UTILI

FORMATO FILE PHP : <nome>.php

sono praticamente file .htm/.html a cui si aggiunge del codice PHP “incapsulato”

INSERIRE CODICE PHP NELL’HTML:

```
<?php
    …codice…
    // nota: tag di apertura e chiusura modificabili dal file php.ini
    //        sconsigliato cambiare tag… diminuisce la portabilità del codice
?>
```

COMMENTI *c-like*

```
//commento inline
#commento inline stile Assembly
/*commento su più righe*/
```

REGOLE SINTATTCHE

- ogni istruzione termina con semicolon
- apertura e chiusura blocchi di codice c-like usando le graffe
- il codice PHP può apparire in ogni parte del codice HTML, a patto di essere contenuto negli appositi tag

INTERPRETAZIONE PHP E HTML

- **HTML** viene restituito sequenzialmente dal server, come al solito
- **PHP** viene eseguito direttamente, senza dare alcuna risposta all’utente; tutto ciò che viene restituito dagli *echo* rientra nel codice HTML; il resto è eseguito internamente al server. *PHP permette di rendere il server capace di rispondere in una certa maniera all’host, in qualunque modo sia necessario, e di operare sui dati nel server.*

FUNZIONI UTILI DA RICORDARE

generatore di numeri random:	rand(a, b) //compreso tra a e b
output su HTML:	echo <expr>;
Potenza di un numero:	pow(<base> , <exp>)

OUTPUT usando *echo*

Nota: funziona anche con AJAX!

formati: echo <expr>; echo(<expr>); echo … , <expr_j> , … ;

echo abbreviate: <?=$<nome_var>?> //nel codice HTML

//SOLO SE l’opzione `short_open_tag` è attiva

particolarità:

- non ha valore di ritorno
- più efficiente di print
- ammette più argomenti

OUTPUT usando *print*

Nota: funziona anche con AJAX!

formati: print <expr>; print(<expr>);

particolarità:

- ha valore di ritorno booleano
- ha un solo argomento

INCLUSIONE DI FILE usando *include*

formato: include <path>

inclusione singola: include_once <path> //ignora alla seconda inclusione dello stesso file

particolarità:

- se ci sono problemi, warning e continua l'elaborazione (E_WARNING)

INCLUSIONE DI FILE usando *require*

formato: require <path>

inclusione singola: require_once <path> //ignora alla seconda inclusione dello stesso file

particolarità:

- se ci sono problemi, errore fatale (E_COMPILE_ERROR)

DATA E ORA

timestamp(ottenere la data in secondi dal 1970) attuale: time()

timestamp da una data qualunque: strtotime(str)

formato stringa:

- aa-mm-gg
- aa mese gg
- *now*
- *next* <giorno della settimana in inglese>
- *last* <giorno della settimana in inglese>
- [+|-]<n> [*day*|*week*]

stampa una data: date(format , timestamp)

format particolari:

- "gg/mm/aaaa h:m:s"
- Y anno completo, d giorno in numero intero da 0, j giorno in numero intero partendo da 1, D i primi 3 caratteri del giorno, F il nome completo del mese, y anno solo le ultime 2 cifre

VEDI

validazione dati in ingresso:
https://www.html.it/pag/62261/validazione-dei-dati-in-php/

elaborazione immagini lato server:
https://www.html.it/pag/67372/manipolare-le-immagini-con-php-e-gd/
https://www.html.it/pag/67924/applicare-filtri-alle-immagini-con-gd-e-php/
https://www.html.it/pag/67924/applicare-filtri-alle-immagini-con-gd-e-php/

creazione di un CAPTCHA:
https://www.html.it/pag/68362/realizzare-un-sistema-captcha-con-php/

formato mail secondo lo standard internet(e altre informazioni):
http://www.faqs.org/rfcs/rfc2822.html

TIPI DI DATO IN PHP

BASILARI

PHP è come ogni linguaggio di scripting debolmente tipizzato. le variabili restano memorizzate dopo diverse aperture del blocco *<?php … ?>* .
le variabili possono anche contenere funzioni. in questo caso si parla di *funzioni anonime*.

