function displayVar() {
                echo $this->var;
        }
}
?>
```

Istanziazione:

```
<?php
        $instance = new SimpleClass();
        // This can also be done with a variable:
        $className = 'SimpleClass';
        $instance = new $className(); // SimpleClass()
        echo $instance->var;
        echo $instance->displayVar();
?>
```

Interazione con un DBMS da PHP

Libreria per l'interazione con database postgres.
(http://it2.php.net/manual/en/book.pgsql.php)

Esempio


```
<?php
        // Connecting, selecting database
        $dbconn = pg_connect("host=localhost dbname=publishing user=www password=foo")
        or die('Could not connect: ' . pg_last_error());
        // Performing SQL query
        $query = 'SELECT * FROM authors';
        $result = pg_query($query) or die('Query failed: ' . pg_last_error());
        // Printing results in HTML
?>
        <html>
        <head><title>Estrazione dati dalla tabella authors</title></head>
        <body>
        <table>
<?      while ($line = pg_fetch_array($result, null, PGSQL_ASSOC)) { ?>
                <tr>
<?              foreach ($line as $col_value) { ?>
                        <td><?=$col_value ?></td>
<?              } ?>
                </tr>
<?      } ?>
        </table>
        </body>
        </html>

<?      // Free resultset
        pg_free_result($result);
        // Closing connection
        pg_close($dbconn);
?>
```

Apertura di una connessione

pg_connect(*connectionString****)***

opens a connection to a PostgreSQL database specified by the ***connectionString***.

Esempio

```
$dbconn = pg_connect("host=dbserver.scienze.univr.it
                      dbname=did2013
                      port=5432
                      user=userlabXX  password=YY")
```

Restituisce in caso di successo una *connection resource* per PostgreSQL, **FALSE** se fallisce.

Esecuzione di una interrogazione

pg_query(*connectionRes, queryString****)***

executes the ***queryString*** on the specified database ***connectionRes***.

Esempio

```
$result = pg_query($conn, "SELECT author, email FROM authors");
if (!$result) {
        echo "An error occurred.\n";
        exit;
}
```

Restituisce in caso di successo una *query result resource* (è simile ad un resultSet di JDBC), **FALSE** se fallisce.

Elaborazione del risultato di una interrogazione

pg_fetch_array(*queryResultRes, [row, resultType])*

returns an array that corresponds to the fetched row (record), belonging to the ***queryResultRes***. **FALSE** is returned if ***row*** exceeds the number of rows in the set, there are no more rows, or on any other error.

row indica la riga da restituire (se non precisato o null, viene restituita la riga successiva).

resultType indica il tipo indice usato nell'array restituito. **PGSQL_ASSOC** (indice associativo), **PGSQL_NUM** (indice numerico) e **PGSQL_BOTH** (entrambi, è il default).

Esempio

```
$result = pg_query($conn, "SELECT author, email FROM authors");
$arr = pg_fetch_array($result);
echo $arr["author"] . " <- Row 1 Author\n";
echo $arr["email"] . " <- Row 1 E-mail\n";
```

Eliminazione del risultato dell'interrogazione

pg_free_result(*queryResultRes****)***

frees the memory and data associated with the ***queryResultRes***.

Esempio

```
$result = pg_query($conn, "SELECT author, email FROM authors");
...
pg_free_result($result)
```

Chiusura della connessione

pg_close(*connectionRes****)***

Closes a connection to a PostgreSQL database specified by the ***connectionRes***.

Esempio

pg_close($dbconn)

Esecuzione di interrogazioni parametriche

Funzioni alternative a ***pg_query*** per l'esecuzione di interrogazioni parametriche.

pg_prepare(*connectionRes, queryName, queryString*);

$result = **pg_execute**(*connectionRes, queryName, arrayOfParameterValues*);

Esecuzione di una interrogazione parametrica

pg_prepare(*connectionRes****, queryName, queryString)***

creates a prepared statement ***queryString*** for later execution with pg_execute(). This feature allows commands that will be used repeatedly to be parsed and planned just once, rather than each time they are executed. ***queryName*** is the name of the prepared command.

Esempio

pg_prepare($conn, ”myQuery”, "SELECT author, email FROM authors WHERE id = $1");

$result = pg_execute ($conn, ”myQuery”, array(189));

…

Esecuzione di una interrogazione parametrica

pg_execute(*connectionRes****, queryName, arrayOfParam)***

Sends a request to execute a prepared statement with given parameters, and waits for the result. The values contained in ***arrayOfParam*** replace the parameter $1, $2, …. in the prepared parametric query. The replacement follows the array order.

Esempio

pg_prepare($conn, ”myQuery”, "SELECT author, email FROM authors WHERE id = $1");

$result_1 = **pg_execute** ($conn, ”myQuery”, array(189));

$result_2= **pg_execute** ($conn, ”myQuery”, array(17));

…

Architettura applicazione PHP

- Una classe PHP organizzata come un Bean per ogni risultato di interrogazione da gestire.
  - Tante variabili private per rappresentare i dati
  - Tanti metodi getter e setter come per i Java Bean
- Una classe PHP per l'interazione con il DBMS, contiene:
  - Tutte le query SQL come stringhe eventualmente con parametri ($1, $2, …)
  - Una funzione getXXX per ogni interrogazione parametrica che restituisce array di oggetti delle classi Bean
- Un file PHP per ogni schema di pagina da gestire nell'applicazione