Burning PHP

1)Che cosa e' PHP?
2)Che cosa posso fare con PHP?
3)Cosa mi serve per programmare in PHP?
4)Cosa impareremo da questo corso?

Burning PHP

1) PHP (acronimo ricorsivo per "PHP: Hypertext Preprocessor") e' un linguaggio di scripting Open Source, lato server, che puo' essere integrato con altri linguaggi di programmazione/scripting.

2)Posso creare applicazione web (siti, portali), oppure dare comandi sul mio sistema operativo (riga di comando), o anche creare interfacce grafiche (gui) per altri programmi.

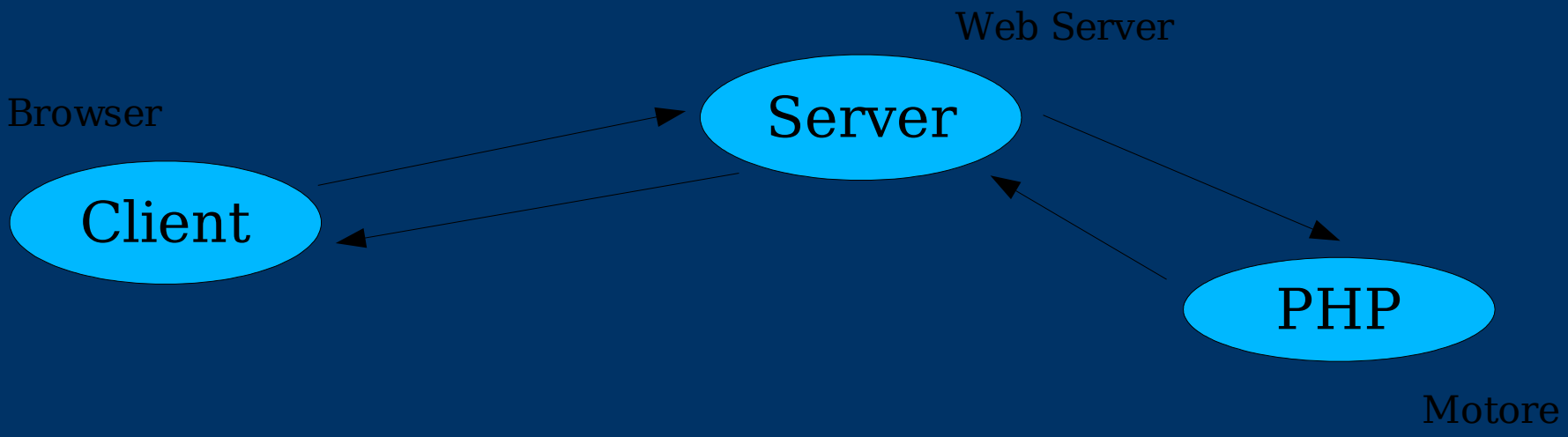
3)Necessitiamo di un server web e del motore PHP! PHP gira su diversi servers e su diversi sistemi operativi. In questo corso useremo un ambiente di lavoro composto da apache 1.3.28, php 4.3.4, mysql 4.01salcazzo su slackware Linux 10.1.

Burning PHP

4)Gli obiettivi di questo corso sono di apprendere le basi del linguaggio, essere in grado di scrivere semplici applicazioni I/O, con interfacciamento ad un database e altro, saper reperire risorse sul web, comprendere la fantastica filosofia del free software.

Burning PHP

Comunicazione Client – Server nel processare files .PHP



1)I client fa una richiesta al web server (una pagina .php)
2)Il server capisce che gli si sta chiedendo una pagina .php e richiama il suo processatore di php.
3)Php compila lo script presente in quella pagina e inoltra il risultato al web server che lo rida' al client
4)Il client riceve il risultato che sara' una semplicissima pagina HTML (vedi codice sorgente).

