PHP5

Praktyczny kurs

Rewelacyjne strony internetowe w PHP 5 — dowiedz się, jak je zrobić!

- **Instalacja i konfiguracja narzędzi,** czyli bez czego nie da się ruszyć dalej
- **Elementy języka i współpraca z systemem plików,** czyli co koniecznie trzeba opanować
- **Obiektowy PHP i współpraca z bazami danych,** czyli co może Ci się przydać

Wydanie II

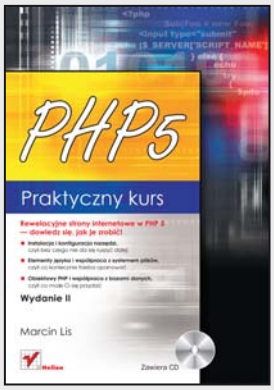
Marcin Lis

Helion

Zawiera CD

PHP 5. Praktyczny kurs. Wydanie II

Autor: Marcin Lis
ISBN: 978-83-246-3393-7
Format: 158×235, stron: 456



Rewelacyjne strony internetowe w PHP 5 – dowiedz się, jak je zrobić!

- Instalacja i konfiguracja narzędzi, czyli bez czego nie da się ruszyć dalej
- Elementy języka i współpraca z systemem plików, czyli co koniecznie trzeba opanować
- Obiektowy PHP i współpraca z bazami danych, czyli co może Ci się przydać

Znajomość języka skryptowego PHP, szczególnie w wersji PHP 5, to w dzisiejszych czasach standard, bez którego nie może się obyć żaden szanujący się twórca stron internetowych. Środowisko PHP 5 pozwala tworzyć dynamiczne witryny, efektywnie komunikujące się z bazami danych. Zapewnia też możliwość współpracy Twojej strony z różnymi rodzajami danych, a ponadto nadaje się do tworzenia samodzielnie działających aplikacji. Jeśli marzy Ci się kariera webmastera, a nie masz jeszcze odpowiednich umiejętności w tej dziedzinie lub chcesz odświeżyć wiedzę, ta książka umożliwi Ci szybkie wejście na grunt praktycznego zastosowania możliwości języka PHP w Twoich własnych projektach.

W podręczniku „PHP 5. Praktyczny kurs. Wydanie II" znajdziesz kompletne informacje o podstawach PHP – od kwestii związanych z nazewnictwem, instalacją i konfiguracją niezbędnych narzędzi, przez omówienie zasad budowy skryptów, aż po programowanie obiektowe i obsługę sieci. Dowiesz się, jak uruchomić działającą (i atrakcyjną) witrynę internetową, jak wykorzystać możliwości grafiki i o czym należy pamiętać, aby uniknąć kłopotów z wyświetlaniem witryny w przeglądarce. Nauczysz się obsługiwać protokoły sieciowe i zrozumiesz, na czym polega obsługa sesji. Sprawdź, jak wiele możesz osiągnąć, programując w PHP!

- Instalacja i konfiguracja narzędzi
- Pierwszy skrypt
- Zmienne, typy danych i operatory
- Instrukcje sterujące i funkcje
- Obsługa daty i czasu, ciągi znaków, tablice
- Operacje na strukturze systemu plików
- Operacje na plikach i praktyczne wykorzystanie plików
- Odbieranie danych z przeglądarki
- Wysyłanie danych do przeglądarki
- Obsługa cookies, sesje i wyjątki
- Programowanie obiektowe
- Obsługa grafiki
- Połączenia sieciowe, poczta i FTP
- PHP i popularne bazy danych
- Obiektowa współpraca z MySQL i SQLite
- Bazy danych w praktyce

PHP 5 – po prostu musisz to znać!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 6.

Grafika i obrazy

Lekcja 20. Obsługa grafiki

Biblioteka graficzna

PHP umożliwia zarówno wykonywanie różnorodnych operacji na obrazach, jak i generowanie plików graficznych. Wraz z PHP jest w tym celu dostarczana standardowo biblioteka GD[1] (dostępna również pod adresem *http://www.libgd.org/* lub *http://www.boutell.com/gd/*). W przypadku systemu Windows uaktywnienie funkcji graficznych uzyskamy przez usunięcie komentarza (początkowego średnika) z wiersza:

```
extension=php_gd2.dll
```

znajdującego się w pliku *php.ini* (oczywiście o ile ten wiersz jest ujęty w komentarz — poprzedzony średnikiem).

W przypadku Linuksa w pakietach dla poszczególnych dystrybucji biblioteka GD jest zwykle dołączana standardowo i nie trzeba wykonywać dodatkowych czynności. Aby włączyć obsługę biblioteki GD w przypadku samodzielnej kompilacji pakietów PHP, należy podczas konfiguracji podać dodatkową opcję:

```
--with-gd
```

jeśli chcemy korzystać z biblioteki dostarczonej wraz z PHP lub:

```
--with-gd=nazwa_katalogu
```

jeśli chcemy korzystać z zewnętrznej wersji biblioteki, np. pobranej spod podanego wyżej adresu. W tym przypadku parametr *nazwa_katalogu* powinien wskazywać katalog, w którym ta biblioteka została zainstalowana.

[1] Jako rozszerzenia dostępne są też inne biblioteki, np. *ImageMagic*, *Gmagick*, *Cairo*. Dokładne informacje o nich można znaleźć w dokumentacji PHP.

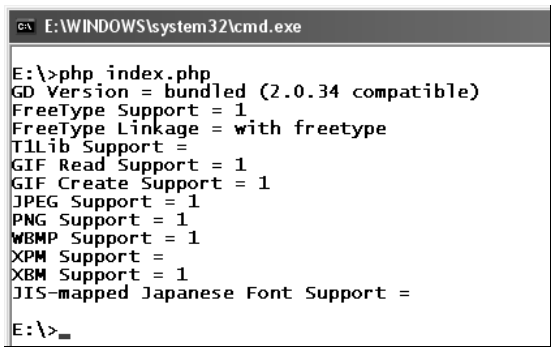
W celu przekonania się, czy dana wersja PHP ma włączoną obsługę biblioteki GD, można użyć funkcji o nazwie `gd_info`. Zwraca ona tablicę asocjacyjną zawierającą informacje o konfiguracji. Wystarczy zatem wywołać skrypt przedstawiony na listingu 6.1.

Listing 6.1. *Informacje o zainstalowanej bibliotece graficznej*

```
<?php
$arr = gd_info();
foreach($arr as $key => $val){
  echo "$key = $val\n";
}
?>
```

Przykładowy efekt działania kodu w konsoli systemu Windows został zaprezentowany na rysunku 6.1 (skrypt można też wywołać w przeglądarce, dodając jednak dla formatowania znacznik `<br />`). W sytuacji gdy wykonanie skryptu nie spowoduje wyświetlania informacji lub też pojawi się komunikat o nieznanej funkcji `gd_info`, będzie to oznaczać, że obsługa biblioteki GD nie została włączona.

Rysunek 6.1. *Informacje o wersji i opcjach biblioteki graficznej GD*



Jak stworzyć galerię obrazów?

Zanim przejdziemy do procedur przetwarzających grafikę, spróbujmy zająć się bardzo popularnym tematem tworzenia galerii obrazów. W większości wypadków nie wymaga to wykorzystywania funkcji udostępnianych przez bibliotekę GD — wystarczą zdobyte do tej pory wiadomości dotyczące języka PHP, w tym współpracy z przeglądarką (lekcje z rozdziału 4.) oraz systemem plików (lekcje z rozdziału 3.). Stwórzmy więc kilka przykładowych galerii.

