Linguaggio PHP

PROF. DIOMAIUTA CRESCENZO

PHP

- PHP è un linguaggio di scripting general-purpose open-source molto diffuso, specialmente indicato per lo sviluppo Web.
- PHP può essere usato su tutti i principali sistemi operativi, inclusi Linux, molte varianti di Unix (compresi HP-UX, Solaris e OpenBSD), Microsoft Windows, MacOS X ed è supportato dalla maggior parte dei web server esistenti.
- PHP viene eseguito (interpretato) da un server web) è un linguaggio di scripting server-side

PHP

- E' un linguaggio di scripting server side
- Quando un browser client effettua una richiesta ad un server Web predisposto per il PHP, il server:
  1. Legge ed individua la pagina sul server.
  2. Esegue le istruzioni PHP contenute all'interno della pagina ed esegue tutti i comandi PHP.
  3. Rimanda la pagina risultante al Browser.

PHP: Vantaggi

- PHP è un linguaggio semplice da utilizzare, ha una sintassi derivata da linguaggi come C/C++, Perl, Java.
- PHP possiede una gestione dei database semplice, infatti, con poche righe di codice è possibile accedere qualsiasi database, estrapolare i dati che ci interessano e inserirli nella pagina Web.
- PHP è OpenSource.
- PHP gira sui principali Web server (elevata portabilità).

PHP: Esempio

- Per distinguere all'interno del codice il linguaggio da interpretare ed eseguire (PHP) dall'HTML si utilizzano dei TAG particolari
- **<?php: Apertura dello script PHP**
- **?>: Chiusura dello script PHP**

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

- Il file non necessita di essere eseguibile, possiede l'estensione **.php**.
- Tutto ciò che fa è visualizzare 'Hello World!' usando la funzione echo di PHP inserita fra gli speciali tag <?php e ?>
- All'interno di un file HTML si può entrare ed uscire dalla modalità PHP quante volte si desidera.

PHP: Esempio

- Al client arriverà il seguente file html:

```
<html>
    <body>
        <h1>Hello World!! </h1>
    </body>
</html>
```

PHP: Uscire dalla modalità HTML

Il PHP permette l'uso delle strutture seguenti, dove l'output e condizionato dal valore di $expression:

```
<?php
    if ($expression) {
?>
        <strong>Condizione vera.</strong>
<?php }
        else {
?>
        <strong>Condizione falsa.</strong>
<?php } ?>
```

PHP: Variabili

- Ogni variabile ha un nome e un valore.
- Il nome di una variabile in PHP:
  - è sempre prefissato dal simbolo del dollaro (**$**)
  - contiene caratteri alfanumerici e inizia con una lettera o con il simbolo "_"
- In PHP, una variabile viene creata automaticamente ogni volta che le si assegna un valore (***dichiarazione implicita***)
- Esempio:
  - $a = 15;
    - Viene creata una nuova variabile chiamata $a
    - Le viene assegnato il valore ''15''

PHP: Tipi di dati

- I tipi di dato supportati da PHP sono:
  - tipi scalari:
    - numeri (interi e a virgola mobile)
    - stringhe
    - booleani
  - tipi compositi:
    - array
    - oggetti

PHP: Tipi di dati

Assegnazione di variabili:

- numeri: `$a = 7; $pi_greco = 3.14;`
- stringhe: `$a = "Ciao!";`
- booleani: `$check = TRUE; $check = FALSE;`
- array (esplicito):

```
$giorni = array("lun", "mar", "mer", "gio", "ven", "sab", "dom");
```

- array (implicito):

```
$giorni[0] = "lun";
$giorni[1] = "mar";
…
$giorni[6] = "dom";
```

PHP: Tipi di dati

- Il tipo di una variabile di solito non è impostato dal programmatore ma deciso a runtime da PHP in base al contesto nel quale la variabile viene utilizzata.
- Per controllare il tipo e i valori di una certa variabile si usa la funzione di PHP ***var_dump***().