Burning PHP

Risorse Web:

Php / Apache /MySQL

www.php.net
www.apache.org
www.mysql.com

Php su Linux

- http://www.latoserver.it/php/install/install-linux.php3

Php su Windows

- http://www.latoserver.it/php/install/install-windows.php3
- http://freephp.html.it/guide/lezioni.asp?idguida=7

Risorse TCP/IP

- http://www.warriorsofthe.net/

Risorse GNU/Linux

http://a2.pluto.it/

Burning PHP

Sintassi – Tipi di dato – Operatori logici
Variabili – Elementi di programmazione

Sintassi:
- Php tags / Commenti / Statements / Caratteri speciali

Tipi di dati:
- Semplici / Composti / Speciali

Variabili:
- Dichiarazione / Inizializzazione
- Utilizzo delle variabili

Operatori:
- Matematici / Logici

Elementi di programmazione:
- If/else/elseif
- switch
- for – while (do-while) –foreach

Burning PHP

Sintassi

-Il codice PHP puo' essere a stretto contatto con altri tipi di codice come HTML e javascript percui e' importante che si capisca cosa e' cosa. Per questo si usano i tag “<?PHP” e “?>” di apertura e chiusura, tutto cio' che e' dentro questi tag viene trattato come codice PHP.
Inoltre vedremo come sra' lo stesso php che scrivera' il codice html.

-Ci sono 2 tipi di commenti in php , come in tutti i linguaggi:

“ // “ e' un commento su una riga
“ /* */” e' un commento su piu' righe

-Uno statement e' una riga di codice, o un insieme di righe, in php ogni statement e' seguito dal carattere di “ ; “.

-Esistono dei caratteri speciali, cioe' caratteri che definiscono delle proprieta' ():
$ indica una variabile;
“ le virgolette doppie indicano dei dati statici;
' le virgolette singole indicano dei dati statici (mostrare le diffferenze);
\ backslash indica di ignorare il carattere successivo;
. il punto e' l'operatore di concatenzione;
>>! il punto esclamativo e' l'operatore di negazione;

Burning PHP

Tipi di dati

Spiegazione sui titi di dato, casting, etc

I tipi di dato semplici utilizzati sono:

- boolean (TRUE [1] o FALSE[0])
- integer (numeri naturali)
- float (numeri reali, aka 'double')
- string (caratteri alfanumerici)

I tipi di dati composti:

-array (matrici)

-oggetti (classi) non le toccheremo :(

I tipi di dato speciali:

-NULL

Burning PHP

VARIABILI

Le variabili sono contenitori di dati e vengono riconosciute dal simbolo $ davanti al loro nome. In Php al contrario di altri linguaggi di programmazione come il C o il Java non c'e' bisogno di dichiarare le variabili (casting), che significa comunicare al compilatore di che tipo di variabile si tratta.
In Php la dichiarazione avviene allo stesso momento dell'inizializzazione, cioe' valorizzare una variabile, e l'operatore che si utilizza e' “=”, con i semplici statement:

```
$pippo = 3;
$pluto = “cane”;
```

Valoriziamo la variabile di nome pippo a 3 e pluto a “cane”.
Ma che tipi di dato sono? Per sapere con che tipo di dato abbiamo a che fare esiste la funzione “gettype” usata in questa maniera:

```
echo gettype($pippo);
echo gettype($pluto);
```

Se si vuole esplicitare il tipo di dato che una variabile deve contenere lo si fa cosi:

```
$minni = (boolean) TRUE;
echo gettype($minni);
```

Burning PHP

VARIABILI

Gli array, sono tipi di dato composti, matematicamente simili alle matrici.
Un array e' un contenitore di contenitori ovvero una variabile a cui e' possibile associare piu' valori ognuno distinto dagli altri attraverso una chiave.
Per istanziare un array si usa l'istruzione array:

```
$lista = array();
echo gettype($lista); //controlliamo che tipo di variabile e'
```

Gliarray possono essere composti da numerosi elementi al proprio interno ed anche da un altro array come elemento, nel caso non esistino altri array allora viene denominato monodimensionale, cioe' formato da una sola riga e piu' elementi, nel caso ci sia pure un array come elemento allora si definisce multidimensionale.

```
$lista_monodimensionale = array(“carote”, “mele”, “pasta”);
$lista_multidimensionale = array(“verdure” => array(“carote”,”melanzane”),
“secco” => array(“pasta”, “fagioli”, “ceci”),
extra => array(“olio”, “aceto”));
```

Come possiamo vedere le chiavi nel primo array non sono definite, percui vengono definite chiavi numeriche a partire da 0, nel secondo invece abbiamo definito delle chiavi string per l'array principale mentre rimangono numeriche per gli array interni.
Le chiavi servono ad accedere al valore di un particolare elemento dell'array

