

# creare una semplice galleria di immagini in PHP

Inviato da Administrator

Thursday 06 April 2006

Ultimo aggiornamento Monday 29 May 2006

L'obiettivo di questo tutorial è quello di dare un'idea su come creare una galleria di immagini dall'inizio alla fine

## Creare Una Galleria Di Immagini

### Introduzione

L'obiettivo di questo tutorial è quello di dare un'idea su come creare una galleria di immagini dall'inizio alla fine, non che questa sia l'unica via, o la via migliore per creare una galleria di immagini, ma solo un approccio per poi personalizzarla a vostro piacimento. Per continuare con questo tutorial, è necessario installare sul vostro server il PHP  $\geq$  4.3 e le librerie GD 2.

Situazione: avete molte immagini che volete visualizzare in un modo molto semplice, e molte immagini arriveranno in poco tempo. Per visualizzare queste immagini volete impiegare il minor tempo possibile e quando dico minor tempo possibile intendo fare l'upload delle immagini e basta.

Il piano: non viene utilizzato nessun database, utilizzeremo il file system per immagazzinare le immagini. Costruiremo un sistema di categorie, dove ogni cartella sarà una categoria e ogni sotto-cartella sarà una sotto-categoria. Quando visualizzeremo le immagini, saremo in grado di scegliere il numero di immagini da visualizzare contemporaneamente, in questo modo visualizzeremo le immagini in più pagine, ma daremo all'utente anche la possibilità di visualizzare tutte le immagini di una categoria contemporaneamente. Inoltre le thumbnails, verranno create automaticamente utilizzando le librerie GD.

costruire un'interfaccia di amministrazione, è più appropriato farlo come estensione di un pannello di controllo già esistente, in questo modo non sarà necessario effettuare numerosi login per ogni sezione del vostro sito. Per ora pianifichiamo solo le opzioni di upload e di eliminazione delle immagini, via FTP.

La nostra galleria sarà costituita da tre script:

- config.php (contiene le informazioni per la configurazione)
- images.php (visualizza la nostra galleria)
- imgsrc.php (crea le thumbnails)

## Creare Una Galleria Di Immagini

### Iniziamo

Per cominciare, ci serve l'informazione più importante contenuta in un file di configurazione. Questa informazione deve essere modificabile, ma non dinamicamente. Per esempio, dobbiamo conoscere la directory con la quale lavoreremo:

```
<?PHP

// directory principale delle immagini

define('PATH',
    '/inetpub/wwwroot/scripts/gallery/images/');
?>
```

Qualche volta i programmi FTP inseriscono automaticamente files di log alle directory che utilizziamo. Per evitare tutto questo ed assicurarci che il nostro script lavori solo con le immagini, definiamo un array di validi tipi di files MIME; dobbiamo utilizzare la funzione `serialize()`, perchè la funzione `define()` lavora solo con valori scalari.

```
<?PHP

// tipi di files MIME validi, tutti
// gli altri verranno ignorati

define('TYPE',
    serialize(array('image/jpg',
        'image/jpeg', 'image/pjpeg')));
?>
```

Considerando che visualizzeremo la nostra galleria in una tabella HTML, dobbiamo determinare anche quante immagini vogliamo visualizzare alla volta, impostando il numero massimo di colonne e di righe che poi andranno a comporre la tabella.

```
<?PHP

// quante righe di immagini da visualizzare
// per ogni pagina

define('ROWS', 3);

// quante colonne di immagini da visualizzare
// per ogni pagina
```

```
define('COLS', 5);
```

```
?>
```

Poi, dobbiamo impostare le dimensioni delle nostre thumbnails.

```
<?PHP
```

```
// larghezza massima delle thumbnails
```

```
define('THMBWIDTH',  
      100);
```

```
// altezza massima delle thumbnails
```

```
define('THMBHEIGHT',  
      100);
```

```
?>
```

Questo è tutto, il nostro primo script (config.php) è concluso..

