Guida creata da Matteo Iammarrone sito web= http://www.matteoiammarrone.com
Raccolta da nonsoloprogrammi sito web= http://www.nonsoloprogrammi.it
Ho pensato di scrivere questa guida per imparare il php da zero.

Introduzione:
Lo scopo è quello di consentire a chi non conosce questo magnifico di linguaggio di impararlo facilmente e gratuitamente tramite questa semplice guida, presenti in questa sezione del forum di questo sito(Sezione Php e Mysql).

Seguite attentamente tutti i tutorial: sono ordinati, numerati e vanno seguiti passo dopo passo(non saltate nessuna lezione se veramente volete imparare il linguaggio!)

Per navigare e seguire correttamente la guida utilizzate il menu "Php da zero" che vedete a destra di tutte le pagine

Introduzione al php e Ambiente di lavoro

Il php è stato sviluppato nel 1994, è un linguaggio lato server, è simile al C e al Perl ed è il maggior linguaggio di sviluppo del web. E' importante ricordare la sua interazione con database come MYSQL.

Ambienti di lavoro

Prima di iniziare a seguire questa guida e' strettamente consigliata una conoscenza minima di html/ftp.
Possedere dunque:
- Un qualsiasi editor html(Es. Dreamwavare, Golive, anche il blocco note va bene)
- Uno spazio web con connessione ftp e supporto php(es. lo puoi avere gratis tramite altervista.org), oppure in alternativa puoi installare server locali come Xamp Server.
Importante notare come L'HTML sia integrabile facilmente nel php. Le pagine php hanno estensione .php.
Questa guida ha la scopo di insegnare a pieno le basi e l'essenziale del php. Buona Fortuna, proseguite alla Prima Lezione.

LEZIONE 1: Il mio primo programma: echo e html

Iniziamo a programmare seriamente,
come detto in passato nel php e'integrabile la piattaforma HTML.
Iniziamo, per facendo qualcosa di veramente veramente semplice: sviluppiamo una pagina php che visualizzi la scritta "Ciao Mondo!".
miapagina.php

Codice:

```
<?php
echo "Ciao Mondo!";
?>
```

Dal codice sovrastante notiamo come si utilizza il comando echo,
che serve per stampare(Cioe' mostrare/visualizzare) un testo o un codice html.
Altro esempio:

Codice:

```
<?php
//COMMENTO DI ESEMPIO,  QUESTO E' UN COMMENTO,
// I COMMENTI IN REALTA'  NON HANNO NESSUN  EFFETTO SUL RISULTATO DEL CODICE,
SERVONO SOLO //PER COMMENTARE,  QUELLO CHE SI PROGRAMMA,  POSSONO RIVELARSI UTILI SE
SI  PUBBLICA LO SCRIPT, //CHI LEGGE  IL CODICE PUÃ² LEGGERE I COMMENTI
echo "<center><img src='immagine.jpg/></center>";
//Qui notiamo perfettamente come il codice html si integri nel php
?>
```

Possiamo concludere, dunque che per integrare un codice html(Qualsiasi) nel php basta utilizzare il comando echo (echo "Codice html";) e inserire il codice html sostituendo perÃ² tutti gli " con '
Quindi, se io ad esempio scrivo:

Codice:

```
<img src="immagine.jpeg"/>
```

e' scorretto!! la forma corretta e' questa:

Codice:

```
<img src='immagine.jpeg'/>
```

LEZIONE 2: Variabili e Defines

Le variabili nella programmazione e nel php sono degli elementi che vanno a sostituire un valore. Ogni variabile, infatti ha nome e rispettivo valore.
Esempio pratico di variabile:

Codice:
```
<?php
//Il classico codice di inizio di tutte le pagine php
$testo = "Ciao ragazzi"; //OSSERVATE E MEMORIZZATE, ECCO COME SI DEFINISCE UNA VARIABILE !!
echo $testo; //Qui apparira'  la scritta Ciao ragazzi! oppure si puo' scrivere anche cosi
echo "$testo";
//Il classico codice di fine di tutte le pagine php
?>
```

Le variabili sono usate spessisimo, servono come punti di riferimento. Si possono controllare in tanti modi (if, ecc..),
imparerete piu' in la' l'utilita'.
Al posto delle variabili possiamo usare i define, spesso questi vengono utilizzati in alcuni casi particolari e
molto piu' raramente delle variabili.
Non tanto per contenere dati, ma piu' che altro
per stabilire punti di riferimento nel codice.

Codice:
```
<?php define("CHEDIRE", "Viva la vita");
echo "".CHEDIRE."";  //Apparira'  la scritta "Viva la vita"
?>
```

Proviamo a fare qualcosa di piu' interessante:
incrociamo in un codice tutto quello che abbiamo
imparato sino ad ora(Anche della lezione precedente)

Codice:
```
<?php
$nome="Matteo";
$anni=14; //QUI NOTATE UNa anticipazione  della prossima lezione, come notate i  numeri, es 14  non hanno bisogno di  "" ma si possono scrivere
```

```
direttamente
$infos="Guida php by matteoiamma";
define("CHESCRIVERE", $infos);  //QUI DIAMO COME VALORE DEL DEFINE UNA
VARIABILE,  OSSERVATE E MEMORIZZATE
echo "<center><b>Mi chiamo $nome, ho $anni anni</b></center>"; //Avremo
come risultato la scritta  Mi chiamo Matteo, ho 14 anni,  centrata e
grassetata  con sotto scritto: Guida php by matteoiamma
echo "".CHESCRIVERE."";
?>
```

LEZIONE 3: Controllo IF & Operazioni

Come anticipato nella lezione scorsa le variabili si definiscono in questo modo:

Codice:
```
<?php
$variabile="valore";
?>
```

Ma ci sono alcune eccezioni in cui non è obbligatorio mettere "" e sono tutte qui sotto elencate:

- Nel caso di numeri, esempio:

Codice:
```
$variabile = 2; //QUESTA SINTASSI E' CORRETTA, PERCHè IL VALORE CHE DIAMO ALLA VARIABILE E' UN NUMERO E QUINDI NON C'è BISOGNO DI METTERE ""
```

- Nel caso di true o false

$variabile = true;

//QUESTA SINTASSI E' CORRETTA, PERCHè IL VALORE DI $VARIABILE E' TRUE(TRUE E FALSE SONO DUE ECCEZIONI CHE NON NECESSITANO L'UTILIZZO DI "")

Operazioni Matematiche

E' possibile sommare variabili con contenuti numerici.