Burning PHP

VARIABILI

Se volessimo accedere al valore del secondo elemento dell'array lista_monodimensionale dovremmo riferirci alla sua chiave inserita tra parentesi quadre.

```
Echo $lista_monodimensionale[1]; //stampera' “mele”
```

occhio che le chiavi (o indici) degli array partono da 0.

```
echo $lista_multidimensionale['secco']['0'] //stampera' pasta
```

Ora che abbiamo visto come accedere ad un elemento di un array vediamo come inserirne di nuovi in un array gia' istanziato. Se omettiamo la chiave automaticamente verra' inserita una chiave numerica che sara' incrementata di 1 rispetto l'ultima presente, occhio alla doppia parentesi quadra aperta e chiusa o con all'interno la chiave che desideriamo.

```
$lista_monodimensionale[] = “olio”; //olio avra' la chiave 3
$lista_multidimensionale['vino'] = array(“bianco”, “rosso”);
```

Per poter visualizzare il contenuto di un array usiamo la funzione print_r(nome_array), usatela solo per debug perche' non e' formattabile:

```
print_r($lista_monodimensionale);
print_r($lista_multidimensionale);
```

Burning PHP

OPERATORI ARITMETICI

Tra variabili di tipo integer/double/float, ma non solo, si possono eseguire le piu' semplici operazioni matematiche, somma e sottrazione, moltiplicazione e divisione. Un' operazione di questo genere tra due tipi string che sono in realta' numeri dara' una nuova variabile di tipo integer.

Un'altra operazione interessante e' anche il calcolo del resto di una divisione che si fa cosi:

```
$num = 10;
$resto = $num%3;
echo $resto; //dara' il resto della divisione (1)
```

Burning PHP

OPERATORI DI CONFRONTO

Come avete potuto vedere per inzializzare una variabile abbiamo usato l'operatore “ = “,
che valorizza una variabile ma non e' l'operatore di uguaglianza. Per sapere se una variabile e' uguale a un'altra:

```
$minni == $pippo;
```

Lo statement sopra ci ritorna il valore TRUE o FALSE.
Gli altri operatori logici usati sono:

```
$minni != $pippo;
```
(diverso)

```
$minni <= $pippo;
```
(ninore o uguale)

```
$minni >= $pippo;
```
(maggiore o uguale)

Gli ultimi 2 usati anche singolarmente “ > “ , “ < “ .

OPERATORI LOGICI

Gli operatori logici sono usati per unire delle comparazioni (), se volessimo per esempio sapere se 2 variabili sono uguali e altre 2 sono una maggiore dell'altra dovremmo usare gli operatori logici per “concatenare” i 2 assiomi precedenti.

```
$minni == $pippo AND $pluto<$topolino;
```

In questo caso i 2 postulati devono essere VERI entrambi.
Nel caso necessitassimo solo che uno dei due sia vero useremo lo'operatore OR

```
$minni == $pippo OR $pluto<$topolino;
```

AND puo' essere sostituito da &&, OR invece da || (doppio pipe)

Ora vedremo a cosa servono questi operatori e queste comparazioni

Burning PHP

ELEMENTI DI PROGRAMMAZIONE

Entriamo nella programmazione vera e propria e conosciamo gli elementi comuni a tutti i linguaggi che utilizzeremo nel nostro codice.

STRUTTURE DI CONTROLLO (IF-ELSE-ELSEIF)

```
if($pippo==$minni){
 //esegue questo statement;
            }

 if($pippo!=$minni){
  //esegue questo statement;
          }else{
//esegue quest'altro statement;
            }

  if($pippo==$minni){
   //esegue questo statement;
}elseif($pippo!=$pluto){
 //esegue quest'altro statement;
            }
```

E cosi all'infinito....

ELEMENTI DI PROGRAMMAZIONE

STRUTTURE DI CONTROLLO (SWITCH)

Al posto dell'ultima porzione di codice, che compara pippo a due variabili diverse, sarebbe piu' opportuno usare la struttura SWITCH:

```
switch($pippo){
case 'valore1':
    break;
case 'valore2':
    break;
default:
    break;
    }
```

Se nessuno dei valori indicati e' uguale a $pippo allora verra' eseguito il codice dentro default. Il costrutto “break” serve per uscire dal ciclo ed e' usato con qualsiasi ciclo.

