Linguaggi di Programmazione per il Web – Parte 8

Linguaggio PHP

I FILE e database testuali

Autore

Prof. **Rio Chierego**
riochierego@libero.it

Siti Utili

http://www.riochierego.it/mobile

http://www.html.it/

http://www.mrwebmaster.it

http://www.php.net/

http://php.html.it/

Linguaggio PHP: I FILE

Come già sappiamo un file può essere considerato la struttura fisica che implementa la struttura astratta “archivio”.

I dati dunque vengono archiviati grazie ai file su un determinato **supporto di memorizzazione** digitale. Mediante un file, dunque, possono essere archiviate le informazioni più disparate.

Ad esempio, un file può essere progettato per memorizzare **un’immagine**, un messaggio scritto, un **video**, una **canzone**, un **programma** per il computer, un **insieme di istruzioni** o una vasta gamma di altri tipi di dati.
Solitamente i file sono organizzati mediante un appropriato ***file system***, il quale ha il compito di tener traccia di dove si trovano i file nonché di consentire agli utenti di potervi accedere.

Le operazioni che generalmente vengono eseguite con i file possono essere molteplici: **verificarne l'esistenza, leggere, scrivere, rinominare, copiare, verificare i permessi**.

Il linguaggio PHP ci mette a disposizione una gamma di funzioni che, se combinate opportunamente, possono gestire tali operazioni.

La funzione file_exists()

La funzione **file_exists() verifica se un file esiste**.

Essa prende come parametro di input il **percorso al file** di cui si vuole verificare l'esistenza. Restituisce un valore booleano (TRUE/FALSE).

Essa può essere impiegata per fare controlli interni nello script.

Ad esempio

```
<?php
$file = "cartella/file.php";
if (file_exists($file) == True)    condizione logica equivalente a     (!(file_exists($nomefile)) == False)
   {
   echo "il file esiste';
   }
else
   {
   echo 'il file NON esiste';
   }
?>
```

La funzione file_get_contents()

La funzione **file_get_contents()** consente di **leggere l'output di un file dentro una stringa**.

Essa prende **come unico parametro il file** (anche esterno al dominio) e **ne restituisce l'output**.

Badate che, se si tratta (ad esempio) di un file .php essa non restituirà il codice sorgente ma l'output (ad esempio l'HTML) da questo generato.

Nel caso il file non viene trovato restituisce FALSE genera un errore di tipo warning.

Ad esempio

```
<?php
echo file_get_contents("http://google.it");
?>
```

La funzione file()

La funzione **file()** esegue una **lettura rigo per rigo di un file**.

Essa accetta **come unico parametro il percorso al file** che si desidera leggere ed esegue una lettura del file rigo per rigo **restituendo un array** in cui ciascun elemento è costituito da un singolo rigo.

N.B. La funzione risulta essere molto utile in caso di piccoli database testuali.

Ad esempio

```
<?php
$var = 'prodotti.txt';
$righe = file($var);
foreach ($righe as $key => $rigo)
  {
  echo $rigo.'<br>';
  }
?>
```

La funzione filesize()

La funzione **filesize()** serve per conoscere la **dimensione di un file** (o di una risorsa) presente sul filesystem del server.

Essa prende come parametro una stringa indicante il percorso al file e restituisce il un numero indicante i byte del file.

Se il file non viene trovato restituisce FALSE e genera un errore di tipo warning.

Ad esempio

```
<?php
$var = 'cartella/file.txt';
$dimensione = filesize($var);
echo 'La dimensione del file è '. $dimensione;
?>
```

La funzione unlink()

La funzione **unlink() elimina un file**.

Essa accetta validamente come parametro una stringa indicante il percorso al file.
Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

Ad esempio

```
<?php
$file_da_cancellare = 'doc.txt';
unlink($file_da_cancellare);
?>
```

La funzione rename()

La funzione **rename() rinomina un file**. Esso accetta due parametri: il primo è una stringa indicante al percorso al file che deve essere rinominato; il secondo è una stringa indicante il percorso al nuovo file con il nuovo nome. Essa restituisce un valore boleano (TRUE/FALSE), TRUE in caso di successo, FALSE in caso di fallimento.

Se il file da rinominare (primo parametro) non esiste si genera un errore di tipo warning.

N.B. Nelle versioni meno recenti di php tale funzione presenta problemi nei sistemi Windows.