Ecco un'esempio:

Codice:
```
<?php

$primonumero = 2;
$secondonumero = 3;

$somma = $primonumero + $secondonumero;

echo "Il risultato è $somma";
```

```
//Qui apparirà il testo "Il risultato è 5"

?>
```

Cosi' come, nell'esempio sopra abbiamo eseguito un'addizione e' possibile eseguire una sottrazione(mettendo al posto di + -), eseguire una moltiplicazione (mettendo al posto di + *) e una divisione (mettendo al posto di + /)

I Controlli

Nei linguaggi di programmazione sono fondamentali i controlli sulle variabili.
Se i controlli non sarebbero esistiti i linguaggi non avrebbero avuto senso perchè quasi inuitili.

Il controllo principale del php è **if**, che letteralmente tradotto dall'inglese significa "se";
infatti if serve propio a questo.

Ecco un'esempio di utilizzo:

Codice:

```
<?php

$eta= 14; //QUI DEFINIAMO BANALMENTE UNA VARIABILE NUMERICA

if ($eta == 18){ //LEGGETE E MEMORIZZATE, ECCO COME SI CREA UN CONTROLLO
IF

echo "Hai 18 anni tondi tondi!"; //MOSTRIAMO UN BANALISSIMO TESTO

} else { //altrimenti

echo "Sei minorenne";

}

//IL CONTROLLO SOVRASTANTE SERVE A DIRE AL PHP:
//SE(if) IL CONTENUTO DELLA VARIABILE ETA($eta) E' UGUALE(==) A 18 CHE
APPAIA LA SCRITTA(echo): "Sei maggiorenne", altrimenti (} else { ) che
```

```
appaia la scritta "Sei minorenne"


?>
```

Nel php è possibile utilizzare i segni < e > per fare "paragoni" tra due variabili numeriche.

Ecco un'esempio molto simile a quello sovrastante,
pero' anzichè utilizzare == utilizziamo il segno maggiore.

Codice:

```
<?php

$eta= 14; //QUI DEFINIAMO BANALMENTE UNA VARIABILE NUMERICA

if ($eta < 18){  //SE IL VALORE DI ETà E' MAGGIORE DI 18

echo "Sei maggiorenne!"; //MOSTRIAMO UN BANALISSIMO TESTO

} else { //altrimenti

echo "Sei minorenne";

}


?>
```

Nella prossima lezione impareremo ad usare due elementi chiave del php,
le variabili predefinite $_POST e $_GET

LEZIONE 4: Post e controlli variabili

Se nel php non fosse possibile l'interazione con elementi fisici in html, il php non avrebbe senso di esistere.

Nel php sono presenti alcune variabili predefinite:

$_POST

e

$_GET.

La variabile $_POST è obbligatoriamente "collegata" a un form fisico html, serve dunque per ricavare un dato inserito in un campo di un form html che abbia come metodo "post".

Andiamo alla pratica.

Creeremo una pagina in html che conterrà la parte fisica dello script(Il form e i campi in html) e una pagina che conterrà la parte php/azione dello script.

pagina.html

Codice:

```
<form action="pagina.php" method="post">
Eta' : <input type="text" name="eta"/>
<br>
Nome: <input type="text" name="nome"/>
<br>
<input type="submit" name="send" value="Invia Form"/>
</form>
```

pagina.php

Codice:

```
echo $_POST['eta'];
```

```
echo "<p></p>"; //Mettiamo questo codice per inserire uno "spazio"
//Qui verrà mostrato ciò che ha inserito l'utente nel campo di nome eta.

echo $_POST['nome'];

//Qui verrà mostrato ciò che ha inserito l'utente nel campo nome
```

Per completare il capitolo della di $_POST dobbiamo fare una anticipazione alle funzioni.

E' possibile controllare tutte le variabili tramite alcune funzioni:

- empty (la sintassi è

Codice:
```
if (empty($nomevariabile)){
```
, serve a controllare se la variabile è vuota)

- isset (Simile al precedente ma meno specifico, serve a controllare se la variabile esiste(sintassi:

Codice:
```
if (isset($nomevariabile)){
```

Voglio anche dirvi che nel php è possibile usare "!",
serve per "invertire" il significato della funzione che lo sussegue.

Ad esempio.
Scrivere:

Codice:
```
if (empty($nomevariabile)){
```

Serve a dire se la variabile $nomevariabile è vuota

Invece scrivere:

Codice:
```
if (!empty($nomevariabile)){ //CON ! DAVANTI
```

Serve a dire se la variabile $nomevariabile non è vuota

Applichiamo queste due funzioni al codice sovrastante,
per completarlo e migliorarlo:

Codice:

```
if ($_POST['send']){

if (!empty($_POST['eta'])){
echo $_POST['eta'];
} else {
echo "Hai lasciato vuoto il campo età";
}

echo "<p></p>"; //Mettiamo questo codice per inserire uno "spazio"
//Qui verrà mostrato ciò che ha inserito l'utente nel campo di nome eta.

if (empty($_POST['nome'])){
echo "Il campo nome è vuoto!";
} else {
echo $_POST['nome'];
}
//Qui verrà mostrato ciò che ha inserito l'utente nel campo nome
}
```

C'è anche da dire che per controllare se una variabile esiste è possibile anche non usare alcuna funzione,
ma metterla normalmente in una parentesi.