Burning PHP

ELEMENTI DI PROGRAMMAZIONE

STRUTTURE LOOPS (while)

Delle strutture parecchio comode nella programmazione sono i cosiddetti “cicli”, cioe' strutture che compiono ciclicamente del codice.
La piu' semplice e' while:

```
while($statement){
//se statement torna TRUE
allora viene eseguito il codice
}
```

Simile a while e' do – while

```
do{
//esegue qeusto codice almeno una volta
}while($statement);
```

La differenza sostanziale e' il momento in cui viene fatto il controllo, nel primo caso prima di eseguire il codice nel secondo dopo averlo eseguito almeno una volta.

Burning PHP

ELEMENTI DI PROGRAMMAZIONE

STRUTTURE LOOPS (for)

For,

```
$valoremassimo = 10;
for($i=0; $i<$valoremassimo; $i++){
    //esegue il codice
}
```

Il primo elemento $i=0 indica da dove parto,
$i<$valoremassimo e' l'evento percui il ciclo si deve fermare,
$i++ vuol dire che aumento il valore di $i (integer) di un'unita' ad ogni passaggio.
Per uscire prima di raggiungere il valore massimo si puo' usare il costrutto break:

```
$valoremassimo = 10;
for($i=0; $i<$valoremassimo; $i++){
    //esegue il codice
      if($pippo==valore){
          break;
        }
    }
```

Break e' utilizzabile all'interno di qualsiasi blocco di codice racchiuso tra parentesi graffe, solitamente cicli.

Burning PHP

ELEMENTI DI PROGRAMMAZIONE

STRUTTURE LOOPS (foreach)

Foreach e' un ciclo particolare usato per scorrere gli array:

```
$lista = array();
foreach($lista as $key => $value){
    echo $key;
    echo $value;
    }
```

Se l'array e' multidimensionale $value sara' uguale ad un array a sua volta e quindi puo' essere scorso anch'esso con foreach.

Burning PHP

Web References

http://www.phpbook.it/
Un nuovo wiki/libro scritto dalla comunita' php ziobudda.net
in italiano

Burning PHP

Esercizio

Dobbiamo fare la spesa per un numero fisso di persone.
Al mercato dovete comprare 4 prodotti diversi a prezzi differenti
1-Visualizzate alla fine la lista della spesa e quanto e' la quota per ogni persona e la quota totale.Calcolate 1 kg di ogni cosa.

Dati:

- pomodori 1 euro al kg
- melanzane 0,50 al kg
- pasta 1 al kg
- mozzarella 2 al kg

2-E se i pomodori non fossero disponibili?

Burning PHP

Variabili locali, globali, superglobali / Costanti / Include – Require/ Input – Output / Interazione con HTML

Le variabili usate fino ora sono definite variabili locali, perche' la “vita” di queste variabili risiede negli script php che fate e in quelli che includete dopo averle valorizzate.

La “vita” o visibilita' di una variabile (in inglese SCOPE di una variabile) e' estremamente importante come concetto di programmazione. Con visibilita' intendiamo che il valore di quella variabile ci e' accessibile e possiamo quindi utilizzarlo.

In progetti php complessi vedremo che il codice e' suddiviso in pagine diverse e molte volte il contenuto di una pagina deve essere accessibile ad un'altra per poter funzionare. In questi casi si usano essenzialmente 2 funzioni: Include e Require.

```
<?PHP
require(“config.inc”);
include(“functions.php”);
?>
```

La differenza fondamentale nell'utilizzo di require o include e' la maniera in cui i due trattano gli errori, require genera un fatal error mentre include solo uno warning. Un fatal error comporta il blocco del codice e il messaggio d'errore, i warning invece sono delle semplici avvertenze.

Burning PHP

Come dicevamo le variabli in php sono visibili anche dentro le porzioni di codice incluso:

pagina stampa.php:

```
$a = 5;
```

pagina che include stampa.php:

```
include("stampa.php");
echo $a;
```

Altro tipo di varibili sono le cosiddette SUPERGLOBALI, variabili predefinite in php, tra queste troviamo $_GET , $_POST, $_REQUEST, $_FILES, $_SERVER.
Queste variabili sono arrays e contengono valori diversi a seconda a chi vi riferite (ricordate phpinfo?), per ora trattiamo GET, POST e REQUEST.
Un passaggio di variabili con il metodo "GET" e questo:

http://www.torre.com/index.php?cmd=sfoglia&par=titolo&pag=1&inf=0

In quest'ultimo esempio i valori dopo il ? Sono passati in modalita' GET e quindi visibili dentro la barra degli indirizzi del nostro browser .... poco sicuro ....