## Creare Una Galleria Di Immagini

### Lo Script Per La Galleria

Ecco il nostro prossimo script: images.php. Sappiamo già come strutturarla, perciò l'idea è quella di scrivere una cosa alla volta e poi organizzare le informazioni. In altre parole, l'ultima function che scriveremo, sarà la prima ad essere inserita nello script.

Fino a quando lavoriamo con il file system del database, il nostro primo compito, è quello di acquisire la directory di lavoro dal nostro file di configurazione ed inserire il suo contenuto in un array associativo. La nostra prima riga di codice, includerà il file config.php così avremo accesso al percorso della directory.

```
<?PHP
```

```
require_once('config.php');
```

```
?>
```

Ora ci serve il contenuto della directory di lavoro. Piuttosto che scrivere un pezzo di codice alla volta e spiegarlo, commenterò il codice ed utilizzerò dei nomi per le variabili molto semplici per permettervi di seguire ogni passo. Questa è la nostra funzione directory:

```
<?PHP
```

```
// lettura del contenuto della directory
di lavoro

function directory($dir)
{
    $mydir = opendir($dir);

    while(false !== ($file
= readdir($mydir)))
    {
        if($file != "."
&& $file != "..")
        {
            //
se il nostro file è una directory, dobbiamo cercare questa
directory

            if(is_dir($dir.$file))
            {
                // facendo attenzione al safe mode, quando
utilizziamo chdir

                // potrebbe
generare un avvertimento, così lo sprimiamo utilizzando
il carattere @ //
prima della funzione

                @chdir('.');

                // siccome la funzione richiama se stessa,
dobbiamo indicizzare

                // tutte le
sotto-directory con il loro nome

                $tree[$file] = directory($dir.$file.'/');

                @chdir('..');
            }

            else
            {
```

```

    $size = getimagesize($dir.$file);

    // Se i nostri files passano il controllo
    della validità del tipo di file,

    // li inseriamo
    nell'array con un numero indicativo

    if(in_array($size['mime'], unserialize(TYPE)))

        $tree[] = $file;

    }

}

}

closedir($mydir);

return $tree;

}

$tree =
directory(PATH);

?>

```

Ora abbiamo un array con l'intera struttura della directory di lavoro. Possiamo facilmente vedere la differenza tra quali elementi sono una directory e quali sono un'immagine, solo osservando l'indice. Il prossimo passo è quello di raffigurare come come determinare ciò che ci viene richiesto. Faremo ciò passando una variabile nella uri che abbiamo chiamato \$nav, il valore di \$nav cambierà dinamicamente in base a ciò che l'utente ha cliccato.

Piuttosto che scorrere tutto l'array ad ogni richiesta dell'utente, è meglio andare direttamente nel punto richiesto dall'utente. In questo caso, il valore di \$nav sarà una lista di elementi che punteranno ad una particolare locazione del nostro array. Per esempio, prendiamo in considerazione questa struttura ad albero:

```

    Array
(
    [category one]
    =>
    Array
(
    [0] =>
        img1.jpg
    [1] => img2.jpg
    [2] => img3.jpg
    )
    [category two]
    =>
    Array

```

```
(
  [sub category]
    =>
    Array
  (
    [0] =>
      img4.jpg
    [1] => img5.jpg
  )
)
```

Se vogliamo visualizzare img4.jpg, il valore di \$nav, sarà ['category two']['sub category'][0]. Se vogliamo visualizzare la galleria della prima categoria, il valore di \$nav sarà ['category one']. Così facendo possiamo prendere il valore di \$nav e richiamarlo all'interno della variabile \$tree, per avere esattamente ciò che vogliamo. Quando creeremo i link ipertestuali per la navigazione della galleria, daremo un nuovo valore a \$nav dopo ogni click e sarà una stringa come questa: images.php?nav=>category+two>sub+category>0 - In questo modo sapremo sempre ed esattamente dove ci troviamo. NOTA: Utilizzerò l'operatore "?" molte volte, così se non lo conoscete eccovi una spiegazione (altrimenti potete visitare il sito ufficiale php.net per ulteriori informazioni):

```
<?PHP
```

```
$value = (espressione) ? 'valore se vero' : 'valore se falso';
```

```
?>
```