Ad esempio:

Codice:

```
if ($nomevariabile){
echo "Ok, ci siamo!";
}
```

significa:

Se la variabile $nomevariabile esiste stampa il testo "Ok, ci siamo"

Nella prossima lezione parleremo di switch e get

LEZIONE 5: Switch e Get

Premetto che tutti i controlli e le funzioni valide per $_POST sono valide per $_GET.

Ma allora qual è la differenza tra $_GET e $_POST?

$_POST è più sicuro e viene utilizzato per ottenere dei dati da un form.

$_GET viene utilizzato per gestire dei dati tramite "url".

Ad esempio.

esempio.php

Codice:

```
<?php
echo "Ciao";

echo $_GET['nome'];
?>
```

Se dal browser raggiungiamo la pagina esempio.php in questo modo:

esempio.php?nome=Matteo

il messaggio che verrà mostrato sarà "Ciao Matteo";

se la raggiungiamo in quest'altro modo:

esempio.php?nome=Giovanni

il messaggio che verrà mostrato sarà "Ciao Giovanni";

e così via.

Un fondamentale componente da utilizzare principale per i $_GET (ma non solo) è **switch**

Scrivere:

pagina.php

Codice:

```
<?php

switch($_GET['stato']){

case "usa":
echo "Washigton";
break;

case "italia":
echo "Roma";
break;

case "francia":
echo "Parigi";
break;

default:
echo "Stato non definito!";
break;



}

?>
```

equivale a scrivere questo:

Codice:

```
<?php
if ($_GET['stato'] == "usa"){
echo "Washigton";
} elseif ($_GET['stato'] == "italia"){
echo "Roma";
} elseif ($_GET['stato'] == "francia"){
echo "Parigi";
} else {
echo "Stato non definito!";
```

```
}
?>
```

Scrivendo i due codici sovrastanti il risultato sarà lo stesso identico!!

Dunque, possiamo affermare che switch può sostituire il controllo if.

Lo script che abbiamo creato sopra fa questo:

Se raggiungiamo la pagina esempio.php dal nostro browser in questo modo:

pagina.php?stato=italia

ci apparirà la scritta "Roma",

raggiungendo in quest'altro modo:

pagina.php?stato=francia

ci apparirà la scritta "Parigi"

e cosi' via.

Invece, raggiungendo la pagina in questo modo:

pagina.php?stato=

oppure cosi':

pagina.php

(Cioè con il $_GET "stato" vuoto)

ci apparirà la scritta:

"Stato non definito!";

Gli Switch sono usatissimi per creare quell'effetto che agli utenti(e ai webmaster) piace tanto,
cioè un'intero sito in una sola pagina


Ad esempio:

index.php

Codice:

```
<?php
echo "<a href='index.php'>Home</a> | <a
href='index.php?page=contatti'>Contatti</a> | <a
href="index.php?page=bio'>Biografia</a> | <a
href='index.php?page=links'>Links</a>";
echo "<p></p>";

switch($_GET['page']){

case "bio":
echo "Contenuto della pagina biografia";
break;

case "links":
echo "contenuto della pagina links";
break;

case "contatti":
?>
Contenuto della pagina contatti(IN HTML, chiudendo i tags php qui posso
usare normale html!!!)
<?php
break;

default:
echo "Contenuto della home page, cioè di quando il get page è vuoto!";
break;

}
?>
```

Nella prossima lezione parleremo di altre variabili predefinite del php.

LEZIONE 6: Variabili Predefinite e array

Con le varie versioni del php sono state introdotte delle variabili predefinite.

Le variabili predefinite spesso sostituiscono lunghi codici.

Ecco la lista completa delle principali:

Codice:
```
echo $_SERVER['HTTP_COOKIE']; //Mostra il valore di tutti i cookie
echo $_SERVER['HTTP_HOST']; // Mostra il nome dell'host su quale risiede
il server web
echo $_SERVER['REMOTE_ADDR']; // Mostra l'indirizzo ip dell'utente
echo $_SERVER['PHP_SELF']; // Mostra il nome file della pagina corrente
echo $_SERVER['SCRIPT_FILENAME']; //Mostra il nome e il percorso completo
dello script corrente
echo $_SERVER['SERVER_NAME']; //Mostrail nome del server web
echo $_SERVER['HTTP_USER_AGENT']; //Mostra il nome univoco del browser
utilizzato dall'utente
```

Poi ci sono $_POST e $_GET che già conosciamo e $_SESSION e $_COOKIE che vedremo in seguito

Adesso vediamo gli array.

Iniziamo con un'esempio:

Codice:
```
<?php
$nomearray = array ("giorgio","nicola","filippo"); //definiamo questa
variabile/array
?>
```

per mostrare il contenuto, poi faremo cosi':

Codice:
```
<?php
echo $nomearray[0] ; //Verrà mostrata la scritta giorgio
echo $nomearray[1]; //Verrà mostrata la scritta nicola
```

```
echo $nomearray[2]; //Verrà mostrata la scritta filippo e cosi' via..
?>
```

Capito il meccanismo?

Adesso per completare il capitolo sugli array sono costretto a farvi un'anticipazione alle funzioni.

E' possibile, infatti, in ambito di array utilizzare la funzione in array(in_array()).

La funzione è strutturata in questo modo:

in_array("paroladacercare", "arrayincuicercare");

Basandoci sul codice di sopra potremmo usare la funzione in_array in questo modo':

Codice:

```
<?php
if ( in _array ("giorgio",$nomearray)) {
echo "si c'e giorgio";
}else{
echo "no, non c'e ";
}
?>
```

Nella prossima lezione vediamo le funzioni: come si crea, quelle predefinite e a cosa servono.

LEZIONE 7: Funzioni: Parte 1

I linguaggi di programmazione senza le funzioni sarebbero come la grammatica senza verbi.