Prosta galeria

Najprostsza galeria wyświetla pojedyncze obrazy wraz z odnośnikami nawigacyjnymi, tak jak zostało to zaprezentowane na rysunku 6.2. Jej utworzenie nie powinno sprawić żadnego kłopotu. Pliki z obrazami zapiszemy w osobnym katalogu, np. o nazwie *images*. Skrypt oczywiście uwzględni automatycznie wszelkie zmiany zawartości tego katalogu, nie trzeba więc będzie troszczyć się o ręczną aktualizację danych. Aby jednak nie komplikować kodu skryptu, przyjęte zostanie założenie, że katalog nie może

Rysunek 6.2. *Przykładowy wygląd galerii*

zawierać innych plików niż pliki galerii ani podkatalogów. Będzie się w nim też musiał znajdować co najmniej jeden plik graficzny.

Numer wyświetlanego obrazu będzie przekazywany do skryptu za pomocą metody `GET` w postaci parametru o nazwie `imgid`. Obrazy będą wyświetlane w kolejności alfabetycznej (według nazw plików), pierwszy z nich będzie miał identyfikator `0`, drugi — `1` itd. Na każdej stronie galerii, tak jak jest to widoczne na rysunku 6.2, będą wyświetlane:

- obraz (umieszczany na stronie za pomocą znacznika `<img>`),
- nazwa pliku graficznego,
- numer obrazu (numeracja będzie się rozpoczynać od 1),
- całkowita liczba obrazów,
- odnośniki do pierwszego, poprzedniego, następnego i ostatniego obrazu.

Skrypt generujący tak określoną galerię został przedstawiony na listingu 6.2. Przyjrzyjmy się mu bliżej.

Listing 6.2. *Prosta galeria obrazów*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Galeria obrazów</title>
```

```
</head>
<body>
<?php
$imgDir = "./images";

//odczytanie parametru
if(isSet($_GET['imgid'])){
  $imgId = $_GET['imgid'];
}
else{
  $imgId = 0;
}

//odczytanie zawartości katalogu
$dir = scandir($imgDir);
array_shift($dir);
array_shift($dir);

$count = count($dir);

//sprawdzenie poprawności parametru
if($imgId < 0 || $imgId >= $count || !is_Numeric($imgId)){
  $imgId = 0;
}

//ustalenie nazwy bieżącego obrazu oraz
//identyfikatorów obrazów dla odnośników
$imgName = $dir["$imgId"];
$first = 0;
$last = $count - 1;
if($imgId < $count - 1){
  $next = $imgId + 1;
}
else{
  $next = $count - 1;
}

if($imgId > 0){
  $prev = $imgId - 1;
}
else{
  $prev = 0;
}
?>
<div>
<div id='obraz' style='text-align:center'>
<?php
  echo "<img src=\"$imgDir/$imgName\" alt=\"$imgName\" />";
?>
</div>
<div id='opis' style='text-align:center'>
<?php
  $imgId++;
  echo "Obraz $imgName ($imgId z $count)";
?>
</div>
<div id='nawigacja' style='text-align:center'>
```

```
<?php
  echo "<a href=\"galeria.php?imgid=$first\">Pierwszy</a> ";
  echo "<a href=\"galeria.php?imgid=$prev\">Poprzedni</a> ";
  echo "<a href=\"galeria.php?imgid=$next\">Następny</a> ";
  echo "<a href=\"galeria.php?imgid=$last\">Ostatni</a> ";
?>
</div>
</div>
</body>
</html>
```

Kod rozpoczyna się od ustawienia zmiennej `$imgDir` wskazującej katalog, w którym będą umieszczane obrazy (musi to być katalog dostępny przez serwer WWW). W tym przypadku jest to podkatalog `images` katalogu, w którym będzie umieszczony skrypt *galeria.php*. Następnie za pomocą instrukcji warunkowej `if` sprawdzane jest, czy do skryptu został przekazany parametr `imgid`, czyli czy w tablicy `$_GET` jest klucz o nazwie `imgid`. Jeśli tak, jego wartość jest odczytywana, konwertowana za pomocą funkcji `intval` do typu `integer` (por. lekcja 4. z rozdziału 2.) i przypisywana zmiennej `$imgId`, w przeciwnym razie wartość zmiennej `$imgId` jest ustawiana na `0`.

Po wykonaniu tych wstępnych czynności za pomocą funkcji `scandir` odczytywana jest zawartość katalogu wskazywanego przez `$imgDir` i przypisywana zmiennej `$dir`. Jak wiadomo z lekcji 10. (rozdział 3.), wynikiem działania `scandir` jest tablica zawierająca wszystkie elementy wskazanego katalogu (dlatego też powinny się w nim znajdować wyłącznie pliki graficzne galerii) posortowane w kolejności alfabetycznej. Pierwsze dwa elementy tej tablicy to wskazanie do katalogu bieżącego (`.`) oraz nadrzędnego (`..`)[2], dlatego też są one usuwane za pomocą dwóch wywołań funkcji `array_shift` (funkcja `array_shift` usuwa z tablicy pierwszą komórkę, czyli pierwszy indeks).

Następnie za pomocą wywołania funkcji `count` pobierana jest i zapisywana w zmiennej `$count` całkowita liczba obrazów, a za pomocą instrukcji `if` następuje sprawdzenie poprawności parametru zapisanego w zmiennej `$imgId`. Nie można bowiem zakładać, że do skryptu została przekazana na pewno poprawna wartość. Dlatego też jest sprawdzane, czy wartość `$imgId` jest:

- mniejsza od 0,
- większa od maksymalnej możliwej wartości (`$count - 1`),

Jeśli jeden z tych warunków jest prawdziwy (czyli kiedy zostanie stwierdzona nieprawidłowość przekazanych danych), zmiennej `$imgId` jest przypisywana wartość `0`, co oznacza, że zostanie wyświetlony pierwszy obraz.

Po ostatecznym ustaleniu identyfikatora wyświetlanego obrazu są ustawiane wartości zmiennych, które w dalszej części posłużą jako parametry dla znaczników HTML. Są to:

- `$imgName` — nazwa pliku graficznego pobrana z tablicy `$dir`,
- `$first` — identyfikator pierwszego obrazu (czyli `0`),

[2] Wyjątkiem jest sytuacja, gdy parametrem przekazanym funkcji `scandir` jest nazwa katalogu głównego.

- `$last` — identyfikator ostatniego obrazu, czyli liczba elementów tablicy `$dir` pomniejszona o 1 (`$dir - 1`),
- `$next` — identyfikator następnego obrazu, czyli wartość aktualnego identyfikatora powiększona o 1 (`$imgId + 1`),
- `$prev` — identyfikator poprzedniego obrazu, czyli wartość aktualnego identyfikatora pomniejszona o 1 (`$imgId - 1`).

Należy przy tym zwrócić uwagę, że jeśli aktualny identyfikator ma wartość `0`, to wartością `$prev` powinno być 0 (nie ma bowiem wcześniejszego obrazu) — i analogicznie jeśli aktualny identyfikator ma wartość równą identyfikatorowi ostatniego obrazu, to wartością `$next` powinien być właśnie on (nie ma bowiem następnego obrazu).

Wszystkie zebrane w opisany sposób dane pozwalają na wygenerowanie znacznika `<img>` w postaci:

```
<img src=\"$imgDir/$imgName\" alt=\"$imgName\" />
```

np.:

```
<img src="./images/obraz001.jpg" alt="obraz001.jpg" />
```

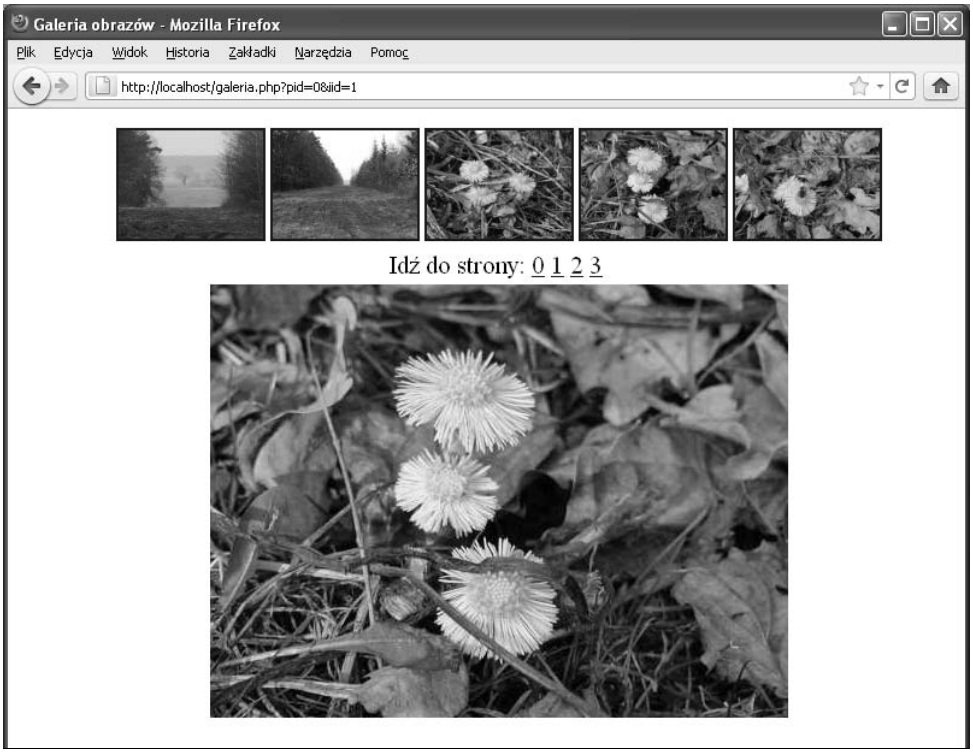
oraz odnośników:

- `<a href="galeria.php?imgid=$first">Pierwszy</a>` — do obrazu pierwszego,
- `<a href="galeria.php?imgid=$prev">Poprzedni</a>` — do obrazu poprzedniego,
- `<a href="galeria.php?imgid=$next">Następny</a>` — do obrazu następnego,
- `<a href="galeria.php?imgid=$last">Ostatni</a>` — do obrazu ostatniego,
- oraz pozostałych danych: nazwy pliku graficznego (`$imgName`), numeru aktualnego obrazu (`$imgId + 1`), całkowitej liczby obrazów (`$count`).

Galeria z miniaturami obrazów

Nieco bardziej złożonym przykładem jest galeria zawierająca miniatury obrazów (taka jak zaprezentowana na rysunku 6.3). U góry strony wyświetlana jest pewna liczba miniatur (zdefiniowana w kodzie skryptu). Kliknięcie każdej z nich powoduje wyświetlenie w dolnej części strony wybranego obrazu w pełnej rozdzielczości. Pod miniaturkami znajdują się z kolei odnośniki pozwalające na nawigację pomiędzy kolejnymi stronami miniatur. Spróbujmy napisać skrypt tworzący taką galerię.

Zacznijmy od przygotowania obrazów. Pliki graficzne zawierające zdjęcia w pełnej rozdzielczości zapiszemy w podkatalogu o nazwie *images*, natomiast miniatury w podkatalogu *thumbnails*. Miniatury można przygotować w dowolnym programie graficznym. Oba podkatalogi zapiszemy w katalogu, w którym będzie znajdował się skrypt realizujący funkcje galerii. Skrypt nazwiemy *galeria.php*. Oczywiście wszystkie wymienione nazwy można zmieniać według własnych potrzeb, będzie to jednak wymagało uwzględnienia wprowadzonych zmian w kodzie. Należy zwrócić przy tym uwagę,



Rysunek 6.3. *Galeria zawierająca miniatury obrazów*

aby w podkatalogach przeznaczonych do umieszczania plików galerii znajdowały się jedynie pliki graficzne, ponieważ skrypt nie będzie sprawdzał typów plików (weryfikacja zostanie pominięta w celu uproszczenia kodu).

Zastanówmy się teraz, jak ma wyglądać struktura strony zawierającej galerię. Może być ona zrealizowana na kilka sposobów. Najprościej jest użyć kilku warstw (które można dowolnie ostylować) albo też do formatowania użyć zwykłej tabeli (choć nie jest to obecnie sposób zalecany). Skorzystajmy zatem z trzech warstw. Górna będzie przechowywać obrazy miniatur, środkowa — odnośniki do kolejnych stron z miniaturami, a dolna — wybrany przez użytkownika obraz w pełnej rozdzielczości. Struktura ta będzie miała zatem schematyczną postać:

```
<div id='miniatury' style='text-align:center'>
  <!-- lista miniatur -->
</div>
<div id='nawigacja' style='text-align:center'>
  <!-- odnośniki do kolejnych stron z miniaturami -->
</div>
<div id='obraz' style='text-align:center'>
  <!-- obraz w pełnej rozdzielczości-->
</div>
```

Do skryptu będą przekazywane dwa argumenty:

- `pid` — (od *page id*) określający numer strony z miniaturkami, która ma zostać wyświetlona w górnej części witryny.
- `iid` — (od *image id*) określający numer obrazu, który ma zostać wyświetlony w dolnej części witryny.

Miniatury powinny mieć takie same nazwy jak właściwe pliki z grafiką i być według nich ustawione w kolejności alfabetycznej. Ponieważ liczba obrazów w galerii nie będzie z góry ustalana (tak aby można było ją aktualizować jedynie przez zmianę zawartości katalogów z grafiką), niezbędne będzie obliczanie liczby stron z miniaturami, co będzie się odbywało według prostego wzoru:

```
liczba stron = całkowita liczba miniatur / liczba miniatur na pojedynczej stronie
```

Kod skryptu generującego galerię został przedstawiony na listingu 6.3. W celu zaoszczędzenia miejsca zostały w nim pominięte wstępne znaczniki HTML, które są takie same jak w poprzednim przykładzie.

Listing 6.3. *Skrypt generujący galerię z miniaturami obrazów*

```
<!-- początek kodu HTML jak w przykładzie z listingu 6.2 -->
<body>
<?php
$imgDir = "./images";
$thumbDir = "./thumbnails";
$thOnPage = 5;

if(isSet($_GET['iid']) && isSet($_GET['pid'])){
  $iId = $_GET['iid'];
  $pId = $_GET['pid'];
}
else{
  $iId = 0;
  $pId = 0;
}

$dir = scandir($thumbDir);
array_shift($dir);
array_shift($dir);

$count = count($dir);
$pages = ceil($count / $thOnPage);

if($iId < 0 || $iId > $count || $pId < 0 || $pId > $pages){
  $iId = 0;
  $pId = 0;
}
?>

<div id='miniatury' style='text-align:center'>

<?php
for($i = 0; $i < $thOnPage; $i++){
  $imgNo = $pId * $thOnPage + $i;
  if($imgNo >= $count) break;
  $imgName = $dir[$imgNo];
```