ritorniamo al nostro progetto:

```
<?PHP
```

```
// impostiamo la navigazione
```

```
$nav
= isset($_REQUEST['nav']) ?
  urldecode($_REQUEST['nav']) :
  null;
```

```
if(isset($nav))
{
```

```
    // dobbiamo eliminare i separatori presenti
```

```
    $crunch =
      explode('>',
        trim($nav));
    foreach($crunch as
      $value)
    {
```

```

        if(!ctype_alnum(str_replace(' ',
", $value)))

            $invalid =
            true;
    }
    if($invalid)
        $nav =
            null;
}
?>

```

Fino a quando lavoreremo con i valori `$_REQUEST`, è importante essere sicuri che i dati che riceviamo, siano del tipo di dati permessi. Non è possibile garantire che un utente non provi a modificare il percorso, ma è possibile garantire che vengano accettati solo i caratteri alfanumerici. Ora che abbiamo il nostro percorso, dobbiamo inserirlo nel nostro array `$tree` e dirigere il nostro script dove andare successivamente.

```

<?PHP

function categories($navtree)

{

    // se abbiamo un percorso col quale lavorare...

    if(isset($GLOBALS['nav']))

{

    // dobbiamo associare il percorso al nostro
array

    // iniziando a contare da 1 (0 è
vuoto)

    for($i=1; $i<count($GLOBALS['crunch']);
        $i++)

        $navtree = $navtree[$GLOBALS['crunch'][$i]];

    // tra poco studieremo questa funzione

    display_body($navtree);
}
else
display_body($navtree);
}
categories($tree);
?>

```

Sappiamo che se `$nav` è impostato, allora anche `$crunch` lo è, così ci risparmiamo il problema di ripetere il codice qui richiamandolo come variabile globale. Notate come colleghiamo il

percorso al nostro array utilizzando un riferimento; l'ho realizzato con `$array.$elements` ma non lavorava come doveva.

La prossima cosa da considerare, è la funzione `display_body`, che è la più lunga di tutte le funzioni del nostro progetto. Questa funzione necessita di determinare se è stata richiesta una singola immagine, una lista di categorie oppure una galleria di thumbnails. Ma prima di addentrarci in questo, dobbiamo impostare le variabili appropriate per la nostra galleria multipagina. Questo lo realizziamo aggiungendo una variabile al nostro percorso uri chiamata `$pg`. Il valore di `$pg` sarà un numero, che indica in quale pagina ci troviamo, oppure il valore "all" che indica che l'utente vuole visualizzare tutte le immagini di una galleria contemporaneamente.

```
<?PHP

// decidiamo quale pagina di quale
galleria stiamo visualizzando

$pg = isset($_REQUEST['pg'])
    && (ctype_digit($_REQUEST['pg']) ||
    $_REQUEST['pg'] ==
    'all') ? $_REQUEST['pg'] :
    0;

$viewall
    = is_numeric($pg) ?
    null : true;

$pg
    = intval($pg) <=
    0 ? 1
    : $pg;

?>
```

Qui siamo sicuri che `$pg` avrà sempre un valore numerico allo scopo di evitare problemi matematici nel codice che segue. Se il valore di `$pg` è "all" dobbiamo solo definire la variabile `$viewall` come true e convertire `$pg` in un numero.

```
<?PHP

// numero di immagini da visualizzare
per pagina

$section = ROWS
* COLS;

// ultimo punto di fine per le immagini, dipendentemente
dalla pagina nella quale ci troviamo

$end = $section *
$pg;

// punto di inizio per le immagini, dipendentemente
dalla pagina nella quale ci troviamo
```

```

$start = $end -
$section;

// pagina precedente

$prev =
$pg - 1;

// pagina successiva

$next = $pg +
1;

?>