Ma a cosa servono?

E' difficile rispondere istantaneamente a questa domanda.

Possiamo dividere le funzioni in tre grandi gruppi:

- Quelle per ricavare informazioni e gestire variabili, defines, stringhe e array

- Quelle per la gestione delle date

- Quelle per la gestione, le operazione e la gestione dei database mysql

- Quelle per la gestione dei files

Prima di iniziare, pero' devo parlare di due funzioni importantissime: **include e require**

Entrambe servono per implementare una pagina in un'altra pagina, quindi per "unire" due pagine.

Solo che se si usa include e la pagina da implementare non esiste, non c'è nessun problema.

Se invece, si usa require e la pagina da implementare non esiste, appare un'errore.

Include serve a dire: se la pagina esiste implementala, altrimenti non fa niente

Require serve a dire: se la pagina esiste implementala, altrimenti arrabbiati!

Esempio di utilizzo:

Creiamo due pagine.

include.php

Codice:

```php
<?php
echo "ciao";
?>
```

pagina.php

Codice:

```php
<?php
include("include.php");
?>
```

- Funzioni per ricavare informazioni e gestire variabili, defines, stringhe e array

Ecco a voi le principali:

- isset (Controlla se una variabile esiste)

- empty (Controlla se una variabile è vuota)

- stripslashes (Elimina gli slash in una variabile, molto utilizzata per fixare i $_POST e i $_GET)

- strlen (Conta il numero di caratteri presenti una variabile)

- str_replace (Serve per sostituire una lettera o una parola in una variabile , si usa cosi':

Codice:

```php
<?php
$variabile="Mi chiamo Matteo";

$variabile_nuova = str_replace("Matteo", "Giovanni", $variabile);
//Significa sostituisci Matteo con Giovanni nella varaibile di nome $variabile)
```

```
echo $variabile_nuova; //IL testo che apparirà sarà Mi chiamo Giovanni, perchè abbiamo sostituito Matteo con Giovanni


?>
```

)

- strip_tags (Serve per eliminare i tags html da una variabile ad esempio:

Codice:

```
<?php
$variabile='Ciao <img src="immagine.jpg"/>';

$variabile=strip_tags($variabile);

echo $variabile;
//Apparirà solo il testo Ciao
?>
```

)

- intval, is_Int (Entrambe servono per controllare se una variabile o un testo è intero, cioè senza spazi
Esempio di utilizzo:

Codice:

```
<?php
$testo = $_GET['testo'];
if (is_int($testo)){
echo "Il testo è intero";
} else {
echo "Il testo non è intero!";
}

?>
```

Il semplice script sviluppato qui sopra funziona cosi':
Se andiamo alla pagina che lo contiene dal browser in questo modo:
nomepagina.php?testo=valoretesto

analizzerà "valoretesto".

Se valoretesto (Cioè il contenuto del $_GET testo) è intero,
apparirà la scritta Il testo è intero, altrimenti:
" IL testo non è intero".
).

- functions_exist (Controlla se una funzione esiste, ad esempio

Codice:

```
if(function_exists("nomefunzione)){
echo "La funzione nomefunzione esiste!";
}
```

)

- preg_match (E' possibile controllare se un testo o una variabile contiene un determinato valore.

Ad esempio:

Codice:

```
<?php
if (preg_match("/php/i", "PHP è il linguaggio scelto.")) {
    echo "Il tuo testo contiene la scritta php!, parola trovata  nel
testo!.";
} else {
    echo "Testo non riconosciuto.";
}
?>
```

- explode (Divide un testo in più array, basandosi su un parametro..
per farvi capire(Esempio di utilizzo):

Codice:

```
<?php
$nomi = "Matteo,Salvio,Alessandro,Federico,Fabio,Alessio";
```

```
$nome = explode(",", $nomi);

echo $nome[0]; //Apparirà "Matteo"
echo $nome[1] //Apparirà "SAlvio"
echo $nome[2]; //Apparirà Alessandro

//E cosi' via
?>
```

- in_array (Controlla se un valore è nell'array, ad esempio:

Codice:

```
<?php
$estensione="gif";
$estensioni_accettate = array('bmp', 'jpg', 'gif');
if( in_array($estensione, $estensioni_accettate) )
{
echo "$estensione";
}
else
{
echo "no";
}
?>
```

)

- is_numeric (Controlla se la variabile ha un valore numerico)

Gran parte delle funzioni sopra elencate hanno questa sintassi:

nomefunzione($nomevariabile);

Ecco un'esempio di utilizzo di stripslashes

Codice:

```
$nomevariabile=stripslashes("valorevariabile");
```

Ovviamente al valore della variabile potete mettere anche un $_POST, quindi mettere nelle variabile un dato proveniente da un form:

Codice:
```
$nomevariabile=stripslashes($_POST['nomecampo']);
```

Funzioni Personalizzate

Come creare una funzione in php?

Semplice.

Codice:
```
function nomefunzione($var){
}
```

dopodichè la richiamiamo come si richiamano tutte le funzioni del php(anche quelle predefinite):

nomefunzione("matteo");
(Apparirà la scritta Matteo).

Se io avessi scritto:

Codice:
```
function nomefunzione($var, $var2){
}
```

avrei dovuto richiamare la funzione in questo modo:

nomefunzione("Valore1", "Valore2");

Inserendo cioè i valori di ciascuna variabile in ordine.

Fare questa operazione(Creare una funzione) serve ad immagazzinare una o più

variabili e gestirle come si vuole in modo semplice.
Ad esempio, posso anche fare cosi':

Codice:

```
<?php
function fixatesto($testo){
$dafixare=stripslashes($testo);
$dafixare = strip_tags($dafixare);
return $dafixare;
}

fixatesto("Ciao ///()))");
?>
```

Nella funzione sovrastante una cosa importante da notare è **return**.
In pratica return è il contrario di global. Serve a far "uscire" la variabile dalla funzione,
cioè se io definisco una variabile nella funzione senza scriverci return la variabile verrà definita solo nella funzione, se io, invece ci metto return la variabile sarà definita per tutta la pagina.