PHP: Stringhe

- Le stringhe in PHP non pongono limiti nella lunghezza.
- Definite usando l'apice singolo: in questo caso la stringa non è mai interpretata.
  - echo 'questa è una semplice stringa';
- Definite usando le doppie virgolette: in questo caso PHP interpreta le variabili e i caratteri di escape contenuti nella stringa.
  - echo "stampo la variabile \$var: $var";
- Operatori di stringa:
  - l'operatore '**.**' Concatena due stringhe
  - l'operatore '**.=**' appende ad una variabile la stringa a destra dell'uguale.
  - Il confronto tra stringhe si esegue utilizzando gli operatori ==, !=, <, >, <=, >=

PHP: Array

- PHP fornisce array associativi per memorizzare coppie chiave-valore. Non ci sono differenze fra array con indici e array associativi: PHP fornisce un solo tipo di array.
- Un array può essere creato usando il costrutto array:
  - $vet = array("foo" => "bar", 13 => true);
  - echo $vet["foo"]; // bar
  - echo $vet[13]; // 1
- La chiave può essere sia un intero che una stringa.
- Un elemento di un array può essere di uno qualunque dei tipi PHP
  - $vet = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));
  - echo $vet["somearray"][6]; // 5
  - echo $vet["somearray"][13]; // 9
  - echo $vet["somearray"]["a"]; // 42
- $vet["x"] = 32; // Aggiunge un nuovo elemento
- unset($vet[6]); // Rimuove l'elemento dall'array
- unset($vet); // Cancella tutti gli elementi

PHP: Strutture di controllo

- Costrutto if.. else

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

- elseif estende if aggiungendo la possibilità di eseguire un'altra istruzione nel caso in cui l'espressione contenuta nel ramo if sia FALSE

```
if ($a > $b) {
    print "a è maggiore di b";
} elseif ($a == $b) {
    print "a è uguale a b";
} else {
    print "a è minore di b";
}
```

PHP: Strutture di controllo

- Costrutto **switch**

```
switch ($i) {
    case 0: print "i è uguale a 0"; break;
    case 1: print "i è uguale a 1"; break;
    case 2: print "i è uguale a 2"; break;
}
```

PHP: Strutture iterative

- Costrutto **while**

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

- Costrutto **do..while**

```php
$i = 0;
do {
    print $i;
}
while ($i>0);
```

- Costrutto **for**

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

PHP: Strutture iterative

- Costrutto **for..each**

```
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
    print "Valore corrente di \$a: $v.\n";
}
```

- **break** termina l'esecuzione di una struttura for, foreach, while, do..while o switch.
- **continue** si utilizza per interrompere l'esecuzione del ciclo corrente e continuare con l'esecuzione all'inizio del ciclo successivo

PHP: Include e require

- **include()** e **require()** includono e valutano uno specifico file.
- Queste due funzioni sono identiche eccetto per come esse trattano gli errori:
  - include() produce un warning
  - require() restituisce un Fatal Error.

CONSIGLIO: Se ho parte di connessione al database con parametri crea file di config a parte. Inserisci stringa in quel file.

PHP: Funzioni

Una funzione può essere definita usando la sintassi

```
function myfunc($arg_1, $arg_2, ..., $arg_n)
{
    echo "Funzione di esempio.\n";
    return $retval;
}
```

- PHP non supporta l'overloading di funzioni
- PHP supporta il passaggio di argomenti per valore e per riferimento (di default, gli argomenti della funzione sono passati per valore)
  - Il passaggio per riferimento si ottiene anteponendo un ampersand **(&)** al nome dell'argomento nella definizione della funzione
- I valori vengono restituiti usando l'istruzione opzionale **return**.
- Può essere restituito qualsiasi tipo, incluse liste ed oggetti.