```
    $imgTag = "<img src=\"$thumbDir/$imgName\" alt=\"$imgName\" />\n";
    $aHead = "<a href=\"./galeria.php?pid=$pId&amp;iid=$imgNo\">";
    $aFoot = "</a>";
    echo "$aHead $imgTag $aFoot\n";
  }
  ?>
  </div>
  <div id='nawigacja' style='text-align:center'>
  Idź do strony:
  <?php
  for($i = 0; $i < $pages; $i++){
    echo "<a href=\"./galeria.php?pid=$i&amp;iid=$iId\">$i</a> ";
  }
  ?>
  </div>
  <div id='obraz' style='text-align:center'>
  <?php
    $imgName = $dir[$iId];
    echo "<img src=\"$imgDir/$imgName\" alt=\"$imgName\" />\n";
  ?>
  </div>
  </body>
  </html>
```

Kod rozpoczyna się od ustawienia zmiennych, które zawierają: nazwę katalogu z obrazami (`$imgDir`), nazwę katalogu z miniaturami (`$thumbDir`) oraz liczbę miniatur wyświetlanych na pojedynczej stronie (`$thOnPage`). Dzięki temu wszelkie modyfikacje tych parametrów będą mogły być wykonywane w prosty sposób w jednym miejscu. Następnie ma miejsce sprawdzenie, czy do skryptu zostały przekazane parametry `pid` i `iid`. Jeśli tak, ich wartości za pomocą funkcji `intval` są konwertowane na liczby całkowite i przypisywane zmiennym `$pid` i `$iid`, jeśli nie — zmienne te otrzymują wartość `0`. Oznacza to, że wywołanie skryptu bez podania parametrów spowoduje, iż zostaną wyświetlone strona miniatur oraz obraz, które będą miały indeks 0.

Kolejne instrukcje to wczytanie i zapisanie w zmiennej `$dir` zawartości katalogu z miniaturami oraz usunięcie dwóch pierwszych komórek tej tablicy, czyli pozbycie się wskazań do katalogu bieżącego i nadrzędnego. Te czynności są wykonywane dokładnie w ten sam sposób jak w poprzednim przykładzie. Po wczytaniu danych następuje obliczenie całkowitej liczby plików (zmienna `$count`) oraz wynikającej z niej liczby stron z miniaturami (zmienna `$pages`). Ponieważ w tym drugim przypadku wynik może być liczbą z częścią ułamkową, jest on zaokrąglany do liczby całkowitej za pomocą funkcji `ceil` (zaokrąglenie w górę).

Gdy znamy już zarówno całkowitą liczbę plików, jak i liczbę stron z miniaturami, a zatem również zakres wartości, jakie mogą przyjmować zmienne `$iId` i `$pId`, dokonujemy weryfikacji danych. Nie można bowiem zakładać, że dane przesłane do skryptu na pewno są poprawne. W przypadku stwierdzenia jakiejkolwiek nieprawidłowości (wartości poza dopuszczalnym zakresem) zmiennym `$iId` i `$pId` zostaną przypisane wartości 0, czyli skrypt zachowa się tak, jakby nie zostały mu przekazane żadne parametry.

Po przeprowadzeniu wszystkich opisanych procedur można już przystąpić do wygenerowania górnego wiersza zawierającego obrazy miniatur. Proces ten odbywa się w pętli

for, która ma tyle przebiegów, ile miniatur ma się znaleźć na stronie. Wewnątrz pętli najpierw jest obliczany indeks aktualnie przetwarzanej miniatury (zmienna $imgNo), który wynika ze wzoru:

```
indeks obrazu = indeks wyświetlanej strony * liczba miniatur na stronie + $i
```

Gdyby się okazało, że tak wyliczona wartość jest większa od całkowitej liczby obrazów lub jej równa, pętla zostanie przerwana. Oznacza to bowiem, że przetwarzana jest ostatnia strona z miniaturami, na której liczba miniatur może być mniejsza niż na pozostałych stronach.

Po obliczeniu indeksu aktualnej miniatury z tablicy $dir jest odczytywana (i zapisywana w zmiennej $imgName) nazwa odpowiadającego jej pliku. Pozwala to na skonstruowanie znacznika <img> w postaci:

```
<img src="$thumbDir/$imgName" alt="$imgName" />
```

który jest przypisywany zmiennej $imgTag. Konstruowany jest również znacznik <a>, który spowoduje, że miniatura będzie jednocześnie odnośnikiem pozwalającym na wyświetlenie odpowiadającego jej obrazu w pełnej rozdzielczości. Znacznik ten przyjmuje postać:

```
<a href="./galeria.php?pid=$pId&amp;iid=$imgNo"> znacznik <img> </a>
```

W kodzie został on rozbity na dwie części — $aHead i $aFoot — tak aby w prosty sposób można było skonstruować zawartość wysyłaną do przeglądarki za pomocą instrukcji echo. Przyjmuje ona postać:

```
echo "$aHead $imgTag $aFoot";
```

Należy zwrócić uwagę na sposób zapisania znaku & w postaci &. Jest to niezbędne, aby zachować zgodność z obowiązującymi standardami HTML.

Druga część strony (druga warstwa) musi zawierać odnośniki do wszystkich stron z miniaturami. Jej konstrukcja nie jest skomplikowana. Odnośniki są generowane w pętli for, a każdy z nich ma postać:

```
<a href="./galeria.php?pid=$i&amp;iid=$iId">$i</a> 
```

Jak widać, zmienia się tu wartość parametru pid, natomiast wartością iid jest indeks aktualnie wyświetlanego obrazu. Dzięki temu obraz w pełnej rozdzielczości wyświetlany w dolnej komórce tabeli będzie się zmieniał tylko na wyraźne życzenie użytkownika przeglądającego galerię, gdy kliknie on wybraną miniaturę.

Równie prosta jest konstrukcja ostatniego, dolnego wiersza witryny (trzeciej warstwy). Powinien się w nim znaleźć jedynie znacznik <img> powodujący wyświetlenie wybranego przez użytkownika obrazu. Nazwa pliku z tym obrazem jest pobierana z tablicy $dir (jej indeks jest zapisany w zmiennej $iId) i przypisywana pomocniczej zmiennej $imgName. Następnie konstruowany jest znacznik <img> w postaci:

```
<img src="$imgDir/$imgName" alt="$imgName" />
```

który jest wysyłany do przeglądarki za pomocą instrukcji echo.

Przetwarzanie grafiki

Tworzenie obrazu

W celu utworzenia nowego obrazu w pamięci należy wykorzystać funkcję `imagecreatetruecolor`[3], której jako argumenty przekazuje się szerokość i wysokość obrazu. Schematyczne wywołanie ma zatem postać:

```
imagecreatetruecolor(szerokość, wysokość)
```

Funkcja zwraca identyfikator obrazu (wartość typu `resource`), który pozwala na dalsze operacje. Gdy obraz utworzony za pomocą `imagecreatetruecolor` nie będzie już potrzebny, powinien zostać usunięty z pamięci za pomocą wywołania `imagedestroy`. Funkcja `imagedestroy` zwraca wartość `true`, jeśli operacja się powiedzie, lub `false` w przeciwnym wypadku. Konstrukcja skryptu (dla przykładowego obrazu o rozmiarach 200×100 pikseli) powinna być więc następująca:

```
<?php
$wysokosc = 100;
$szerokosc = 200;
$img = imagecreatetruecolor($szerokosc, $wysokosc);
//operacje na obrazie
imagedestroy($img);
?>
```

W przypadku gdy chcemy wczytać obraz z pliku graficznego, należy użyć funkcji dedykowanej dla danego formatu graficznego[4]:

- `imagecreatefromgif` — dla formatu GIF,
- `imagecreatefromjpeg` — dla formatu JPEG,
- `imagecreatefrompng` — dla formatu PNG.

Każda z nich przyjmuje jako argument nazwę pliku graficznego (o ile konfiguracja środowiska na to pozwala, może być to też ciąg wskazujący URL obrazu). Wczytanie zawartości przykładowego pliku o nazwie *obraz1.jpg*, który znajduje się w katalogu */var/www/images*, osiągniemy przez wykonanie instrukcji:

```
$img = imagecreatefromjpeg("/var/www/images/obraz1.jpg");
```

a gdy ten obraz znajduje się w katalogu *c:\www\obrazy*, dzięki instrukcji:

```
$img = imagecreatefromjpeg("c:\\www\\obrazy\\obraz1.jpg");
```

lub:

```
$img = imagecreatefromjpeg("c:/www/obrazy/obraz1.jpg");
```

[3] Można także użyć funkcji `imagecreate`, zaleca się jednak korzystanie z `imagecreatetruecolor`.

[4] Dostępne są także funkcje dla mniej popularnych formatów, np. gd2, bmp, xpm itp. Ich opis można znaleźć w dokumentacji PHP.

Zapisywanie plików graficznych

Obraz utworzony w pamięci (niezależnie od tego, jak ta czynność została wykonana) może być w prosty sposób zapisany do pliku lub wysłany do standardowego wyjścia (np. przeglądarki). W zależności od wybranego formatu należy użyć jednej z dedykowanych funkcji:

- `imagegif` — dla formatu GIF,
- `imagejpeg` — dla formatu JPEG,
- `imagepng` — dla formatu PNG.

Każda z nich przyjmuje dwa argumenty — pierwszy określa obraz, natomiast drugi nazwę pliku. Jeśli parametr określający nazwę pliku zostanie pominięty, obraz jest wysyłany do standardowego wyjścia. Aby na przykład zapisać na dysku plik typu JPEG o nazwie *image.jpg* utworzony z obrazu wskazywanego przez zmienną `$img`, zastosujemy konstrukcję:

```
imagejpeg($img, "image.jpg");
```

aby natomiast wysłać taki obraz do przeglądarki — konstrukcję[5]:

```
imagejpeg($img);
```

Funkcje `imagejpg` oraz `imagepng` mogą też przyjmować trzeci argument określający jakość pliku wynikowego. W przypadku `imagejpg` jest to wartość od 0 do 100, przy czym 0 oznacza największą kompresję i największą utratę jakości (ale najmniejszy rozmiar pliku wynikowego), natomiast 100 — najmniejszą kompresję i najmniejszą utratę jakości (ale największy rozmiar pliku wynikowego). W przypadku `imagepng` jest to wartość od 0 do 9, gdzie 0 oznacza brak kompresji, a 9 największą kompresję[6].

Jeżeli stosowany jest trzeci argument, a obraz powinien być wysłany do przeglądarki, jako nazwy pliku (drugi argument) należy użyć wartości `null`, np.:

```
imagejpeg($img, null, 85);
```

Kolory

Wiele funkcji operujących na obrazie wymaga podania w postaci argumentu koloru, który ma być użyty (np. funkcja rysująca linie wymaga podania koloru linii). Aby jednak móc skorzystać z danego koloru, niezbędne jest wcześniejsze jego zaalokowanie, które odbywa się przez wywołanie funkcji `imagecolorallocate`. Jej wywołanie ma postać:

```
imagecolorallocate($obraz, czerwony, zielony, niebieski)
```

gdzie *$obraz* to identyfikator zwrócony przez jedną z funkcji tworzących obraz, natomiast pozostałe argumenty określają poszczególne składowe koloru w formacie RGB. Mogą być one podawane w postaci dziesiętnej lub szesnastkowej. Wybrane kolory

[5] W takim przypadku niezbędne byłoby również wcześniejsze wysłanie odpowiedniego nagłówka HTTP.

[6] Ten argument jest dostępny od PHP w wersji 5.1.2. Począwszy od wersji 5.1.3, jest też możliwe stosowanie czwartego argumentu określającego filtry, jakie mają być zastosowane do obrazu.

i odpowiadające im składowe RGB zostały przedstawione w tabeli 6.1. Funkcja `imagecolorallocate` zwraca identyfikator koloru lub wartość `false` (w wersjach PHP poniżej 5.1.3 — –1), jeśli jej działanie nie zakończyło się powodzeniem. Przykładowo, uzyskanie indeksu koloru czerwonego dla obrazu wskazywanego przez zmienną `$img` będzie wymagało wywołania w postaci:

```
$czerwony = imagecolorallocate($img, 255, 0, 0);
```

lub:

```
$czerwony = imagecolorallocate($img, 0xFF, 0, 0);
```

Tabela 6.1. *Wybrane kolory i odpowiadające im składowe RGB*

Kolor	Składowa R	Składowa G	Składowa B
Beżowy	245	245	220
Biały	255	255	255
Błękitny	0	191	255
Brązowy	139	69	19
Czarny	0	0	0
Czerwony	255	0	0
Ciemnoczerwony	192	0	0
Ciemnoniebieski	0	0	139
Ciemnoszary	169	169	169
Ciemnozielony	0	100	0
Fiolet	238	130	238
Koralowy	255	127	80
Niebieski	0	0	255
Oliwkowy	128	128	0
Purpurowy	128	0	128
Srebrny	192	192	192
Stalowoniebieski	70	130	180
Szary	128	128	128
Zielony	0	255	0
Żółtozielony	192	255	62
Żółty	255	255	0

Jeśli chcemy wypełnić wybranym kolorem pewien obszar obrazu, można skorzystać z funkcji `imagefill`, której wywołanie ma postać:

```
imagefill($obraz, wspx, wspy, kolor);
```

gdzie `$obraz` określa obraz, *wspx* i *wspy* — współrzędną punktu, od którego rozpocznie się procedura wypełniania, a *kolor* — kolor wypełnienia. Aby na przykład w obrazie wskazywanym przez zmienną `$img` wypełnić kolorem niebieskim obszar rozpoczynający się od punktu o współrzędnych (15, 25), należy zastosować instrukcję:

```
$niebieski = imagecolorallocate($img, 0, 0, 0xFF);
imagefill($img, 15, 25, $niebieski);
```

Informacje o obrazie

Jeśli chcemy otrzymać informację o rozmiarach znajdującego się w pamięci obrazu, możemy skorzystać z funkcji `imagesx` (z ang. *image size x*) i `imagesy` (z ang. *image size y*). Pierwsza z nich zwraca szerokość, a druga wysokość obrazu. W obu przypadkach jako argument należy podać odnośnik do obrazu. Jeśli zatem utworzyliśmy lub wczytaliśmy z pliku obraz, który jest identyfikowany przez zmienną `$img`, to jego rozmiary możemy uzyskać za pomocą przykładowych wywołań:

```
$szerokosc = imagesx($img);
$wysokosc = imagesy($img);
```

Równie przydatną funkcją jest `getimagesize`, która (nieco wbrew swojej nazwie) zwraca wiele przydatnych informacji, a nie tylko rozmiary obrazu. Jej wywołanie ma postać:

```
getimagesize(nazwa_pliku[, $tablica]);
```

gdzie *nazwa_pliku* określa nazwę pliku, z którego dane chcemy odczytać (może to być zarówno plik lokalny, jak i zdalny), a *tablica* — opcjonalną tablicę, w jakiej zostaną zapisane dodatkowe informacje (w obecnej wersji PHP zapisywane są w niej jedynie niektóre znaczniki APP z plików JPG).

Wartością zwracaną przez `getimagesize` jest tablica o następującej zawartości:

- indeks `0` — szerokość obrazu w pikselach,
- indeks `1` — wysokość obrazu w pikselach,
- indeks `2` — określenie typu pliku (dostępne wartości zostały zaprezentowane w tabeli 6.2),
- indeks `3` — ciąg znaków zapisany w postaci `height="`*wysokość*`" width="`*szerokość*`"`, który określa rozmiary obrazu,
- klucz `mime` — ciąg znaków określający typ *mime* pliku.
- klucz `channels` — liczba kanałów (3 dla RGB, 4 dla CMYK).
- klucz `bits` — liczba bitów, na których zapisywany jest kolor (z czego bezpośrednio wynika maksymalna możliwa liczba kolorów w obrazie).

Generowanie grafiki

Na obrazie utworzonym lub wczytanym w sposób opisany w sekcji „Tworzenie obrazu” można za pomocą odpowiednich funkcji wykonywać różne operacje graficzne. Na kolejnych stronach omówione zostanie kilka z nich. Funkcje te pozwalają m.in. na rysowanie linii, łuków, prostokątów, wielokątów i elips.

Rysowanie linii

Do rysowania linii służy funkcja `imageline`, której wywołanie ma postać:

```
imageline($obraz, xp, yp, xk, yk, kolor)
```

Tabela 6.2. *Typy plików rozpoznawane przez funkcję getimagesize*

Typ	Wartość
GIF	1
JPEG	2
PNG	3
SWF	4
PSD	5
BMP	6
TIFF_II[7]	7
TIFF_MM[8]	8
JPC	9
JPEG_2000	9
JP2	10
JPX	11
JB2	12
SWC	13
IFF	14
WBMP	15
XBM	16
ICO[9]	17

gdzie:

- `$obraz` — to określenie obrazu,
- *xp* — współrzędna x początku linii,
- *yp* — współrzędna y początku linii,
- *xk* — współrzędna x końca linii,
- *yk* — współrzędna y końca linii,
- `kolor` — określenie koloru.

Należy przy tym pamiętać, że współrzędne lewego górnego rogu to (0, 0). Przykładowo, aby utworzyć obraz o białym tle i o rozdzielczości 100×30 pikseli, który by zawierał dwie przekątne (pierwszą koloru zielonego, a drugą czerwonego), i zapisać go na dysku pod nazwą *obraz1.jpg*, należy wykonać kod widoczny na listingu 6.4.

[7] Kolejność bajtów zgodna z formatem Intela (z ang. *intel byte order*).

[8] Kolejność bajtów zgodna z formatem Motoroli (z ang. *motorola byte order*).

[9] Począwszy od PHP w wersji 5.3.0.

Listing 6.4. *Rysowanie przekątnych*

```
<?php
$img = imagecreatetruecolor(100, 30);
$bialy = imagecolorallocate($img, 255, 255, 255);
$zielony = imagecolorallocate($img, 0, 255, 0);
$czerwony = imagecolorallocate($img, 255, 0, 0);

imagefill($img, 0, 0, $bialy);

imageline($img, 0, 0, 99, 29, $czerwony);
imageline($img, 0, 29, 99, 0, $zielony);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Obraz jest tworzony za pomocą funkcji `imagecreatetruecolor`. Ta funkcja zwraca zasób będący odwołaniem do obrazu (czy też identyfikatorem obrazu). A zatem w dalszej części skryptu obraz będzie reprezentowany przez zmienną `$img`. W uproszczeniu często mówi się po prostu o obrazie `$img`.

Następnie za pomocą funkcji `imagecolorallocate` alokowane są trzy kolory: biały, zielony i czerwony. Wyniki działania funkcji (identyfikatory kolorów) są przypisywane zmiennym `$bialy`, `$zielony` i `$czerwony`. Obszar obrazu jest wypełniany kolorem białym dzięki funkcji `imagefill`, a przekątne są rysowane przez funkcję `imageline`. Na zakończenie obraz jest zapisywany do pliku o nazwie *obraz1.jpg* (odpowiada za to funkcja `imagejpg`) oraz usuwany z pamięci (odpowiada za to funkcja `imagedestroy`).

Rysowanie prostokątów

Do rysowania prostokątów służą funkcje `imagerectangle` i `imagefilledrectangle`, których wywołanie ma postać:

```
imagerectangle($obraz, xp, yp, xk, yk, kolor)
imagefilledrectangle($obraz, xp, yp, xk, yk, kolor)
```

gdzie:

- `$obraz` — to określenie obrazu,
- *xp* — współrzędna x lewego górnego rogu,
- *yp* — współrzędna y lewego górnego rogu,
- *xk* — współrzędna x prawego dolnego rogu,
- *yk* — współrzędna y prawego dolnego rogu,
- *kolor* — określenie koloru.

Pierwsza funkcja tworzy sam kontur prostokąta, a druga prostokąt wypełniony kolorem. Aby zatem utworzyć obraz o białym tle i o rozdzielczości 100×100 pikseli, który by zawierał cztery ułożone w szachownicę prostokąty w kolorach niebieskim i zielonym (w tym dwa wypełnione własnym kolorem i dwa wypełnione kolorem tła), oraz zapisać go na dysku pod nazwą *obraz1.jpg*, należy wykonać kod widoczny na listingu 6.5.

Listing 6.5. *Tworzenie prostokątów*

```
<?php
$img = imagecreatetruecolor(100, 100);
$bialy = imagecolorallocate($img, 255, 255, 255);
$zielony = imagecolorallocate($img, 0, 255, 0);
$niebieski = imagecolorallocate($img, 0, 0, 255);

imagefill($img, 0, 0, $bialy);

imagefilledrectangle($img, 10, 10, 50, 50, $niebieski);
imagefilledrectangle($img, 50, 50, 90, 90, $niebieski);
imagerectangle($img, 10, 50, 50, 90, $zielony);
imagerectangle($img, 50, 10, 90, 50, $zielony);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Rysowanie wielokątów

Do rysowania wielokątów służą funkcje `imagepolygon` i `imagefilledpolygon`, których wywołanie ma postać:

```
imagepolygon($obraz, $punkty, ile, kolor)
imagefilledpolygon($obraz, $punkty, ile, kolor)
```

gdzie:

- `$obraz` — to określenie obrazu,
- `$punkty` — tablica zawierająca współrzędne kolejnych punktów,
- `ile` — liczba wierzchołków,
- `kolor` — określenie koloru.

Tablica *punkty* musi być zbudowana w taki sposób, że klucze `0` i `1` zawierają współrzędne `x` i `y` pierwszego wierzchołka, klucze `2` i `3` — drugiego wierzchołka, `4` i `5` — trzeciego wierzchołka itd. Funkcja `imagepolygon` rysuje sam kontur wielokąta, natomiast `imagefilledpolygon` wielokąt wypełniony wskazanym kolorem. Aby na przykład uzyskać obraz o białym tle i o rozdzielczości 320×200 pikseli, który by zawierał sześciokąt wypełniony kolorem żółtym, należy wykorzystać kod widoczny na listingu 6.6.

Listing 6.6. *Tworzenie sześciokąta wypełnionego kolorem żółtym*