Devo parlarvi, poi delle **variabili globali**. Se io definisco una variabile fuori da una funzione per usarla dovro' mettere global seguito dal nome della variabile e da ; all'inizio della funzione (subito dopo {).
Esempio pratico:

Codice:

```
<?php
$testo2 = "Alessandro";

function stampa_testo($testo1){
global $testo2; //LA VARIABILE $TESTO2 LA DEFINISCO FUORI DALLA FUNZIONE,
QUINDI PER POTERLA USARE NELLA FUNZIONE DEVO SCRIVERE GLOBAL $TESTO2;

echo $testo1;
echo ",   ";
echo $testo2;
}

echo stampa_testo("Matteo"); //Verrà mostra la scritta Matteo, Alessandro
?>
```

Adesso, prima di passare a vedere le principali funzioni predefinite del php voglio terminare la lezione sulle funzioni personalizzate creando una funzione che unisce tutto ciò che abbiamo imparato (una funzione con tante variabili, con variabili globali, con i returns e una funzione che include anche le conoscenze delle lezioni precedenti).

Codice:

```
<?php

$marquee=true;
$direction = "up";

function show_text($nome, $frase, $tradotta, $lingua, $type){
global $marquee, $direction;
if ($marquee == true){
echo "<marquee direction='$direction'>";
}
echo "<$type>";
if ($lingua == "it"){
echo $frase;
} else {
echo $tradotta;
}
echo "</$type>";
if ($marquee == true){
echo "</marquee>";
}


$solonome = str_replace("Iammarrone", "", $nome);
$solonome = str_replace(" ", "", $nome);

return $solonome;
}


echo show_text("Matteo Iammarrone", "Il mio nome è", "My name is", "it",
"b"); //Qui apparirà la scritta Il mio nome è Matteo, scorrevole verso
l'alto, grassetta.
```

```
echo $solonome; //Qui apparirà la scritta Matteo
?>
```

Nella prossima lezione vedremo le funzioni per la gestione delle date.

LEZIONE 8: Funzioni: Parte 2

Con php è possibile mostrare la data corrente(Ora e/o giorno e/o mese e/o anno), sotto qualsiasi forma.

La funzione chiave è **date();** .

Ecco a voi una tabella di utilizzo:

d giorno del mese numerico 01-31
D giorno della settimana in abbreviazione di 3 caratteri
m mese numerico 01-12
M mese in abbreviazione di 3 caratteri
F mese in parola
Y anno a quattro cifre
y anno a due cifre
H ore 00-24
h ore 00-12
i minuti
s secondi

I valori che vedete sopra sono i valori da inserire nella funzione date, in questo modo date("valore");
Quindi, ad esempio se vogliamo stampare giorno, mese e anno corrente:

Codice:
```
echo date("d.m.Y");
```

Se vogliamo mostrare ore, minuti e secondi correnti:

Codice:
```
echo date("h.i.s.");
```

e cosi' via (Possiamo mettere insieme tantissime combinazioni, utilizzando i valori della tabella sovrastante)

Funzione Time();

Il concetto fondamentale alla base della manipolazione del tempo con PHP è il timestamp ovvero il numero di secondi trascorsi dal 1 gennaio 1970 00:00:00 (la cosiddetta Unix Epoch) all'istante specificato.

Per fare ciò usiamo time:

Codice:

```
<?php
echo time();
?>
```

Checkdate();

La funzione **checkdate** serve a controllare la validità e l'esistenza di una data.

Ad esempio:

Se noi scriviamo:

Codice:

```
<?php
if (checkdate(8, 32, 1995)){
echo "La data esiste!";
} else {
echo "La data non esiste!";
}
?>
```

E' normale che il testo che visualizzeremo sarà "La data non esiste!"

se noi scriviamo

Codice:

```
<?php
if (checkdate(1, 1, 1995)){
echo "La data esiste!";
```

```
} else {
echo "La data non esiste!";
}
?>
```

Il testo che apparirà sarà "La data esiste!";

Questa funzione può sembrare inutile, ma invece risulta molto utile per controllare, ad esempio i dati provenienti da un form:

Codice:

```
<?php
if (checkdate($_POST['mese'], $_POST['giorno'], $_POST['anno'])){
echo "La data che hai immesso nel form non esiste!!";
} else {
echo "La data non esiste!";
}
?>
```

(ps. ovviamente il codice sopra manca della parte fisica del form, la parte html).

Nella prossima lezione parleremo delle funzioni per la gestione dei files.

LEZIONE 9: Funzioni: Parte 3 e $_FILES

Con php è possibile creare file,
eliminarli, modificarli, creare cartelle, eliminare files da cartelle e tante altre cose.

Ecco a voi la lista delle principali funzioni(ed esempi di utilizzi) per la gestione dei files.

fopen($filedaaprire, $mode); (Apre un file)

fread($filedaleggere); (Legge un file)

unlink($daeliminare); (Elimina un file)

file_exists($dacontrollare); (Controlla se un file esiste)

is_writable($dacontrollare) (Controlla se un file è rescrivibile)

is_readable($dacontrollare) (Controlla se un file è leggibile)

is_file($dacontrollare); (Ha la stesso uso di file_exists, controlla se è un file è un file, cioè se esiste)

fwrite ($testodascrivere, $fileincuiscrivere); (Scrive un file, eliminando pero' il vecchio contenuto)

file_get_contents (Legge un file, si puo' usare al posto di fread, ma non ha bisogno di mettere fopen prima(a differenza di fread))

fputs (Identico a fwrite)

fclose($filedachiudere) (Chiude un file)

Iniziamo considerando fread(oltre che fclose e fopen)

Premettiamo una cosa.