PHP: Classi e Oggetti

- Le classi sono tipi del linguaggio.
- Una classe si definisce usando la seguente sintassi:

```
class Carrello{
    var $items;  // Articoli nel carrello
    // lo uso come array associativo
    // Aggiunge $num articoli di $artnr nel carrello
    function add_item ($artnr, $num) {
        $this->items[$artnr] += $num;
    }
}
```

PHP: Classi e Oggetti

- Per creare una variabile oggetto si usa l'operatore '**new**'.

```
$carrello = new Carrello;
```

- Si può accedere ai metodi della classe usando l'operatore '**->**'

```
$carrello->add_item("Pere",15 );
```

Questa operazione aggiunge all'array associativo items la nuova coppia (Pere,10).

- Per stampare i valori dell'array si può usare ancora l'operare **->**

```
echo $carrello->items["Pere"];     //stampa 15
```

si specifica un solo simbolo di **$** per accedere alla variabili

- Per poter accedere all'interno della classe alle funzioni e alle variabili interne della stessa classe si usa la pseudo-variabile '**$this**'

NOTA: se uso metodo o attributo il dollaro va messo solo al nome dell'oggetto *
Dove è funzione che fa morire il programma in un punto.

PHP: Classi e Oggetti

- PHP supporta l'ereditarietà
- Una classe estesa o derivata ha tutte le variabili e le funzioni della classe di base più tutto ciò che viene aggiunto dall'estensione.
- Non è possibile che una sottoclasse ridefinisca variabili e funzioni di una classe madre.
- L'eredità multipla non è supportata
- Le classi si estendono usando la parola chiave **'extends'**.

```
class Named_Carrello extends Carrello{
    var $owner;
    function set_owner($name){
        $this->owner = $name;
    }
}
```

PHP: Classi e Oggetti

- In PHP si possono definire i costruttori di classe che vengono invocati automaticamente quando viene istanziato un oggetto con l'operatore new.

```
class Auto_Carrello extends Carrello {
    function Auto_Carrello() {
        $this->add_item ("15", 1);
    }
}
```

- L'operatore :: è usato per riferirsi alle funzioni di classi senza istanziarle. Si possono usare funzioni della classe, ma non le variabili della classe.

PHP: Variabili speciali

PHP definisce un certo numero di array associativi speciali disponibili all'interno degli script server-side tra le quali:

- **$_SERVER**: contiene variabili impostate dal web server e relative all'ambiente di esecuzione dello script
- **$_GET**: contiene variabili ricevute dallo script via HTTP GET
- **$_POST**: contiene variabili ricevute dallo script via HTTP POST
- **$_COOKIE**: contiene variabili ricevute dallo script tramite l'invio di cookie
- **$_ENV**: contiene variabili d'ambiente dello script.
- **$_SESSION**: contiene variabili che sono correntemente registrate nella sessione di esecuzione dello script.

PHP: Variabili speciali - Esempio

- Verifichiamo che tipo di browser sta utilizzando chi visita un sito web. Per fare questo si controlla la stringa dell'user agent che il browser invia come parte della richiesta HTTP. L'informazione viene registrata in una variabile.
- La variabile alla quale ci riferiamo adesso è `$_SERVER["HTTP_USER_AGENT"]`
  - `<?php echo $_SERVER["HTTP_USER_AGENT"]; ?>`
- L'output di questo script potrebbe essere:

```
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
```

- `$_SERVER` è soltanto un array che automaticamente viene reso disponibile da PHP.

PHP: Gestione di form

- Qualsiasi elemento inviato tramite una form HTML è automaticamente disponibile negli script PHP.
- Es:

```
<form action="action.php" method="POST">
    Nome: <input type="text" name="nome" value="" />
    Età: <input type="text" name="eta" value ="" />
    <input type="submit">
</form>
```

- Quando l'utente riempie questa form, premendo il pulsante submit, viene richiamata la pagina ***action.php***.