```

Ora possiamo scrivere i due collegamenti prev e next. La variabile passata, \$total, sarà il totale delle immagini con le quali lavoreremo.

```

<?PHP

function navigator($total)

{

    // link alla pagina precedente

    $p = '<a
href="$_SERVER["PHP_SELF"]

    .'?nav='.rawurlencode($GLOBALS['nav'])

    . '&pg='.$GLOBALS['prev'].'"><<</a>';

    // link alla pagina successiva

    $n = '<a
href="$_SERVER["PHP_SELF"]

    .'?nav='.rawurlencode($GLOBALS['nav'])

    . '&pg='.$GLOBALS['next'].'">>></a>';

    // link alla pagina vedi tutto

    $a = '<a
href="$_SERVER["PHP_SELF"]

    .'?nav='.rawurlencode($GLOBALS['nav'])

    . '&pg=all">view all</a>';

    // il navigatore

    print('<tr><td align="center" colspan="
. COLS . '">');

```

```

    if(($GLOBALS['pg'] == 1
    && $GLOBALS['end'] >=
    $total) || isset($GLOBALS['viewall']))

        print('<< : view all : >>');

    elseif($GLOBALS['pg'] == 1)

        print('<< : '.$a.' : '.$n);

    elseif($GLOBALS['end'] < $total)

        print($p.' : '.$a.' : '.$n);

    else

        print($p.' : '.$a.' : >>');

    print('</td></tr>');
}
?>

```

La funzione navigator, deve fare qualche calcolo per determinare quali links è appropriato visualizzare in base a quante pagine di immagini abbiamo. Questo lo abbiamo inserito in una funzione nel caso che volessimo avere il nostro navigatore in più punti della pagina, ad esempio sopra e sotto la galleria di immagini.

```

<?PHP

// visualizziamo i collegamenti alle
// categorie e alle immagini

function display_body($navtree)
{
    if(isset($GLOBALS['nav']))
    {
        // ci serve il percorso come una directory
        // per visualizzare le immagini

        for($i=1; $i<count($GLOBALS['crunch']); $i++)

            $uri .= 'is_numeric($GLOBALS['crunch'][$i]) ? $GLOBALS['crunch'][$i].'/ ' :
";
    }
}

```

```

// se abbiamo un array, allora la nostra
richiesta sarà o una

// categoria di links o una galleria di
immagini

if(is_array($navtree))

{

// qui combiniamo tutte le chiavi del nostro
array in una stringa

// se non contiene un valore numerico
allora abbiamo

// delle categorie da visualizzare

if(!is_numeric(implode("", array_keys($navtree))))

{

print('<tr><td>');

foreach($navtree as $key=>$value)

{

if(!is_int($key))

{

// se il valore della nostra categoria è
un array, allora

//
creeremo un link indietro allo script delle immagini e

//
assegneremo a $nav il valore appropriato

if(is_array($value))

print('<a href="'.$_SERVER['PHP_SELF']

.'?nav='.rawurlencode($GLOBALS['nav']

.'>'. $key).'>'. $key.'</a><br />');

else

// se la categoria è vuota,
il link non serve

print($key.'<br />');

}

}

print('</td></tr>');

}

else

```

```

{
    // se non stiamo lavorando con le categorie,
    allora siamo

    // lavorando con
    le immagini, perciò decidiamo quante

    // righe visualizzare
    (dipende se $viewall è true o false)

    $total = count($navtree);

    if(isset($GLOBALS['viewall']))
    {
        $check = $total
/ COLS;

        $rows = is_int($check) ? $check
: floor($check) + 1;
    }
    else
        $rows = ROWS;

    // visualizziamo il nostro navigatore
    navigator($total);

    // iniziamo a visualizzare il contenuto
    della tabella

    for($i=0; $i<$rows; $i++)
    {
        print('<tr>');

        for($n=0; $n<COLS; $n++)
        {
            $index = $GLOBALS['start']++;

            // vogliamo o il nome del file dell'immagine,
            oppure

            //
            uno spazio vuoto per riempire il resto della tabella

            $image = $index
< $total ?
$navtree[$index] : '&nbsp;';

            print('<td align="center" width="'.THMBWIDTH.'" height="'.THMBHEIGHT.'">');

            if($image != '&nbsp;')
        {