Come avrete notato fopen ha il parametro $mode. Di solito si riempie w+ (Quindi la funzione si usa cosi': fopen("file.txt", "w+");).

Quel parametro sta ad indicare la modalità di apertura file.
- Scrivendo w+ potrete sia leggere che scrivere il file
- Scrivendo w potrete solo scriverlo
- Scrivendo r potrete solo leggerlo

Ma andiamo al sodo.

Usando **fread** (e fopen, fclose) possiamo leggere il contenuto di un file di testo(php, txt, html) con php.

Ecco il codice:

Codice:

```
<?php

$fp = fopen("nomefile.txt", "w"); //Per prima cosa apriamo il file,
scrivendo come modalità "w", in modo da poter leggerlo

$contenuto_file = fread($fp); //Dopodichè "leggiamo" il file e diamo alla
variabile $contenuto_file come valore il risultato della funzione fread

fclose($fp); //Chiudiamo il file, serve più che altro per motivi di
sicurezza.

?>
```

Adesso consideriamo la funzione **unlink** e la funzione **file_exists**.

Codice:

```
<?php

if (file_exists("pagina.html")){ //Se il file pagina.html esiste
unlink("pagina.html"); //Elimino il file pagina. html
} else { //Altrimenti
echo "Il file non esiste!";
}
?>
```

Adesso parliamo di **fwrite**.

Con questa funzione è possibile scrivere un file, eliminando però il vecchio contenuto.

Codice:

```
<?php

$fp = fopen("nomefile.txt", "w+"); //Per prima cosa apriamo il file, scrivendo come modalità "w", in modo da poterlo sia leggerlo che scriverlo

fwrite($fp, "Ciao");

fclose($fp); //Chiudiamo il file, serve più che altro per motivi di sicurezza.


?>
```

Adesso parliamo della variabile predefinita **$_FILES**.

A cosa serve?

Semplice.

Per gestire i files provenienti da un form.

In parole povere: serve per fare un sistema di upload.

In un sistema di upload gli elementi fondamentali sono i seguenti:

- Form html con campo file e enctype multipart/form-data

- Funzione move_uploaded_file

- Variabile $_FILES

$_FILES['nomecampofile']['name']; //Mostra il nome "vero" del file caricato dall'utente con tanto di estensione

$_FILES['nomecampofile']['tmp_name']; //Mostra il nome temporaneo del file caricato dall'utente

$_FILES['nomecampofile']['type']; //Mostra il tipo di file caricato

In pratica(Ecco un codice completo per l'upload):

Codice:

```
<?php

echo "<form action='pagina.php'  enctype='multipart/form-data'
method='post'>";
echo "<input type='file' name='upload'/>";
echo "<input type='submit' name='carica' value='Carica File'/>";
echo "</form>";

if ($_POST['carica']){

//Se il tipo di file è un'immagine jpg o una gif o una png
if (($_FILES['type'] == "image/jpeg") or ($_FILES['type'] == "image/gif")
or ($_FILES['type'] == "image/png"))
{
move_uploaded_file($_FILES['file']['tmp_name'], $_FILES['file']['name']);
//Con questa funzione il file verrà caricato dal computer dell'utente
allo spazio web.
} else { //Altrimenti
echo "Estensione non consentita";
}
}

?>
```

Nella prossima lezione parleremo del ciclo while e chiuderemo il capitolo della gestione dei files parlando delle funzioni per eseguire operazioni sulle cartelle e parlando di operazioni complesse con i files.

LEZIONE 10 Funzioni: Parte 4 e While

Con php è possibile anche svolgere **operazioni su cartelle**.

Le funzioni principali sono le seguenti:

mkdir(); (FUnzione che consente di creare una cartella)

rmdir(); (Funzione che consente di eliminare una cartella)

is_dir(); (FUnzione che controlla se una cartella esiste)

opendir(); (Apre una cartella)

readdir (); (Legge una cartella, per usare questa funzione bisogna mettere opendir all'inizio e closedir alla fine. LO stesso meccanismo del trio: fopen, fread, fclose.)

closedir(); (Chiude una cartella)

Proviamo ad utilizzare in un solo esempio le prime tre funzioni:

Codice:

```
<?php

if (!is_dir("prova")){ //Se prova non è una cartella, dunque se la
cartella "prova" non esiste
mkdir("prova"); //Crea la cartella "prova"
}

rmdir("prova");  //Dopodichè la eliminiamo

//Questo script è controsenso, ma l'ho fatto per farvi capire come si
usano tutte e tre le funzioni :XD

?>
```

Con tutto ciò che abbiamo imparato (FUnzioni per gestire files e cartelle e variabile $_FILES) possiamo sviluppare degli script complessi ragionando un po' che eseguono delle operazioni particolari,
impossibili da eseguire con funzioni predefinite del php.

Non è difficile svilupparne, basta usare la logica.

Prima, pero' dobbiamo parlare del **ciclo while**.

Il ciclo while viene utilizzato principalmente:

- Per visualizzare piu' dati provenienti da un database mysql

- Per "ciclare/visualizzare" dati provenienti da files e cartelle

Vedremo, adesso il secondo aspetto.

Ecco un'esempio di utilizzo di ciclo while.

Adesso, sviluppiamo uno script che mostra tutti i files presenti una cartella. (Lo commento ben benino)

Codice:

```
<?php
$op = opendir("cartella"); //Apro la cartella, definisco la variabile
$op, la variabile op è uguale al risultato della funzione opendir

while($file=readdir($op)){ //Apro il ciclo while, definisco la variabile
file, la variabile file è uguale al risultatodella funzione readdir.
Readdir opera leggendo il risultato della funzione opendir( Cioè la
variabile $op)
echo $file; //Stampo il nome del file(La variabile file, cioè)
echo "<p></p>"; //Mettiamo uno spazio in modo che i files verranno
visualizzati uno sotto l'altro ordinatamente
} //Chiudo il ciclo

closedir("cartella"); //Chiudo la cartella, questa funzione serve più che
altro per motivi di sicurezza
?>
```

Nella prossima lezione parleremo di cookie e sessioni con php,
dopodichè passeremo finalmente a parlare dei database mysql.