PHP: Gestione di form

- Nella pagina ***action.php*** uno script di esempio di gestione della form precedente potrebbe essere:

```
Ciao <?php echo $_POST["nome"]; ?>.
La tua età è <?php echo $_POST["eta"]; ?> anni.
```

- Ecco un possibile output di questo script:

```
Ciao SWBD.
La tua età è 25 anni.
```

- Le variabili $_POST["nome"] e $_POST["eta"] vengono impostate automaticamente dal PHP.
- Se usassimo il metodo GET le informazioni ricavate dalla nostra form si troverebbero invece in **$_GET**.

PHP: Gestione di sessioni

- PHP fa uso delle sessioni per conservare alcune informazioni attraverso accessi successivi.
- Il sistema di gestione delle sessioni supporta un numero di opzioni di configurazione che possono essere impostate nel file di configurazione ***php.ini***.
- Al visitatore del sito web viene assegnato un id unico, il cossidetto id di sessione. Questo viene registrato in un cookie lato cliento è propagato tramite l'URL. Quando un visitatore accede al sito, PHP controllerà automaticamente (se **session.auto_start** è settato a 1 nel file php.ini) se uno specifico id di sessione sia stato inviato con la richiesta.
- Le sessioni sono gestite un 2 modi
  - **session_register():** che registra variabili nella sessione corrente.
  - **$_SESSION:** Si può accedere alle variabili di sessione come a variabili normali in un array.

PHP: Gestione di sessioni

- Registrare una variabile con $_SESSION.

```php
//isset verifica se la variabile è definita if
(!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
}
else {
    $_SESSION['count']++;
}
```

- Resettare una variabile con $_SESSION.

```php
unset($_SESSION['count']);
```

- Registrare una variabile con session_register().

```php
$nome = "SWBD";
session_register($nome);
```

Per evitare di far entrare gente non loggata, devo fare controllo su sessione. Se è settata bene. Altrimenti no. Setta una variabile in sessione per vedere se utente può fare accesso.

PHP: Accesso a DB MYSQL

- Prima di accedere ai dati è necessario creare una connessione con il DB
- PHP offre la funzione ***mysql_connect***, la cui sintassi è:
  - mysql_connect(servername, username, password);
- Ad esempio:

```
mysql_connect("localhost", "test", "test_pwd")
```

PHP: Accesso a DB MYSQL

- Conviene “salvare” il risultato della connessione in una variabile per usi successivi nello script
- Se la connessione fallisce (es. password non corretta), si può usare la funzione die per sollevare un’eccezione con messaggio d’errore
- Esempio di connessione:

```
$con = mysql_connect("localhost", "test", "test_pwd");
if (!$con) {
    die('Connessione fallita: ' . mysql_error());
}
// qua andrà altro codice per uso DB
```

PHP: Accesso a DB MYSQL

- La connessione viene implicitamente chiusa al termine dell'esecuzione dello script. Per farla terminare esplicitamente all'interno dello script si può usare la funzione **mysql_close**:

```
$con = mysql_connect("localhost", "test", "test_pwd");
if (!$con) {
    die('Connessione fallita: ' . mysql_error());
}
// qua andrà altro codice per uso mysql_close($con);
// qua può andare altro codice che non usa il DB
```

PHP: Accesso a DB MYSQL

- Una volta aperta con successo una connessione al server MySQL, occorre selezionare un DB
- In PHP è possibile usare la funzione mysql_select_db, con sintassi: ***mysql_select_db***(database, connessione)

```php
<?php
    $con = mysql_connect("localhost", "test", "test_pwd");
    if (!$con) {
        die('Connessione fallita: ' . mysql_error());
    }
    mysql_select_db("impiegato", $con);
    // altro codice per uso DB impiegato
?>
```

PHP: Query SQL

- In PHP si può usare la funzione ***mysql_query*** per inviare una query
- La funzione restituisce un "cursore", tramite il quale è possibile accedere alle tuple nel risultato come se si trattasse di un array utilizzando la funzione mysql_fetch_array

```php
<?php
    $con = mysql_connect("localhost", "test", "test_pwd");
    if (!$con) {
        die('Connessione fallita: ' . mysql_error());
    }
    mysql_select_db("impiegato", $con);
    $result = mysql_query("SELECT * FROM impiegato");
    while($row = mysql_fetch_array($result)) {
        echo $row['cognome'] . " " . $row['nome'];
        echo "<br />";
    }
    mysql_close($con);
?>
```