```
<?php
$img = imagecreatetruecolor(320, 200);
$bialy = imagecolorallocate($img, 255, 255, 255);
$zolty = imagecolorallocate($img, 255, 255, 0);

imagefill($img, 0, 0, $bialy);

$tab = array(80, 100, 120, 20, 200, 20, 240, 100, 200, 180, 120, 180);

imagefilledpolygon($img, $tab, 6, $zolty);
```

```
imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Rysowanie elips

Do rysowania elips służą funkcje `imageelipse` i `imagefilledelipse`, których wywołanie ma postać:

```
imageelipse($obraz, xc, yc, xw, xh, kolor)
imagefilledelipse($obraz, xc, yc, xw, xh, kolor)
```

gdzie:

- *$obraz* — to określenie obrazu,
- *xc* — współrzędna x środka elipsy,
- *yc* — współrzędna y środka elipsy,
- *xw* — szerokość elipsy (średnica pozioma),
- *xh* — wysokość elipsy (średnica pionowa),
- *kolor* — określenie koloru.

Funkcja `imageelipse` rysuje sam kontur, natomiast `imagefilledelipse` elipsę wypełnioną wskazanym kolorem. Aby na przykład uzyskać obraz o białym tle i o rozdzielczości 320×200 pikseli, który by zawierał czarny okrąg oraz elipsę wypełnioną kolorem niebieskim, należy wykorzystać kod widoczny na listingu 6.7.

Listing 6.7. *Rysowanie elips*

```
<?php
$img = imagecreatetruecolor(320, 200);
$bialy = imagecolorallocate($img, 255, 255, 255);
$czarny = imagecolorallocate($img, 0, 0, 0);
$niebieski = imagecolorallocate($img, 0, 0, 255);

imagefill($img, 0, 0, $bialy);

imagefilledellipse($img, 100, 100, 100, 30, $niebieski);
imageellipse($img, 220, 100, 80, 80, $czarny);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Rysowanie wycinków elips

Do rysowania wycinków elips służą funkcje `imagearc` i `imagefilledarc`, których wywołanie ma postać:

```
imagearc($obraz, xc, yc, xw, xh, k1, k2, kolor)
imagefilledarc($obraz, xc, yc, xw, xh, k1, k2, kolor, styl)
```

gdzie:

- *$obraz* — to określenie obrazu,
- *xc* — współrzędna x środka elipsy,
- *yc* — współrzędna y środka elipsy,
- *xw* — szerokość elipsy (średnica pozioma),
- *xh* — wysokość elipsy (średnica pionowa),
- *k1* — kąt (w stopniach) określający linię początkową,
- *k2* — kąt (w stopniach) określający linię końcową,
- *kolor* — określenie koloru,
- *styl* — styl wypełnienia.

Pierwsza z nich rysuje sam łuk, druga wycinek elipsy zgodnie ze stylem przekazanym w argumencie *styl*. Argument ten składa się z następujących stałych, które można łączyć przez sumę bitową (operator |):

- IMG_ARC_PIE — wycinek, którego końce są połączone łukiem, standardowo wypełniony zadanym kolorem.
- IMG_ARC_CHORD — wycinek, którego końce połączone są linią prostą, standardowo wypełniony zadanym kolorem.
- IMG_ARC_NOFILL — wycinek nie będzie wypełniany kolorem.
- IMG_ARC_EDGED — w połączeniu z IMG_ARC_NOFILL powoduje, że zostanie wykreślony pełny kontur wycinka.

Opcje IMG_ARC_PIE i IMG_ARC_CHORD wzajemnie się wykluczają, czyli nie mogą być użyte jednocześnie. Wygenerowanie obrazu widocznego na rysunku 6.4 osiągniemy zatem, stosując kod zaprezentowany na listingu 6.8.

Rysunek 6.4. *Wykorzystanie różnych opcji funkcji imagefilledarc*

Listing 6.8. *Różne sposoby rysowania wycinków elips*

```
<?php
$img = imagecreatetruecolor(400, 200);
$bialy = imagecolorallocate($img, 255, 255, 255);
$czerwony = imagecolorallocate($img, 255, 0, 0);
$niebieski = imagecolorallocate($img, 0, 0, 255);

imagefill($img, 0, 0, $bialy);

imagearc($img, 110, 100, 80, 80, 0, 90, $czerwony);
imagearc($img, 90, 100, 80, 80, 90, 180, $czerwony);
imagearc($img, 90, 90, 80, 80, 180, 270, $czerwony);
imagearc($img, 110, 90, 80, 80, 270, 360, $czerwony);
```

```
imagefilledarc($img, 310, 100, 80, 80, 0, 90, $niebieski, IMG_ARC_PIE);
imagefilledarc($img, 290, 100, 80, 80, 90, 180, $niebieski, IMG_ARC_CHORD);
imagefilledarc($img, 290, 90, 80, 80, 180, 270, $niebieski, IMG_ARC_NOFILL);
imagefilledarc($img, 310, 90, 80, 80, 270, 360, $niebieski, IMG_ARC_NOFILL |
    IMG_ARC_EDGED);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Przetwarzanie obrazów

Nakładanie filtrów

Wśród funkcji przetwarzających obrazy znajduje się `imagefilter`, która nakłada na nie jeden z dostępnych filtrów. Jej wywołanie ma postać:

```
imagefilter($obraz, filtr[, arg1[, arg2[, arg3]]])
```

Argument $*obraz* określa obraz poddawany zmianie, *filtr* — rodzaj filtra (dostępne wartości zostały podane w tabeli 6.3), natomiast *arg1*, *arg2*, *arg3* i *arg4* to parametry niezbędne do działania niektórych filtrów. Funkcja zwraca wartość `true`, jeśli jej działanie zakończyło się sukcesem, lub `false` w przeciwnym razie.

Tabela 6.3. *Filtry dostępne dla funkcji imagefilter*

Nazwa filtra	Opis
`IMG_FILTER_NEGATE`	Inwersja kolorów.
`IMG_FILTER_GRAYSCALE`	Konwersja obrazu do odcieni szarości.
`IMG_FILTER_BRIGHTNESS`	Zmiana poziomu jasności obrazu. Do jego ustalenia należy wykorzystać argument `arg1`.
`IMG_FILTER_CONTRAST`	Zmiana kontrastu obrazu. Do ustalenia kontrastu należy wykorzystać argument `arg1`.
`IMG_FILTER_COLORIZE`	Działanie podobne do `IMG_FILTER_GRAYSCALE` z tą różnicą, że istnieje możliwość ustalenia koloru bazowego. Kolor ten należy podać w formacie RGB, gdzie R to wartość `arg1`, G — wartość `arg2`, B — wartość `arg3`, a kanał alfa (przezroczystość) — `arg4` (dostępny od PHP 5.2.5).
`IMG_FILTER_EDGEDETECT`	Uwypuklenie krawędzi w obrazie.
`IMG_FILTER_EMBOSS`	Wytłoczenie obrazu.
`IMG_FILTER_GAUSSIAN_BLUR`	Rozmycie obrazu metodą Gaussa.
`IMG_FILTER_SELECTIVE_BLUR`	Rozmycie obrazu.
`IMG_FILTER_MEAN_REMOVAL`	Uwypuklenie krawędzi.
`IMG_FILTER_SMOOTH`	Wygładzenie (zmiękczenie, rozmycie) obrazu. Parametr `arg1` pozwala na określenie poziomu intensywności efektu.
`IMG_FILTER_PIXELATE`	Pikselizacja obrazu. Argument `arg1` określa rozmiar bloku, natomiast `arg2` ustawiony na `true` (domyślnie `false`) włącza przetwarzanie zaawansowane (większe wygładzenie obrazu wynikowego).