Ad esempio

```
<?php
$old_name = 'file.txt';
$new_name = 'myfile.txt';
if(rename($old_name, $new_name) === TRUE)
  {
  echo 'File rinominato';
  }
?>
```

La funzione copy()

La funzione **copy()** copia un file. Esso accetta due parametri: il primo è una stringa indicante al percorso al file che deve essere copiato; il secondo è il percorso (comprensivo del nome del file) in cui sarà destinato la copia file.

Essa restituisce un valore boleano (TRUE/FALSE), TRUE in caso di successo, FALSE in caso di fallimento.

Se nella cartella di destinazione esiste già un file con quel nome esso verrà sovrascritto. Se il file da copiare non esiste genera un errore di tipo warning.

Ad esempio

```
<?php
$file = 'directory/file.txt';
$new_file = 'percorso/myfile.txt';
if(copy($file, $new_file) === TRUE)
  {
  echo 'File copiato';
  }
?>
```

La funzione is_writable() e is_readable()

Le funzioni **is_writable()** e **is_readable() verificano rispettivamente se il file può essere letto e se può essere scritto** (dipenderà dai permessi posti alla cartella e/o al file).

Esse prendono come parametro una stringa indicante il percorso al file e daranno come risultato un valore boleano, TRUE se il file può essere letto/scritto.

Ad esempio

```
<?php
$var = 'cartella/file.txt';
If (is_writable($var))
  {
 echo 'Il file può essere scritto';
  }
If (is_readable($var))
  {
 echo 'Il file può essere letto';
  }
?>
```

La funzione fopen()

La funzione **fopen()** serve per **aprire un file**. Essa richiede due parametri: il primo una stringa indicante il percorso al file; il secondo è una stringa che ci indicherà il tipo di operazione che si desidera effettuare sul file.
Il secondo parametro potrà assumere i seguenti valori:

- **r**: Apre il file in modalità di sola lettura
- **r+:** Apre il file in modalità di lettura e scrittura
- **w**: Apre il file in modalità di sola scrittura cancellando il contenuto esistente. Se il file non esiste, PHP tenta di crearlo.
- **w+:** Apre il file in modalità di scrittura e lettura cancellando il contenuto esistente. Se il file non esiste, PHP tenta di crearlo.
- **a**: Apre il file in modalità di sola scrittura posizionando il puntatore alla fine del file per l'inserimento. Se il file non esiste, PHP tenta di crearlo.
- **a+:** Apre il file in modalità di scrittura e lettura posizionando il puntatore alla fine del file per l'inserimento. Se il file non esiste, PHP tenta di crearlo.

Se l'apertura del file riesce essa restituisce l'identificatore della risorsa (detto anche **puntatore** o **handle**), altrimenti restituisce FALSE e genera un errore di tipo warning.

La funzione fread()

La funzione **fread()** serve per **leggere il file** dopo averlo aperto.

Essa richiede due parametri: il primo è **l'identificatore** della risorsa detto anche **puntatore**, ottenibile con fopen(); il secondo è il numero di byte che si desiderano leggere.

In relazione al secondo parametro, qualora si voglia leggere l'intero file è possibile ricorrere alla funzione **filesize()** che restituisce le dimensione in byte del file.

Nel caso in cui si voglia ottenere l'output di un file, occorre impiegare altre funzioni: funzione **file_get_contents()** se si desidera ottenere il contenuto del file in una stringa, oppure **file()** se si desidera trattare in maniera isolata i singoli righi del file.

La funzione **fgets()** è analoga a **fread().**

La funzione fwrite()

La **fwrite()** serve per **scrivere una stringa in un file**.

Essa accetta tre parametri: il primo è l'identificatore della risorsa detto anche puntatore, ottenibile con fopen(); il secondo è la stringa che si desidera scrivere; il terzo è opzionale e ci indica il numero di byte che si desidera scrivere.

Essa restituisce il numero di byte scritti oppure FALSE generando un errore di tipo warning.

La funzione **fputs()** è analoga a **fwrite().**

La funzione fclose()

La funzione **fclose()** serve per **chiudere la risorsa**. Essa accetta come unico parametro l'identificatore della risorsa. Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

Facciamo un esempio in cui utilizzeremo molte delle funzioni sopra descritte:

Uso nei file nei DATABASE testuali

I **database testuali** sono dei semplici documenti di testo (**.txt**) in cui vengono salvati dei dati per poi essere letti tramite script php.

E' possibile utilizzare la gamma di funzioni illustrate finora per procedere alla scrittura e lettura dei file in cui potranno essere salvati dei dati e letti all'occorrenza.

LIMITI

Occorre fin da subito chiarire un punto: i database testuali sono utilizzabili solo nel caso in cui **la mole di dati in esso contenuta è relativamente piccola e soggetti a sporadiche modifiche**.