PHP: Query SQL - Esempio

Per formattare l'output come tabella HTML:

```
<html>
    <body>
        <?php
            $con = mysql_connect("localhost", "test", "test_pwd");
            if (!$con) {
                die('Connessione fallita: ' . mysql_error());
            }
            mysql_select_db("impiegato", $con);
            $result = mysql_query("SELECT * FROM impiegato");
            echo "<h3>Impiegati</h3> <table border='1'> <tr> <th>Cognome</th> <th>Nome</th> </tr>";
            while($row = mysql_fetch_array($result)) {
                echo "<tr>";
                echo "<td>" . $row['cognome'] . "</td>";
                echo "<td>" . $row['nome'] . "</td>";
                echo "</tr>";
            } echo "</table>";
            mysql_close($con);
        ?>
    </body>
</html>
```

PHP: Form per query

- Tramite form HTML è possibile predisporre delle query parametriche, es (file query_form.htm):

```
<html>
    <body>
        <h3>Cerca Impiegato per Cognome</h3>
        <p>Introduci i dati per la ricerca dell' Impiegato:</p>
        <form method="post" action="query_cognome.php">
            Cognome Impiegato:<input type="text" name="cognome_impiegato">
            <input type="submit" value="Cerca Impiegato" />
        </form>
    </body>
</html>
```

PHP: Form per query

- Il file query_cognome.php è ad esempio il seguente:

```
<html> <body>
	<h3>Selezione Impiegato</h3>
	<? php /* prepara variabili per connessione MySQL */
		$server = "localhost:3306"; // indirizzo server e porta
		$username = "test"; // DB username
		$password = "test_pwd"; // DB password /* Connessione al server MySQL */
		$con = mysql_connect ($server, $username, $password) or die (mysql_error()); /* Selezione il DB per l'accesso */
		if (!mysql_select_db("impiegato", $con)) {
			echo "<p> C'è stato un errore. Questo è un messaggio d'errore:</p>";
			echo "<p><b>" . mysql_error() . "</b></p>";
			echo "Per favore, per dettagli contattare gli amministratori di sistema"; } /* Prepara la Query SQL */
			$sql = "SELECT * FROM impiegati";
			$sql .= " WHERE ( cognome = '{$_POST['cognome_impiegato']}' )"; /* Invia la Query SQL al DB attivo */
			$result = mysql_query($sql, $con);
			if (!$result) {
				echo("<p>Errore nell'esecuzione della query: " . mysql_error() . "</p>");
				exit(); }
	?> <!-- Inizializza la tabella con le intestazioni -->
	<table border=1>
		<tr> <td><b>Matricola</b></td> <td><b>Cognome</b></td> <td><b>Nome</b></td> <td><b>Residenza</b></td> <td><b>Citta</b></td></tr>
	<? /* Recupera le tuple dal result set MySQL e le formatta come righe di tabella HTML */
		while ($row = mysql_fetch_array($result)) {
			echo("<tr>\n<td>" . $row["matr"] . "</td>");
			echo("<td>" . $row["cognome"] . "</td>");
			echo("<td>" . $row["nome"] . "</td>");
			echo("<td>" . $row["residenza"] . "</td>");
			echo("<td>" . $row["Citta"] . "</td>\n</tr>\n\n"); }
	?> <-- Chiude la tabella HTML -->
	</table>
	<? /* Chiude la connessione col server MySQL */
		mysql_close ($con);
	?>
</body> </html>
```