Burning PHP

```
<form name=”iscrizione” method=”POST” action=”index.php”>
<input type=”text” name=”nome” value=””>
<input type=”password” name=”password” value=””>
<input type=”submit” name=”bottone”  value=”Login”>
</form>
```

Quest'altra maniera di passare i dati, classica delle form HTML e' invece quella per metodo POST (), i dati che vengono passati non saranno visualizzabili nella barra degli indirizzi ... ma non per questo sicuro .....

Indovinate un po' cosa riceveranno le pagina di risposta?

Nel primo caso (GET):
stiamo passando alla pagina index.php le variabili cmd, par, pag, inf i valori di queste saranno accessibili con la sintassi:

```
echo $_GET['cmd'];
```

e se scrivessimo:

```
print_r($_GET);
```

Nel caso POST come potete ben immaginare invece le variabili le leggiamo cosi:

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

Burning PHP

Se per esempio non sappiamo in quale maniera ci arrivera' quella variabile possiamo usare l'array $_REQUEST che incorpora sia $_GET che $_POST.

Lasciamo da parte per ora $_FILES che come dice il nome stesso tratta di files uplodati via HTTP (immagini, video, etc) che toccheremo durante le funzioni sul filesystem.

Per presentare le variabili GLOBALI dobbiamo prima presentare le FUNZIONI.
Come in tutti i linguaggi di programmazione, in php si possono scrivere funzioni che espletano diversi compiti, oltre alle funzioni predefinite del php posso crearne di nuove a mio piacere.
La sintassi per dichiarare una nuova funzione e':

definizione:

```
function CalcolaAreaTriangolo($base, $altezza){

        $area = ($base*$altezza)/2;
        return $area;
        }
```

utilizzo:

```
$area_triangolo = CalcolaAreaTriangolo($base, $altezza);
```

Il termine RETURN fa si che il valore di $area mi venga passato nell'altra variabile $area_triangolo.

Burning PHP

Nel precedente esempio la locale variabile $a definita fuori dalla funzione non sarebbe stata visibile all'interno della funzione e cosi avviene per le variabili che vengono definite dentro la funzione.

Se proprio si deve utilizzare una variabile dentro una funzione senza passarla come elemento si puo' usare il termine GLOBAL percui

```
$a = 5;
function CalcolaAreaTriangolo($base, $altezza){
GLOBAL $a;

$area = ($base*$altezza)/2;
$prodotto = $area*$a;
return $prodotto;
}

$numero = CalcolaAreaTriangolo($base, $altezza);
```

Burning PHP

COSTANTI

Le costanti sono trattate come le variabili superglobali , quindi si puo' accedere al loro valore in qualunque punto degli script. Per regola i nomi delle costanti possono avere numeri, underscore e tutti i caratteri ascii.

Per avere una lista delle costanti gia' predefinte nel php usiamo la funzione GET_DEFINED_COSTANTS(), altrimenti possiamo definire costanti usando la funzione DEFINE(), in questa maniera:

```
define('SALUTO', “Ciao Mondo”);
echo SALUTO;
```

Una volta definita una costante non puo' essere sovrascritta.
Per sapere se una costante e' stata definita si usa invece la funzione DEFINED(nome_costante)

```
if(defined('SALUTO')){
    echo SALUTO;
    }
```

In generale per sapere se una variabile e' stata definita e/o ha un valore, sia per costanti che non, si usa la funzione ISSET(nome_variabile)

Burning PHP

INTERAGIAMO CON L'HTML

Abbiamo gia' visto come il php generi codice html e come concatenare i due codici insieme. Ma l'HTML e' anche il mezzo con cui passare dati in INPUT ai nostri script PHP(e l'abbiamo visto). Questa operazione avviene tramite le cosidette FORM.

```
<FORM NAME=”Prima_form” ACTION=”ricevi_dati.php” METHOD=”POST”>
<input type=”text” name=”username”>
<input type=”password” name=”password”>
</FORM>
```

Analizziamo il codice: il tag form accetta l'attributo NAME, utile anche nel linguaggio javascript, ACTION che direziona verso la pagina ricevi_dati.php che sara' colei in cui gestiremo i nostri dati in INPUT, METHOD che decide in che maniera passare i dati.