```

```
// questo link fornisce l'esatto percorso

//
dell'immagine che vogliamo visualizzare

    print('<a
href="$_SERVER['PHP_SELF']

    .'?nav='.rawurlencode($GLOBALS['nav']

    .'>'.$index.'">');

// il tag img crea la chiamata al nostro
prossimo

//
script, imgsrc.php, il quale viene richiamato

//
con l'intero percorso della directory dell'immagine

    print('');

    print('</a>');

    }

    else
// se l'immagine non esiste, visualizza
uno spazio

    print($image);

    print('</td>');

    }

    print('</tr>');

    }

// visualizza un altro navigatore

navigator($total);

}

}

elseif(file_exists(PATH.$uri.$navtree))

{

    // se $navtree non è un array, allora
dobbiamo richiamare direttamente

    // un'immagine, prima controlleremo se
```

il file esiste

```
// ora abbiamo il percorso appropriato
per visualizzare

// l'immagine

$root = explode('/', dirname(PATH.''));

print('<tr><td></td></tr>');

}

else // niente
array e niente file = richiesta non valida

print('<tr><td>invalid request</td></tr>');

}

?>
```

Se studiate bene il codice, l'intero script inizierà ad avere un senso, specialmente dove creiamo i links e assegnamo a \$nav l'appropriato valore. Come avrete sicuramente notato in questa sezione, codifichiamo sempre i dati prima di inviarli, e li decodifichiamo quando vengono richiesti. abbiamo utilizzato molte funzioni del PHP e se volete avere maggiori informazioni su di esse, visitate il sito ufficiale scrivendo questo indirizzo: [www.php.net/il\\_nome\\_della\\_funzione\\_che\\_volete\\_conoscere](http://www.php.net/il_nome_della_funzione_che_volete_conoscere)

L'ultimo script che ci serve, è `imgsrc.php`, il quale crea le nostre thumbnails in base al percorso che inviamo tramite la variabile \$src come abbiamo già visto nella funzione `display_body`. Prima di addentrarci in questo script, voglio rivedere quello che abbiamo fatto fino ad ora e mostrarvi com'è organizzato il file `images.php`.

## Creare Una Galleria Di Immagini

### Rivediamo Ed Organizziamo Il Codice

Il primo script creato è il `config.php`:

```
<?php

// directory principale delle immagini

define('PATH', '/inetpub/wwwroot/scripts/gallery/images/');
```

```
// i files MIME validi tutti gli altri
verranno ignorati

define('TYPE', serialize(array('image/jpg',
'image/jpeg', 'image/pjpeg')));

// Quante righe di immagini da visualizzare
per pagina

define('ROWS', 3);

// Quante colonne di immagini da visualizzare
per pagina

define('COLS', 5);

// larghezza massima delle thumbnail

define('THMBWIDTH',
100);

// altezza massima delle thumbnail

define('THMBHEIGHT',
100);

?>
```

Il secondo script creato, è images.php, è stato creato passo per passo. Il seguente codice è il nostro intero script; notate che ho eliminato molti commenti e che ho inserito i tag di apertura e di chiusura delle tabelle.

```
<?php
```

```
require_once('config.php');

// il navigatore

$nav = isset($_REQUEST['nav']) ? urldecode($_REQUEST['nav']) : null;

if(isset($nav))
```

```

{
    $crunch = explode('>', trim($nav));

    foreach($crunch as $value)
    {
        if(!ctype_alnum(str_replace(' ',
", $value)))
            $invalid = true;
    }

    if($invalid)
        $nav = null;
}

// decidiamo quale pagina di quale particolare
galleria stiamo visualizzando

$pg = isset($_REQUEST['pg']) &&
(ctype_digit($_REQUEST['pg']) || $_REQUEST['pg'] == 'all') ? $_REQUEST['pg'] : 0;

$viewall =
is_numeric($pg) ? null
: true;

$pg = intval($pg) <= 0
? 1 :
$pg;

// immagini per pagina

$section =
ROWS * COLS;

// ultimo punto di fine per le immagini, dipendentemente
dalla pagina nella quale ci troviamo

$end = $section *
$pg;

// punto di inizio per le immagini, dipendentemente
dalla pagina nella quale ci troviamo

$start = $end -
$section;

// pagina precedente

$prev =
$pg - 1;

// pagina successiva

$next = $pg +
1;