LEZIONE 11: Sessioni e Cookie

Le sessioni e i cookie sono entrambe variabili speciali in cui è possibile immagazzinare dati per un certo periodo di tempo.

Se io creo una variabile normale,
il valore varrà solo per la pagina in cui la variabile è definita;

invece se setto un cookie o creo una sessione il valore di quel cookie e di quella sessione varrà per un certo tempo(anche lungo) in tutte le pagine del mio sito.

Iniziamo parlando dei **cookie**.

Per creare un cookie si usa una semplice funzione di nome setcookie();
(La funzione setcookie ha la seguente sintassi:

setcookie("nomecookie", "valorecookie", "tempocookie");
)
,

dopodichè per richiamare il cookie basterà stampare la variabile $_COOKIE(In questo modo:

Codice:

```
<?php
echo $_COOKIE['nomecookie'];
?>
```

)

Ma andiamo nella pratica.

Codice:

```
<?php

$name="Matteo";

setcookie("nome", $name, time() + 3600); //Creo un cookie che durerà 3600 secondi, che si chiama "nome" e che ha come valore il valore della variabile $name, cioè Matteo


echo $_COOKIE['nome']; //Mostro il valore del cookie
```

```
?>
```

Adesso ogni volta che in una pagina del mio sito scrivero',
mi apparirà la scritta "Matteo", finchè il cookie non scadrà(Cioè fra 3600 secondi).:

Codice:

```
<?php
echo $_COOKIE['nome'];
?>
```

Per "distruggere" il cookie, cioè eliminarlo prima della scadenza.

Si può impostare un tempo di durata "negativo":

Codice:

```
<?php
setcookie("nome", "", time() - 3600);
?>
```

Parliamo adesso delle **Sessioni**.

In realtà cookie e sessioni si possono usare indistintamente.

Si chiamano e si usano diversamente, ma in pratica servono alla stessa cosa.

Per le sessioni si usano alcune funzioni:

session_start(); (Questa funzione bisogna usarla obbligatoriamente prima di settare una funzione)

session_destroy(); (Con questa funzione vegono distrutte/eliminate tutte le sessioni)

Una sessione si definisce in questo modo:

$_SESSION['nomesessione'] = "valore";

e si richiama in questo modo:

Codice:

```
<?php
echo $_SESSION['nomesessione'];
?>
```

Per capirci meglio, ecco il codice completo:

Codice:

```
<?php

session_start(); //Funzione che "crea un terreno" adatto alla definizione
delle sessioni

$_SESSION['username'] = "Matteo"; //Stabiliamo la sessione di nome
username

echo $_SESSION['username']; //Mostriamo il valore della sessione username

?>
```

Se vogliamo distruggere la sessione, come prima anticipato:

Codice:

```
<?php
session_start();
session_destroy();
?>
```

Tramite le sessione e i cookie che abbiamo appena di visto, con un po' di logica si possono sviluppare sistemi di registrazione e login, ma anche semplici aree private, sfruttando i dati provenienti da form($_POST)

pagina.php

Codice:

```
<form action="pagina.php" method="post">
<input type="text" name="username"/>
<input type="password" name="pass"/>
<input type="submit" name="login" value="Login"/>
</form>

<?php

if ($_POST['login']){

$password="123"; //Password per il login

if  ($_POST['pass'] == $password){
//Se il dato proveniente dal campo username del form e il dato è uguale
alla variabile username e il dato //proveniente dal campo pass del form è
uguale alla variabile password

setcookie("username", $_POST['username'], time() + 3600);
//Crea un cookie di nome username, che contenga l'username inserito nel
campo username del form

setcookie("login", "ok", time() + 3600);
//Crea un cookie di nome login, di valore "ok" e che duri 3600 secondi

} else { //Altrimenti
echo "Username e/o Password errati. Riprova"; //Fai apparire questo
messaggio
}

}


?>
```

Nella prossima lezione incomincieremo a parlare di database mysql.

LEZIONE 12: MYSQL

In questa lezione conclusiva vedremo insieme come è possibile far interagire le nostre pagine .php con i database MySql.
Per dovere di completezza non possiamo non ricordare che PHP è in grado di connettersi a diversi database server (MySql, MS Access, PostgreSql, Oracle, Microsoft Sql Server, Sybase,...) tuttavia noi ci limiteremo a vedere l'interazione con MySql che è senza dubbio la soluzione più comune e diffusa.

MySql è un database veloce e potentissimo in grado di gestire applicazioni con un elvato grado di criticità e, cosa non secondaria, è un software open source, liberamente scaricabile dal sito www.mysql.com.

Come abbiamo accennato nella lezione precedente PHP mette a disposizione dello sviluppatore diverse funzioni per interagire con i db MySql. Vediamo insieme le più importanti.

Per prima cosa vediamo come fa PHP a connetersi al MySql Server.
Allo scopo soccorre la funzione **mysql_connect()** che si utilizza con la seguente sintassi:

```
mysql_connect(server, utente, password);
```

Ad esempio:

```
$myconn = mysql_connect('localhost', 'pippo', 'xxxxxx') or die("Errore...);
```

Una volta stabilita la connessione è necessario selezionare uno specifico db sul quale lavorare. A questo scopo PHP ci fornisce la funzione **mysql_select_db()** da utilizzarsi con la seguente sintasi:

```
mysql_select_db(database, connessione);
```

Ad esempio:

```
mysql_select_db('mio_database', $myconn) or die("Errore...");
```

Per prima cosa vediamo come è possibile recuperare dei dati presenti nel nostro database.
Per fare questo dobbiamo formulare ed eseguire una *query*, la quale consiste in una interrogazione che lo sviluppatore rivolge al database. Per fare ciò si fa ricorso alla funzione **mysql_query()** con la seguente sintassi:

```
mysql_query(query, connessione);
```