All'interno delle form si possono inserire parecchi altri tag: TEXT, CHECKBOX, RADIOBUTTON, SELECT, PASSWORD, FILE (vedremo in seguito), TEXTAREA.

Da qualsiasi di questi oggetti riceviamo un determinato input.

Facciamo un po' di esercizi....

Burning PHP

Webography:

HTTP
http://en.wikipedia.org/wiki/HTTP
http://it.wikipedia.org/wiki/HTTP
http://www.w3.org/Protocols/rfc2616/rfc2616

SUPERGLOBALS
http://it2.php.net/manual/it/reserved.variables.php#reserved.variables.request

COSTANTI
http://it2.php.net/manual/it/reserved.constants.core.php

Burning PHP

Funzioni Predefinite

Data/Ora
- -date, mktime, strftime, getdate

Mail
- -mail

Configurazione php
- -set_time_limit, error_reporting

Stringhe
- -semplici funzioni (substr, strpos, ucwords)
- -espressioni regolari (ereg, ereg_replace, preg_replace)

Array
- -output, operazioni su un array singolo
- -operazioni su 2 o + array

Http
- -header

Burning PHP

Esercizio

1. Creare una form di login con username e password,
cryptare sia username che password in un file incluso.
Se il login va a buon fine dare la possibilita' di mandare
un messaggio che verra' recapitato via mail.
Nella mail ci deve essere il mittente, l'ora e data in cui scrive la form
e tutto il testo del messaggio in testo semplice.

2. L'indirizzo della mail va scelto da un'altro campo
e va controllato se corrisponde a un indirizzo valido o no.

3. Al posto del login via form creiamone uno via http.

Burning PHP

Funzioni FileSystem

Quando abbiamo parlato dell'input di dati attraverso le form abbiamo tralasciato la parte relativa a come uplodare via HTTP dei files. Vi e' mai capitato di uplodare un'immagine su indymedia.org? Beh, e' tutto gestito da html e php.
Capiamo a cosa stare attenti e cosa cambia in una form se vogliamo uplodare files.

Innanzitutto l'intestazione:

```
<form name=”iscrizione” method=”POST” action=”index.php”
enctype="multipart/form-data">
<input type=”text” name=”nome” value=””>
<input type=”password” name=”password” value=””>
<input type="file" name="userfile">
<input type="hidden" name="MAX_FILE_SIZE" value="100000000">
</form>
```

Come potete notare abbiamo aggiunto nell'intestazione della form la parola chiave enctype, senza di questa la form non uplodera' niente sul server.
Dopodiche abbiamo inserito un input type FILE , che altro non e' che un campo testo con la possibilita' di sfogliare il nostro computer.

Burning PHP

Subito sotto abbiamo inserito anche un nuovo tipo di campo, HIDDEN, che e' un campo che non viene visualizzato nella nostra pagina PHP, ma nel sorgente si, quindi non metteteci le password :P, questo campo hidden definisce la dimensione massima che il campo file puo' inviare, se questa dimensione e' superata si avra' un errore direttamente in HTML, non contate troppo su questa direttiva perche' e' facilmente raggirabile.
A questo punto possiamo uplodare files sul server!

Come li gestiamo dopo averli uplodati?
Ecco che entra in scena un'altra variabile superglobals $_FILES.
Nella pagina che riceve l'input proviamo a scrivere:

```
print_r($_FILES);
```

Surprise! Ecco che leggiamo delle informazioni interessanti.
$_FILES['userfile']['name'] = nome del nostro file,
$_FILES['userfile']['size'] = dimensione
$_FILES['userfile']['tmp_name'] = nome nella cartella temporanea
$_FILES['userfile']['error'] = se sono successi degli errori , vedere la codifica
$_FILES['userfile']['type'] = che tipo di file abbiamo uplodato (mimetype), dipendente dal browser e da come lo gestisce.