```

```

// inizio tabella

print('<table border="0" align="center" cellpadding="1"
cellspacing="1">');

// creiamo i link alla pagina precedente
e successiva

function navigator($total)

{

    $p = '<a
href="$_SERVER['PHP_SELF']

        .'?nav='.rawurlencode($GLOBALS['nav'])

        .'&pg='.$GLOBALS['prev'].'"><<</a>';

    $n = '<a
href="$_SERVER['PHP_SELF']

        .'?nav='.rawurlencode($GLOBALS['nav'])

        .'&pg='.$GLOBALS['next'].'">>></a>';

    $a = '<a
href="$_SERVER['PHP_SELF']

        .'?nav='.rawurlencode($GLOBALS['nav'])

        .'&pg=all">view all</a>';

// il navigatore

    print('<tr><td align="center" colspan="
. COLS . '">');

    if(($GLOBALS['pg'] == 1
&& $GLOBALS['end'] >=
$total) || isset($GLOBALS['viewall']))

        print('<< : view all : >>');

    elseif($GLOBALS['pg'] == 1)

        print('<< : '.$a.' : '.$n);

    elseif($GLOBALS['end'] < $total)

        print($p.' : '.$a.' : '.$n);

    else

        print($p.' : '.$a.' : >>');

```

```
print('</td></tr>');
}

// lettura del contenuto della directory
di lavoro

function directory($dir)
{
    $mydir = opendir($dir);

    while(false !== ($file
= readdir($mydir)))
    {
        if($file != "."
&& $file != "..")
        {
            if(is_dir($dir.$file))
            {
                chdir('.');

                $tree[$file] = directory($dir.$file.'/');

                chdir('..');
            }
            else
            {
                $size = getimagesize($dir.$file);

                if(in_array($size['mime'], unserialize(TYPE)))

                    $tree[] = $file;
            }
        }
    }

    closedir($mydir);

    return $tree;
}

$tree =
directory(PATH);
```

```

// visualizziamo i link delle categorie
e delle immagini

function display_body($navtree)

{

    if(isset($GLOBALS['nav']))

    {

        // ci servirà il percorso delle immagini
        sistemato come una directory

        for($i=1; $i<count($GLOBALS['crunch']); $i++)

            $uri .= !is_numeric($GLOBALS['crunch'][$i]) ? $GLOBALS['crunch'][$i].'/ ' :
";

    }

    if(is_array($navtree))

    {

        if(!is_numeric(implode(", array_keys($navtree))))

        {

            print('<tr><td>');

            foreach($navtree as $key=>$value)

            {

                if(!is_int($key))

                {

                    if(is_array($value))

                        print('<a href="'.$_SERVER['PHP_SELF']

                            .'?nav='.rawurlencode($GLOBALS['nav']

                            .'>'. $key).'</a><br />');

                    else

                        print($key.'<br />');

                }

            }

            print('</td></tr>');

        }

    }

    else

    {

```

```

// decidiamo quante righe visualizzare

$total = count($navtree);

if(isset($GLOBALS['viewall']))
{
    $check = $total
/ COLS;

    $rows = is_int($check) ? $check
: floor($check) + 1;
}

else
$rows = ROWS;

navigator($total);

for($i=0; $i<$rows; $i++)
{
    print('<tr>');

    for($n=0; $n<COLS; $n++)
    {
        $index = $GLOBALS['start']++;

        $image = $index
< $total ?
$navtree[$index] : '&nbsp;';

        print('<td align="center" width=".THMBWIDTH." height=".THMBHEIGHT.">');

        if($image != '&nbsp;')
        {
            print('<a href="'. $_SERVER['PHP_SELF']

                .'?nav='.rawurlencode($GLOBALS['nav']

                .'.'. $index).'>');

            print('');

            print('</a>');

        }

        else

            print($image);