Definizione di variabile: $<id> = <val>; //non esiste dichiarazione, solo definizione
definizione di costante: define(<id>, <value>) //CONVENZIONE: nome della cost. in maiuscolo

regole per gli identificatori:

- iniziano sempre per $ (segnala all'interprete la presenza di una variabile)
- segue underscore, carattere o numero
- prosegue con qualunque carattere alfanumerico, trattino o underscore

output variabili in stringhe:

- usando la concatenazione stile Java/JS
- piazzare la variabile all'interno della costante stringa con la solita convenzione dollaro, ad es.:
  "ciao *$nome*"

accesso a variabili globali: *global* $<id>; //anche usando $GLOBALS['<id>']
dichiarazione variabile statica: *static* $<id> …
variabili dinamiche: $$<id> //se $<id> stringa, allora usa il contenuto di
// $<id> come nome della variabile

TIPI DI DATI: associati diamicamente in base al valore come in JS

boolean true false 1 0 1 && 0 0 || 1 …
integer
float 10.3 -33.5 15.3e-19 3E-7 //precisione arbitraria? vedi lib. BCMath
double
string "stringa" 'stringa' 'mi chiamano "stringa"' // + escape …
//attenzione nell'uso di apici quando si resituisce del markup!
array e matrici //considerati oggetti
oggetto //PHP supporta il paradigma OOP
// vedi https://www.html.it/guide/guida-programmazione-ad-oggetti-con-php-5/

VALORI PARTICOLARI
true false null

ARRAY

definizione normale: $<id> = array(… , <arg_k> , …);
definizione array associativo: $<id> = array(… , <arg_key_k> => <arg_value_k> , …);
notazione alternativa: $<id> = [… , <arg_j> , …]
$<id> = [… , <arg_key_k> => <arg_value_k>, …]
accesso normale: $<id>[<n>]
accesso associative: $<id>[<key>]
matrici e array multidimensionali: array che contengono array
accesso multidimensionale: diverse quadre concatenate
dimensioni di un array: count(<array>) //equivale a size
verifica se un elemento è presente: in_array(<occorrenza> , <array>)
mescolare gli elementi: shuffle(<array>)

ordine inverso:	array_reverse(<array>)
concatenazione:	array_merge(… , <array_k> , …)
estrarre una porzione di array:	array_slice(<array>, <from> , <to>)
lista di chiavi:	array_keys(<array>)
lista di valori:	array_values(<array>)

ELABORAZIONE DI STRINGHE

strlen(stringa)	lunghezza della stringa
substr(str, start, end)	sottostringa estratta da str da start incluso a end escluso
strpos(str, pattern, offset)	ritorna la posizione della prima occorrenza di pattern in str a partire dalla posizione offset
strrev(str)	ritorna la reverse di str
strreplace(search, replace, subject)	
strreplace(search, replace, subject, count)	nella stringa subject cerca la prima occorrenza di search (opzionale: dopo un certo indice) e rimpiazzala con replace nota: search, replace e subject possono anche essere matrici!
explode(separator, string, limit)	analogo dello split in java, usando separator come separatore
implode(glue, array)	ricomponi una stringa usando come separatore glue
strtoupper(str)	tutto maiuscolo
strtolower(str)	tutto minuscolo
ucfirst(str)	primo carattere della stringa maiuscolo
lcfirst(str)	primo carattere della stringa minuscolo
trim(str)	elimina spazi non necessari all'inizio e alla fine
ltrim(str)	…solo a sinistra
rtrim(str)	…solo a destra
sprintf(<format> , … , <val_j> , …)	come la printf() in C; supporta i placeholder stile C, con %<char>
strcmp(A, B)	compara A con B; <0 se A<B; =0 se A==B; >0 se A>B
strcasecmp(A, B)	analogo di strcmp() ma non case sensitive

ELENCO DEI PLACEHOLDER

Formato	**Descrizione**
%%	Segno di percentuale.
%b	Numero binario.
%c	Carattere in formato ASCII.
%d	Decimale con segno.
%e	Notazione scientifica in minuscolo (e.g. 1.2e+2).
%E	Notazione scientifica in maiuscolo (e.g. 1.2E+2).
%u	Decimale senza segno.
%f	Numero in virgola mobile (con verifica delle impostazioni di sistema).
%F	Numero in virgola mobile (senza verifica delle impostazioni di sistema).
%o	Numero in formato ottale.
%s	Stringa.