Infatti essi avranno una gestibilità decisamente molto più complessa e limitata rispetto a quella ottenibile con un database MySql dato che non sarà possibile sia formulare le query, sia gestire in automatico la concorrenza.

Inoltre, risulta quasi del tutto impossibile istaurare relazioni fra i dati.

Uso nei file nei DATABASE testuali

Al fine di gestire i dati contenuti in un file .txt occorrerà:

- PASSO 1: "scrivere" il file con una idonea formattazione;
- PASSO 2: eseguire operazioni su di esso ricorrendo ad alcune funzioni; fra le tante, ve ne sono alcune che assumono particolare rilievo: **fopen(), fread(), fwrite(), fclose(), file()** ed **explode()**.

Chiariamo i due concetti esaminandoli separatamente.

DATABASE testuali: PASSO 1

Un file .txt si articola in **più righi**: ognuno di essi potrà rappresentare un **set di dati**.

Ad esempio poniamo di voler costruire un database in cui salvare i dati riguardanti dei **prodotti** su un file di testo: ogni rigo potrà rappresentare un prodotto.

Tuttavia, per ogni prodotto inserito occorrerà salvare più di una informazione: poniamo, ad esempio, di voler salvare **il nome, la taglia e il prezzo**.

Affinchè più dati siano scritti su un unico rigo occorrerà strutturare una formattazione adeguata.

Vediamone un esempio:

capo|taglia|prezzo

DATABASE testuali: PASSO 1

Pertanto il file di testo che possiamo chiamare **prodotti.txt** avrà una struttura di questo genere:

camicia|S|150

felpa|XXL|220

t-shirt|M|250

In pratica i tre dati (nome, taglia e prezzo) che desideriamo salvare su un unica riga del file .txt saranno separati gli uni dagli altri da un **carattere separatore** (poco comune) a nostra scelta, nell'esempio "|".

Quindi, **un database testuale è un file .txt in cui ciascun rigo contiene uno o più dati e all'interno del rigo i singoli dati sono separati da un carattere** (poco comune) **separatore**.

Ciò sarà rilevante sia in fase di **lettura**, sia in fase di **scrittura** del file.

DATABASE testuali: PASSO 2

In fase di "lettura" è possibile far ricorso a diverse funzioni che tuttavia presentano ognuna delle specifiche particolarità.

Quella che in caso di database testuali risulta essere maggiormente utile è **file()** (oppure **fread()**)

La funzione **file()** accetta come unico parametro il percorso al file che si desidera leggere ed esegue una **lettura del file rigo per rigo restituendo un array in cui ciascun elemento è costituito da un rigo**.

Ovviamente, avendo ottenuto un **array**, questo potrà essere letto con un classico **ciclo foreach.**

Scorrendo il nostro array otterremo un singolo rigo per ogni ciclo; per isolare i singoli dati presenti all'interno di tale rigo si farà ricorso alla funzione **explode().**

La funzione **explode()** riceve due parametri obbligatori:

- il primo è il carattere separatore;
- il secondo è una stringa da trasformare in un array.

DATABASE testuali: PASSO 2

Per poter scrivere su un file di testo occorre anzitutto assicurarsi che questo abbia i permessi di scrittura: a questo scopo può essere utile la funzione **is_writable().**

Essa prende come parametro il percorso al file e restituisce un valore boleano, TRUE se il file può essere scritto.

La scrittura del file avverrà con le funzioni **fopen(), fwrite() e fclose().**

La funzione **fopen()** serve per aprire il collegamento con la risorsa (il file da scrivere). Essa prevede due parametri obbligatori: il percorso al file e una stringa che ci indicherà modalità con la quale si vorrà operare sul file (si rimanda al manuale per maggiori dettagli).

La funzione **fwrite()** esegue la scrittura sul file e prevede due parametri: la risorsa e la stringa da scrivere.

Infine, la funzione **fclose()** esegue la chiusura del file e prevede come unico parametro la risorsa.

DATABASE testuali: Esempio svolto

Dai un'occhiata al sito www.riochierego.it/mobile e prendi in considerazione l'esempio completo trovato su Internet database-testuali.zip scaricabile nella sezione "Risorse Aggiuntive" per la classe quinta che ha l'id di risorsa 19-J

Oppure fai riferimento agli esempi con id di risorsa EX-C1, EX-C2 e EX-C3 nella medesima sezione da me realizzati che effettuano le operazioni C.R.U.D. su un database implementato con un fle di testo contenente la sola tabella "Dipendente"