```

```
        print('</td>');
    }
    print('</tr>');
}
navigator($total);
}
}
elseif(file_exists(PATH.$uri.$navtree))
{
    $root = explode('/', dirname(PATH.''));
    print('<tr><td></td></tr>');
}
else
    print('<tr><td>invalid request</td></tr>');
}

// navighiamo nelle categorie appropriate
function categories($navtree)
{
    if(isset($GLOBALS['nav']))
    {
        for($i=1; $i<count($GLOBALS['crunch']); $i++)
            $navtree = &$navtree[$GLOBALS['crunch'][$i]];

        display_body($navtree);
    }
    else
        display_body($navtree);
}
categories($tree);
```

```
// chiusura tabella

print('</table>');

?>
```

## Creare Una Galleria Di Immagini

### Thumbnails

L'ultima parte del nostro progetto, è composta dallo script `imgsrc.php`. Questo script viene richiamato con l'intero percorso di un'immagine, tramite la variabile `$src`, e restituirà in output la thumbnail di quell'immagine facendo riferimento alle dimensioni che abbiamo inserito nel file `config.php`.

```
<?php

require_once('config.php');

$src
    = isset($_REQUEST['src']) ?
      urldecode($_REQUEST['src']) :
      null;

    // utilizziamo la funzione file_exists
    // per prevenire lavori con files remoti

    if(isset($src)
        && file_exists($src))
{
    header('Content-Type:
        image/jpeg');

    list($width,
        $height, $type,
        $attr) = getimagesize($src);

    $img =
        imagecreatefromjpeg($src);

    $lowest =
        min(THMBWIDTH
            / $width,
            THMBHEIGHT / $height);

    if($lowest <
        1)
    {
        $smallwidth =
            floor($lowest*$width);
        $smallheight =
```

```

        floor($lowest*$height);
$tmp =
        imagecreatetruecolor($smallwidth,
        $smallheight);
imagecopyresized($tmp,
        $img, 0, 0, 0, 0, $smallwidth,
        $smallheight, $width,
        $height);
imagedestroy($img);
$img =
        $tmp;
}
imagejpeg($img,
        "", 100);
imagedestroy($img);
}
?>

```

## Creare Una Galleria Di Immagini

### Conclusioni

Abbiamo creato una completa e funzionale galleria di immagini. Basta solo inserire i tre files nella stessa directory e impostare il file config.php. Ci sono molte cose da notare. Questo progetto, presuppone che tutte le immagini appartengano ad una particolare categoria, perciò avere delle immagini nella stessa directory che contiene sotto-directory potrebbe non dare i risultati voluti. Nei miei test, sono state ignorate le immagini e visualizzate le sotto-categorie.

### Consigli:

- nel file imgsrc.php, è possibile trarre vantaggio dal secondo argomento della funzione imagejpeg e copiare automaticamente cos'è thumbnail nella directory di vostra scelta, poi controllate se la thumbnail esiste ed utilizzate la funzione readfile() per visualizzarla se necessario.
- se programmate di creare un pannello di controllo per questo script per effettuare l'upload delle immagini e per eliminarle, vi consiglio i utilizzare lo schema esistente piuttosto che riscrivere il codice. Per esempio, se un amministratore è collegato, basta solo modificare il codice per visualizzare una casella di controllo (check box) per l'eliminazione a fianco della categoria e dell'immagine e inserire due campi vicino alle categorie per effettuare l'upload o per crearne delle nuove. Ovviamente le directory vuote, per gli utenti non vengono visualizzate con il loro collegamento, ma possono essere visualizzate col loro collegamento solo per gli admin per permettere loro di modificarle.