Burning PHP

Dobbiamo comunque conoscere alcune configurazioni del php per poter usare bene questa funzionalita'. Leggiamo il nostro php.ini con phpinfo() soprattutto alle voci: FILE_UPLOADS, UPLOAD_MAX_FILESIZE, UPLOAD_TMP_DIR e POST_MAX_SIZE.

FILE_UPLOADS:

Ci dice se possiamo o no uplodare files.

UPLOAD_MAX_FILESIZE

Setta un limite massimo alla dimensione dei files uplodabili, vedere come si comporta l'interazione tra max_file_size in hidden e questa.

UPLOAD_TMP_DIR

Decidiamo in quale directory temporanea uplodare i files prima di salvarli sul filesystem del server.

POST_MAX_SIZE

Decide la dimensione massima di tutta una chiamata POST.

Burning PHP

A questo punto abbiamo parzialmente caricato il nostro file, perche' questo ora e' anche presente nella cartella tmp del server, ma non ancora utilizzabile ne' visualizzabile.

Il manuale php consiglia di usare la funzione MOVE_UPLOADED_FILE() per copiare dalla tmp al nostro filesystem il file in questione, anche perche' e' una funzione che non viene alterata da diversi settaggi del php.

Void move_uploaded_file($_FILES['tmp_name'], cartella_di_destinazione);

Naturalmente dobbiamo avere i permessi di scrittura su quella directory altrimenti avremo un messaggio di errore.

Per controllare che il file e' nella cartella di destinazione usiamo invece la funzione IS_UPLOADED_FILE(FILE) , file deve essere l'intero path?.

Bool is_uploaded_file($filename);

Abbiamo caricato il file sul server. :)

Burning PHP

Esercizio

Facciamo che siamo bastardi e logghiamo tutt i messaggi
che vengono inviati dalla nostra textarea.
Creaimo un file log.txt che contiene:
-indirizzo IP
-host
-browser
-data e ora
-testo messaggio
-destinatario

Ad ogni messaggio inseriamo in coda queste info.

Burning PHP

Connettiamoci al database!

Php interagisce facilmente con numerosi database (db): mysql, pgsql, oracle, mssql.
A cosa serve un database?
Un database come dice il nome stesso e' una BASE di DATI, cioe' una serie di dati raccolti ordinatamente all'interno di una struttura, questa struttura e' composta da tabelle, all'interno delle tabelle ci sono righe e colonne. Le colonne sono i cosiddetti CAMPI del database e sono denominate con un titolo proprio , univoco all'interno della tabella.
Ogni campo ha delle proprieta' come il tipo di dati che puo' ricevere, la loro lunghezza, se puo' essere nullo o no e altre ancora.

Per avere le informazioni da un db bisogna effettuare una QUERY che e' semplicemente una domanda, e per farci capire dal DB si usa il linguaggio SQL (STRUCTURED QUERY LANGUAGE), questo e' un linguaggio standard per tutti i db ed e' molto simile all'inglese!
Il linguaggio SQL si divide in 2 grosse branchie: DML e DDL, Data Manipulation Language e Data Definition Language

La differenza tra i due e' semplicemente che con il DML si lavora sui dati e con il DDL si lavora invece sulla struttura del db, quindi la creazione di nuovi DB, tabelle e la modifica di queste come aggiunta di campi o cancellazione.

Burning PHP

MYSQL

Il database che utilizzeremo sara' MySQL, un database open source che si sposa perfettamente con Php, essendoci parecchie utili funzioni e soprattutto non abbisogna di moduli aggiuntivi al momento dell'installazione.

Ma perche' si usano i db?
Fondamentalmente per raccogliere una quantita' di dati piu' o meno grande e per facilitare la ricerca di quest'ultimi.

Per poter lavorare con un db , questo deve “girare” su una macchina e noi dobbiamo connetterci a quel db.
Per poter stabilire una connessione dobbiamo fornire il nostro username, password, host e nome del db.

```
$link = mysql_connect($username,$password,$host);
mysql_select_db($link, $dbname) or die(“Impossibile connettersi”);
```

La connessione viene gestita come per i file attraverso un “puntatore a connessione” ($link), che viene richiamato al momento della scelta del db.

```
mysql_close($link)
```

E' invece il comando di chiusura della connessione.
Tra i due comandi di apertura e chiusura noi possiamo interagire con il db, chiedere dei valori, inserirne altri o modificarne altri ancora , e pure cancellarli.