Ad esempio:

```
$query = "SELECT * FROM tabella";
$result = mysql_query($query, $myconn) or die("Errore...");
```

Facciamo ora **un esempio completo**.
Poniamo di voler recuperare dalla tabella "amici" una serie di dati (nome, cognome e telefono) e di volerli stampare a video per ogni occorrenza trovata nel nostro database.
Ecco il codice completo opportunamente commentato:

```
<?
```

```
//Mi connetto al MySql Server
$myconn = mysql_connect('localhost', 'pippo', 'xxxxxx') or die("Errore...);

//Mi connetto al database degli amici
mysql_select_db('database_degli_amici', $myconn) or die("Errore...");

//Imposto ed eseguo la query
$query = "SELECT nome, cognome, telefono FROM amici";
$result = mysql_query($query, $myconn) or die("Errore...");

//conto il numero di occorrenze trovate nel db
$numrows = mysql_num_rows($result);

//se il database è vuoto lo stampo a video
if ($numrows==0){
  echo "Database vuoto!";
}
//Se invece trovo delle occorrenze...
else
{
  //Avvio un ciclo for che si ripete per il numero di occorrenze trovate
  for($x=0; $x<$numrows; $x++){
  //Recupero il contenuto di ogni record rovato
    $resrow = mysql_fetch_row($result);
    $nome = $resrow[0];
    $cofgnome = $resrow[1];
    $telefono = $resrow[2];

    //Stampo il risultato
    echo "nome: <b>" . $nome . "</b><br/>";
    echo "cognome: <b>" . $cognome . "</b><br/>";
    echo "telefono: <b>" . $telefono . "</b>";
  }
}
?>
```

Quello che abbiamo fatto qui sopra dovrebbe eservi abbastanza chiaro (ricordate la lezione sui cicli?)...
Resta pertanto da chiarire solo il significato di due funzioni specifiche che abbiamo utilizzato:

- **mysql_num_rows()**
  Serve per conteggiare il numero di records trovati all'interno del nostro db sulla base di una data query;
- **mysql_fetch_row()**
  Recupera il contenuto dei records trovati. Più precisamente restituisce una array contenente i valori di ogni campo riscontrato nel recordset.

Cenni di SQL: INSERT INTO, UPDATE e DELETE
Per finire vediamo brevemente come eseguire altre importanti operazioni con i database attraverso i più comuni comandi del linguaggio SQL.

Con **INSERT INTO** si inseriscono nuovi dati nel db, con **UPDATE** si aggiornano dei dati già presenti, con **DELETE** si cancellano dei dati.

Dal punto di vista di PHP queste operazioni non differiscono tra loro, l'unica cosa che cambia è la query che viene eseguita, ma questo discorso attine al linguaggio SQL.
Facciamo degli esempi:

Per INSERT INTO useremo:

```
$query = "INSERT INTO tabella VALUES('valore1','valore2','valore3')";
```

Per UPDATE useremo:

```
$query = "UPDATE tabella SET campo1='valore1', campo2='valore2',
campo3='valore3' WHERE id = 1";
```

Per DELETE useremo:

```
$query = "DELETE FROM tabella WHERE id = 1";
```

Attenzione!
Se nelle query di UPDATE e DELETE non usiamo la clausola "WHERE" verranno aggiornati/eliminati tutti i record del db!

Facciamo un esempio di utilizzo di queste query; vediamo come cancellare con PHP un record dal nostro database MySQL:

```
<?
//Mi connetto al MySql Server
$myconn = mysql_connect('localhost', 'pippo', 'xxxxxx') or die("Errore...");

//Mi connetto al database degli amici
mysql_select_db('database', $myconn) or die("Errore...");

//Imposto ed eseguo la query
$query = "DELETE FROM tabella WHERE id = 1";
$result = mysql_query($query, $myconn) or die("Errore...");
?>
```

Per le altre operazioni basterà sostituire la query mantenendo inalterata la struttura del PHP.

LEZIONE 13: Le Classi & Note Finali

Le classi servono per raggruppare le funzioni.

Esempio di definizione di classe:

Codice:

```
<?php

class database { //Definiamo la classe database

//Scriviamo le funzioni contenute nella classe database:

function connettiti($username, $host, $password, $db_name){
mysql_connect($host, $username, $password);
mysql_select_db($db_name);
}


function delete($table){
mysql_query("DELETE FROM $table");
}

function showtable($table){
mysql_query("SHOW TABLE $table");
}

}
?>
```

Ad esempio per utilizzare la classe e quindi richiamarla

Codice:

```
<?php

$db= new database; //La variabile $db è uguale a "un nuovo richiamo "
della classe database

$db->connettiti("blizardcms", "localhost", "", "my_blizardcms");
//Richiamiamo la funzione connettiti dalla classe attribuita alla
variabile $db, cioè la classe "database"
```

```
$db->delete("tabella");

//e cosi' via..

?>
```

Considerazioni Finali

In questa guida ho cercato di trasmettere a voi utenti principalmente le basi del php.

Una volta imparati i concetti chiave è il programmatore che, seguendo una logica deve riuscire a mettere il tutto insieme sviluppando delle applicazioni (O come si chiamano nello specifico per il web: scripts).

Vi linko alcuni siti dove potete trovare scripts pronti, anche da poter modificare o semplicemente "osservare" il codice:

http://www.mrwebmaster.it/script/script-php

http://php.html.it/script

E, ad esempio sul blog di matteoiamma community nella categoria "Web-Engineer/Php e Mysql" potete trovare un sacco di tutorials sul php:

http://www.matteoiammarrone.com/public/blog.php?cat=3

nell'area forum di matteoiamma laboratory

http://www.matteoiammarrone.com/public/php-e-mysql-f3.html