%x Numero in formato esadecimale (lettere in minuscolo).
%X Numero in formato esadecimale (lettere in maiuscolo).

OPERATORI PER COSTRUIRE LE ESPRESSIONI:

assegnazione	=										
aritmetici	+	-	*	/	%						
immediati	+=	-=	*=	/=	%=						
unitari	++$<id>		--$<id>		$<id>++		$<id>--				
logici	and	&&	or	||	xor	!					
confront	==(uguale)		===(identico)		!=	!==	>	>=	<	<=	
			//anche confronto tra stringhe e tra valori interni alle stringhe								
bit a bit											
stringhe	.	//concatenazione									
tipizzazione											

COSTANTI DI DEFAULT:

Costante	Descrizione
`__FILE__`	Contiene il percorso (*path*) del file su cui ci troviamo.
`__DIR__`	Contiene il percorso della directory in cui è contenuto il file corrente.
`__FUNCTION__`	Contiene il nome della funzione che stiamo utilizzando.
`__LINE__`	Contiene il numero di riga corrente.
`__CLASS__`	Contiene il nome della classe corrente.
`__METHOD__`	Contiene il nome del metodo corrente.
`__NAMESPACE__`	Contiene il nome del namespace corrente.
`__TRAIT__`	Contiene il nome del trait corrente.

ESPRESSIONI REGOLARI

FUNZIONI DI LIBRERIA PER UTILIZZARE LE ESPRESSIONI REGOLARI

preg_match(pattern, subject)	cerca corrispondenze con un'espressione regolare in subject, restituisce la prima occorrenza
preg_match(pattern, subject, array_result)	salva tutti I risultati di ricerca in array_result(concatena), restituisce tutte le occorrenze
preg_match_all(…)	restituisci tutte le occorrenze in subject
preg_replace(pattern, replace, subject, (opt)limit)	limit: limite di sostituzioni effettuabili sostituisce tutte le occorrenze con replace
preg_replace_callback(pattern, callback, subject, limit)	al posto di rimpiazzare, esegue una elaborazione data dalla funzione di callback(arg: l'occorrenza)

SINTASSI ESPRESSIONI REGOLARI

- inizia per /
- ^ indica che la stringa deve iniziare col pattern indicato
-

ELEMENTI SINTATTICI ESPRESSIONI REGOLARI(Perl Compatible Regular Expression)

Meta carattere	**Descrizione**
\	Carattere generico di escape
^	Delimitatore di inizio della stringa
$	Delimitatore di fine della stringa
.	Definisce ogni carattere eccetto il carattere di invio
[	Carattere di inizio della definizione di classe
]	Carattere di fine della definizione di classe
|	Inizio di un ramo alternativo
(	Inizio subpattern
)	Fine subpattern
?	Indica 0 o 1 occorrenze
*	0 o più occorrenze
+	1 o più occorrenze
{	Inizio intervallo minimo/massimo di occorrenze
}	Fine intervallo minimo/massimo di occorrenze
–	Indica un range di caratteri all'interno di parentesi []
.	Indica un singolo carattere
\s	Un carattere di spaziatura (space, tab, newline)
\S	Tutto eccetto un carattere di spaziatura
\d	Un carattere numerico (0-9)
\D	Tutto eccetto un carattere numerico

\w	Una lettera (a-z, A-Z, 0-9, _)
\W	Tutto eccetto una lettera
[aeiou]	Uno dei caratteri compresi nella parentesi
[^aeiou]	Tutto eccetto i caratteri compresi nella parentesi
(foo|bar|baz)	Una delle alternative tra parentesi

COSTRUTTI DI FLUSSO

IF – ELSEIF – ELSE:

```
if (condizione)
{
        //codice
}
elseif (condizione)
{
        //codice
}
//tanti elseif dopo…
else
{
        //codice
}
```

IF – ELSE BREVE

```
(condizione) ? …codice if : codice else ;        //NOTA: si può inserire anche in espressioni più complesse
                                                 //anche concatenazioni
```

PARTICOLARE DEL COSTRUTTO CONDIZIONALE:

```
<?php if ($error): ?>
    <p style="color: red"><?php echo $error ?></p>
<?php else: ?>
    <p>Benvenuto <?php echo $username ?></p>
<?php endif ?>
```

il costrutto condizionale non interrompe il suo funzionamento anche se il blocco di codice si chiude! e così accade anche per tutti gli altri costrutti.