PHP: Esempio di INSERT

- Tramite form HTML è anche possibile predisporre modifiche es. (file insert_form.htm):

```
<html>
    <body>
        <h3>Inserimento nuovo Impiegato</h3>
        <p>Introduci i dati dell'Impiegato:</p>
        <form method="post" action="insert_impiegato.php">
            Matricola: <input type="text" name="matricola_impiegato"> <br/>
            Cognome: <input type="text" name="cognome_ impiegato"> <br/>
            Nome: <input type="text" name="nome_ impiegato"> <br/>
            Residenza: <input type="text" name="residenza_ impiegato"> <br/>
            Citta: <input type="text" name="citta_ impiegato"> <br/>
            <input type="submit" value="Inserisci Impiegato" />
        </form>
    </body>
</html>
```

PHP: Esempio di INSERT

- Il file insert_studente.php è ad esempio il seguente:

```
<?php
    $con = mysql_connect("localhost", "test", "test_pwd");
    if (!$con) {
        die('Connessione fallita: ' . mysql_error());
    }
    mysql_select_db("impiegato", $con); /* Costruzione statement SQL */
    $sql="INSERT INTO impiegato (matricola, cognome, nome, residenza, citta) VALUES
    ('$_POST[matricola_impiegato]', '$_POST[cognome_ impiegato]',   '$_POST[nome_ impiegato]',
    '$_POST[residenza_ impiegato]', '$_POST[citta_ impiegato]'  )";
    if (!mysql_query($sql,$con)) {
        die('Errore: ' . mysql_error());
    }
    echo "1 record inserito";
    mysql_close($con)
?>
```

PHP: cURL

- **cURL** è un comando che serve per scriptare, cioè per scrivere degli **script per trasferire dei dati**, è usato nelle televisioni, nelle automobili, in una notevole varietà di dispositivi. Viene usato per le comunicazioni attraverso la rete.
- cURL è un **tool da linea di comando** quindi si può usare al di fuori di PHP, è uno strumento di networking, per interagire con sistemi remoti e ottenere delle risposte.
- Esempio di download di una pagina web

```php
<?php
$handle = curl_init();
$url = "...url qui...";
// Set the url
curl_setopt($handle, CURLOPT_URL, $url); // Set the result output to be a string
curl_setopt($handle, CURLOPT_RETURNTRANSFER, true);
$output = curl_exec($handle);
curl_close($handle);
echo $output;
?>
```

PHP: cURL

- curl_init() inizializza la sessione;
- curl setopt() opzioni della sessione;
- CURLOPT_URL connessione verso un dominio web;
- CURLOPT_RETURNTRANSFER trasferito come stringa;
- curl_exec() esegue la sessione con tutti i parametri;
- curl_close() chiude la connessione

PHP: cURL

- Esempio di download di un file da remoto
- CURLOPT_URL richiamo un url in remoto;
- CURLOPT_FILE file nel quale scrivere;

```php
<?php
// Remote file
$url = "https://google.it/images/google.jpg";
// Local file
$file = "img.jpg";
$handle = curl_init();
// Open the local file for writing
$fp = fopen($file, "w");
curl_setopt_array($handle, array(CURLOPT_URL=> $url,CURLOPT_FILE => $fp, CURLOPT_HEADER => true ))
curl_exec($handle);
curl_close($handle);
fclose($fp);
?>
```

↳ Caricare o scaricare immagini

PHP: Inviare i dati in POST via cURL

- Possiamo compilare un form di registrazione attraverso cURL passando in POST i valori

```
<?php
$handle = curl_init();
$url = "https://localhost/saw/registration.php";
// Array with form fields names and values
$postData = array( 'firstname' => 'SWBD', 'lastname' => 'Eng', 'email' =>
'swbd@unicampania.it', 'pass'=>'swbd', 'confirm'=>'swbd' );
curl_setopt_array($handle, array( CURLOPT_URL => $url, CURLOPT_POST => true,
CURLOPT_POSTFIELDS => $postData, CURLOPT_RETURNTRANSFER => true ));
$data = curl_exec($handle);
curl_close($handle);
echo $data;
?>
```