Burning PHP

IL LINGUAGGIO SQL

Queste 4 azioni elencate sotto mostrano le basi del DML:
select , insert, update, delete.

Come si diceva e' molto simile all'inglese. Ogni volta che si usa uno di questi comandi
si sta eseguendo una query al db.

"SELECT nome FROM utenti"
trad: seleziona tutti i nomi dalla tabella utenti.

"SELECT nome FROM utenti WHERE eta>18"
trad: seleziona tutti i nomi dalla tabella utenti che hanno eta' maggiore di 18.

"SELECT nome FROM utenti WHERE eta>18 ORDER BY cognome ASC"
trad: seleziona tutti i nomi dalla tabella utenti che hanno eta' maggiore di 18 ed ordinameli per cognome ascendente.

"SELECT * FROM utenti WHERE eta>=18 AND nome='Piero' GROUP BY cognome"
trad: seleziona tutti i dati dalla tabella utenti che hanno eta' maggiore o uguale a 18 e nome uguale a Piero e raggruppameli per cognome.

Burning PHP

IL LINGUAGGIO SQL (DML)

Sintassi:

“INSERT INTO utenti (campo1, campo2) VALUES ('valore1', 'valore2')”;
Trad: Inserisci nella tabella utenti , nei campi1 e campi2 rispettivamente i valori valore1 e valore2.

“UPDATE utenti SET campo1='valore1', 'campo2'='valore2' WHERE nome='Piero' ”;
Trad: Modifica i campi1 e campi2 con i nuovi valori1 e valore2 dove il campo nome ha valore uguale a Piero.

“DELETE FROM utenti WHERE nome='Piero'”;
Trad: Cancella dalla tabella utenti i record che hanno come valore del campo nome , Piero

APPLICAZIONI SUPPORTO

Un' applicazione molto interessante e semplice da utilizzare per gestire un DB sono i software (scritti in php) PHPMyAdmin e PGMyAdmin.
Sono semplici software con cui e' possibile fare numerose operazioni sul db semplicemente clikkando su links.

Si possono creare nuovi db, nuove tabelle all'interno di db, aggiungere campi, modificare dati, il tutto senza dover conoscere il linguaggio SQL, sono anche buoni strumenti didattici per capire come scrivere in SQL.

Altrimenti esistono numerosi altri client per MySQL che visualizzano graficamente il DB.

Burning PHP

INTERAZIONE PHP–SQL

Vediamo come si possa interagire tra php e SQL.
Per poter eseguire una query sul db, dopo essermi connesso si usa la funzione MYSQL_QUERY

```
$result = mysql_query("SELECT * FROM utenti");
```

In questo caso mettiamo il risultato della query nella variabile $result.
Si puo' fare anche cosi:

```
$query = "SELECT * FROM utenti";
 $result = mysql_query($query);
```

Per poter accedere al risultato contenuto in $result si usa un'altra funzione:
MYSQL_FETCH_ARRAY che carica ogni riga del risultato in un array, e si puo' scegliere se quest'array ha degli indici numerici o no tramite il secondo parametro della funzione:

```
$riga = mysql_fetch_array($result, MYSQL_ASSOC);
 $riga = mysql_fetch_array($result, MYSQL_NUM);
```

Nella prima istruzione verra' creato un array che avra' come indici i nomi dei campi e nel secondo invece saranno numerici.

Burning PHP

INTERAZIONE PHP–SQL

Un'altra funzione importante per capire se la nostra query e' andata a buon fine o no

e' MYSQL_ERROR, che ritorna una stringa nela caso ci sia stato un errore (sintattico, connessione o logico), usata in combinazione con MYSQL_ERRNO che da' invece un identificativo numerico dell'errore sono dei buoni mini debugger.

Non stiamo ad affrontare teoricamente tutte le funzioni.
Le vedremo e capiremo facendo esercizi.

Burning PHP

WEBOGRAPHY

http://www.w3schools.com/sql/default.asp
Buoni tutorial per cominciare

http://dev.mysql.com/doc/
La documentazione di MySQL

http://us3.php.net/manual/it/ref.mysql.php
Documentazione delle funzioni PHP per MySQL