SWITCH

```
switch(variabile)
{
        case <valore_1>:
        //codice valore 1
        break;

        //…altri casi
        //nota: proprio come in C, Java, … anche sequenza di casi e raggruppamenti di casi
```

```
        default:
        //codice di default
        break;
}
```

CICLI come in C

- for con 3 statement
- while
- do…while…

soliti meccanismi di controllo del ciclo utilizzando break, continue.

FOREACH

```
//per strutture semplicemente iterabili…
foreach(<struttura> as <elem_struttura_alias>)
{
        //…codice…
}

//per strutture associative…
foreach(<struttura_associativa> as <key_elem_alian> => <value_elem_alias>)
{
        //…codice…
}
```

USER DEFINED FUNCTION

```
function ( … , $<arg_k> , … )       //solo passaggio per valore
{
        //…codice…
}
```

ritorno o chiusura:

```
        return <qualcosa>
```

è possibile utilizzare anche funzioni anonime da passare ad altre funzioni come argomento, o da assegnare a delle variabili:

```
$var = function(<arg_list>) {…codice…};                    //assegno ad una variabile una funzione anonima
myFunct( … , function(…arglist…){…} , …)                    //passaggio di funzione ad argomento
```

JSON

METODI PER ELABOARE IL JSON

json_decode(json_str)	restituisce un array associativo ritorna NULL se la stringa è mal formata
json_decode(json_str, assoc)	restituisce un array se assoc=false (valore di default) altrimenti se assoc=true ritorna uno stdClass
json_last_error_msg()	ritorna una stringa con l'ultimo errore json
json_encode(array)	converti un array associativo in stringa json

GESTIONE FILES IN PHP

PRIMITIVE

file(path)	trasforma il contenuto del file in un array
file_exists(path)	controlla se un file nel server esiste (true o false)
is_file(path)	controlla se il path passato è un file o una cartella (true o false)
filesize(path)	ritorna le dimensioni del file indicato, in bytes (false per problemi)
fileatime(path)	data dell'ultimo accesso al file
filectime(path)	data dell'ultima modifica al file
file_get_contents(path)	ritorna TUTTO IL CONTENUTO del file
readfile(path)	leggi l'intero file
file_put_contents(path, data, flag)	scrive su file FLAG = FILE_APPEND per la scrittura in append altrimenti sostituisce l'intero file
copy(path_from, path_to)	copia il file in un altro file
unlink(path)	elimina dal file system un file

MUTUA ESCLUSIONE

flock(stream, mode)

modalità:

- LOCK_SH blocca in lettura, non in scrittura
- LOCK_EX blocca in scrittura, non in lettura
- LOCK_UN rimuovi il blocco, il processo rilascia il file

quando tento di bloccare la risorsa, la funzione ritorna TRUE se la risorsa è libera, altrimenti FALSE.

METODI TRADIZIONALI

fopen(path, mode)	classica fopen del C ritorna uno "stream"

tipica aperture di un file:

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
```

modalità:

r Open a file for read only. File pointer starts at the beginning of the file

w Open a file for write only. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

a Open a file for write only. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

x Creates a new file for write only. Returns FALSE and an error if file already exists

r+ Open a file for read/write. File pointer starts at the beginning of the file

w+ Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

a+ Open a file for read/write. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

x+ Creates a new file for read/write. Returns FALSE and an error if file already exists

fread(stream, n_bytes)	leggi un tot di bytes dal file
fgets(stream)	leggi una sola riga del file
fgetc(stream)	leggi un singolo carattere
feof(stream)	verifica se sono arrivato o no a fine file
fflush(stream)	forza il sistema a rilasciare l'output
fclose(stream)	chiudi la comunicazione col file

FUNZIONALITA' SERVER-SIDE

URL CON RICHIESTA IN GET

<dominio>/<path>?<varlist>

dove <varlist> rispetta il format

<var_key>=<var_value>

per tanti valori di seguito si usa …&… senza spazi; nessun carattere di terminazione.
una richiesta in post mette le variabili e i suoi valori direttamente nello header http
ATTENZIONE: SANIFICARE SEMPRE I DATI PRIMA DI UTILIZZARLI NEL PROPRIO SCRIPT!

$_GET[<key>]

- *array associativo*, che conserva tutte le variabili inviate in get dal client
- lo script vede solo le variabili associate alla richiesta attualmente servita, non quelli di altre richieste parallele
- i dati in GET vengono passati direttamente dall'URL, sono perciò visibili al client

$_POST[<key>]

- *array associativo*, conserva i dati inviati in post; il comportamento finale è lo stesso di GET
- i dati non sono contenuti nell'URL ma nello header http

COOKIES

a cosa servono i cookies:

- a memorizzare un login
- a tener traccia di alcune opzioni di configurazione (es: chiudi quel popup…)
- a caratterizzare la richiesta dell'utente al server, in modo che la risposta risulti adeguata all'attività dell'utente

aggiungere un cookie: setcookies(name , value , expire , path , domain , secure , httponly)

name Nome del cookie, nel nostro caso `user_id`.

value Valore del cookie (`345`).

expire Scadenza del cookie (1 anno).

path Percorso per cui è valido. Di default è impostato su tutto il sito, ma se ad esempio scegliamo `/sottodirectory/` esso sarà valido solo per quel percorso.

domain Dominio per cui è valido.

secure Indica se il cookie dovrebbe essere trasmesso attraverso una connessione HTTPS.

httponly Indica se il cookie è accessibile solo attraverso il protocollo HTTP. Questo implica, ad esempio, che se il valore è `true` il cookie non è accessibile da Javascript.

modificare il valore di un cookie: sovrascrivi con setcookie()

eliminare un cookie: sovrascrivi con setcookie() usando un cookie scaduto
unset($_COOKIE[<name>])

$_COOKIE[<name>]

- *array associativo*, conserva i cookie in arrivo dal client (inviati durante la richiesta al server tramite HTTP)
- lo script vede solo i cookies associati alla richiesta che sta servendo attualmente, non quelli di tutte le altre richieste
- i cookies vengono memorizzati su client, spediti al server all'occorrenza, spediti dal server al momento della creazione del cookie o dell'aggiornamento, e eliminati dal server una volta che la richiesta è stata risolta

SESSIONI

caratteristiche:

- una sessione è un insieme di variabili che vengono conservate sul server (eventualmente nel DB)
- i cookies servono a tenere traccia di una sessione aperta
- i dati di sessione sono accessibili solo dopo aver inizializzato la sessione
- l'id assegnato alla sessione si può trovare in $_COOKIE['PHPSESSID']

inizializzare una sessione: session_start()

riprendere una sessione già aperta: session_start() //continua in automatico

distruggere una sessione: session_destroy()

$_SESSION[<name>]

- *array associativo*
- accessibile solo dopo aver inizializzato la sessione; non più accessibile subito dopo
- rimuovi una variabile di sessione utilizzando unset()

HASHING PASSWORD e altri dati sensibili:

md5(str)
md5(str, salt) //aumenta la complessità combinando la stringa con una "salt"
sha(str)
crypt(str, salt) //più sicura, ma anche più macchinosa da usare
password_hash(psw , algoritmo) //algoritmi: PASSWORD_DEFAULT, PASSWORD_BCRYPT

operazione a senso unico: non esiste un algoritmo per la decodifica. Ma ad ogni stringa è associato un hash unico.**Mai lasciare le password in chiaro nel database!** Inoltre, è buona norma fare diversi hash innestati delle password in modo da aumentare la complessità computazionale di un ipotetico algoritmo che potrebbe decodificare quelle password (l'unica decodifica possibile è quella per tentativi…)

hash_equals(hash_1 , hash_2) //guarda se i codici hash coincidono
password_hash(psw, hash)

INVIO DI MAIL USANDO PHP

mail(indirizzo e-mail, oggetto mail, contenuto mail) //utilizzo tipico
mail(to, oggetto, messaggio, header, parametri) //firma completa

il campo "to" può contenere un solo indirizzo, o più indirizzi (più email separate da virgola nella stessa stringa, vedi funzione implode()). formati accettati:

- user@example.com
- user@example.com , anotheruser@example.com
- user_alias <user@example.com>

il campo header permette di inserire indirizzi in CCN. e altre opzioni, tutto separato da virgola.

in particolare per poter inserire del markup nel contenuto della mail, usare una componente dello header del tipo

```
Content-type: text/html
```