Można zatem napisać przykładowy skrypt, który podczas wywoływania w wierszu poleceń będzie otrzymywał nazwę pliku graficznego w formacie JPG oraz wartość całkowitą, a także wykona zmianę poziomu kontrastu tego pliku. Działający w ten sposób kod został zaprezentowany na listingu 6.9.

Listing 6.9. *Zmiana poziomu kontrastu*

```
<?php
if($argc < 3){
  exit("Wywołanie skryptu: php konwertuj.php nazwa_pliku poziom_kontrastu\n");
}

if(!$img = imagecreatefromjpeg($argv[1])){
  exit("Nie udało się wczytać pliku {$argv[1]}.\n");
}

if(!is_numeric($argv[2])){
  exit("Poziom kontrastu musi być liczbą całkowitą!\n");
}

if(!imagefilter($img, IMG_FILTER_CONTRAST, $argv[2])){
  exit("Nie udało się zmienić poziomu kontrastu w pliku {$argv[1]}.");
}

if(!imagejpeg($img, $argv[1])){
  exit("Wystąpił błąd podczas zapisu pliku {$argv[1]}.");
}

echo "Operacja zmiany kontrastu zakończona sukcesem!";
imagedestroy($img);
?>
```

Skalowanie

W celu przeskalowania obrazu do zadanych rozmiarów można użyć funkcji `imagecopyresized` lub `imagecopyresampled`. W rzeczywistości pobierają one określony parametrami wycinek z obrazu źródłowego i wstawiają go w miejsce obrazu docelowego (również określone parametrami). Jeśli wyznaczone obszary obrazu źródłowego i docelowego nie będą takie same, nastąpi odpowiednie przeskalowanie. Różnica pomiędzy `imagecopyresized` a `imagecopyresampled` jest taka, że druga z nich podczas skalowania dokonuje interpolacji pikseli, dzięki czemu uzyskuje się dokładniejszy efekt. Obie funkcje przyjmują identyczne zestawy argumentów, a ich wywołania mają postać:

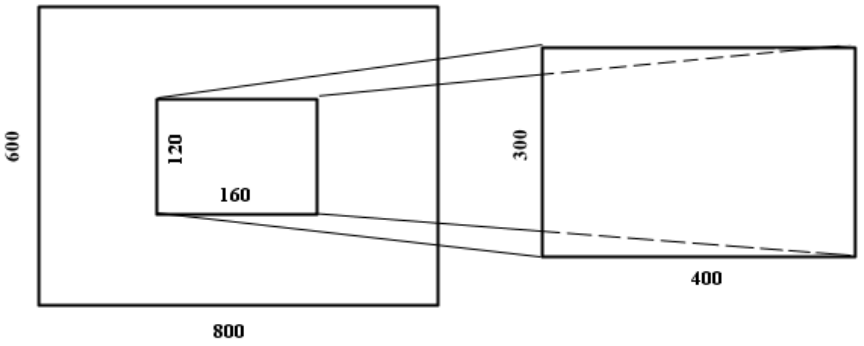
```
imagecopyresized($obraz_docelowy, $obraz_źródłowy, xd, yd, xs, ys, wd, hd, ws, hs);
imagecopyresampled($obraz_docelowy, $obraz_źródłowy, xd, yd, xs, ys, wd, hd, ws, hs);
```

gdzie:

- `$obraz_docelowy` — określa obraz docelowy,
- `$obraz_źródłowy` — określa obraz źródłowy,
- *xd* — współrzędna x lewego górnego rogu kopiowanego obszaru w obrazie docelowym,

- *yd* — współrzędna y lewego górnego rogu kopiowanego obszaru w obrazie docelowym,
- *xs* — współrzędna x lewego górnego rogu kopiowanego obszaru w obrazie źródłowym,
- *ys* — współrzędna y lewego górnego rogu kopiowanego obszaru w obrazie źródłowym,
- *wd* — szerokość kopiowanego obszaru w obrazie docelowym,
- *hd* — wysokość kopiowanego obszaru w obrazie docelowym,
- *ws* — szerokość kopiowanego obszaru w obrazie źródłowym,
- *hs* — wysokość kopiowanego obszaru w obrazie źródłowym.

Jak by to wyglądało w praktyce? Załóżmy na przykład, że mamy plik o nazwie *obraz.jpg* zawierający obraz o rozdzielczości 800×600 pikseli. Chcielibyśmy powiększyć jego środkowy fragment o wielkości 160×120 pikseli do rozdzielczości 400×300 (chodzi zatem o wykonanie operacji przedstawionej schematycznie na rysunku 6.5) i tak przetworzoną część zapisać w pliku o nazwie *obraz2.jpg*.



Rysunek 6.5. *Schemat skalowania obrazu*

Kolejność wykonywanych czynności będzie zatem następująca:

1. Wczytanie obrazu z pliku *obraz1.jpg* za pomocą funkcji `imagecreatefromjpeg`.
2. Utworzenie nowego obrazu za pomocą funkcji `imagecreatetruecolor`.
3. Przeskalowanie wycinka obrazu oryginalnego za pomocą funkcji `imagecopyresampled`.
4. Zapisanie przeskalowanego obrazu do pliku *obraz2.jpg* za pomocą funkcji `imagejpeg`.
5. Usunięcie obrazów z pamięci za pomocą funkcji `imagedestroy`.

Pozostaje jeszcze ustalenie argumentów funkcji `imagecopyresampled`. Nie jest to trudne. Ponieważ przeskalowany fragment będzie zajmował cały obszar obrazu docelowego, współrzędne lewego górnego wierzchołka to 0,0, a rozdzielczość to 400×300

pikseli. Długość i szerokość powiększanego fragmentu obrazu źródłowego też jest nam znana — to 160×120. Pozostaje więc ustalenie współrzędnych lewego górnego wierzchołka tego obszaru. Można to wyliczyć ze wzorów:

```
x = (szerokość obrazu źródłowego - szerokość powiększanego wycinka) / 2
y = (wysokość obrazu źródłowego - wysokość powiększanego wycinka) / 2
```

co w tym przypadku daje x = 320 i y = 240. Kod skryptu wykonującego przedstawione zadania został zaprezentowany na listingu 6.10.

Listing 6.10. *Skrypt skalujący środkową część obrazu do zadanej rozdzielczości*

```
<?php
if(!$img_src = imagecreatefromjpeg("obraz1.jpg"))
  exit("Nie udało się wczytać pliku obraz1.jpg.\n");

if(!$img_dest = imagecreatetruecolor(400, 300))
  exit("Nie udało się utworzyć nowego obrazu.\n");

if(!imagecopyresampled($img_dest, $img_src, 1, 1, 320, 240, 400, 300, 160, 120))
  exit("Nie udała się operacja skalowania.\n");

if(!imagejpeg($img_dest, "obraz2.jpg"))
  exit("Wystąpił błąd podczas zapisu pliku obraz2.jpg.\n");

echo "Operacja skalowania zakończona sukcesem!\n";

imagedestroy($img_src);
imagedestroy($img_dest);
?>
```

Obracanie

Do obracania obrazów służy funkcja `imagerotate`, której wywołanie ma postać:

```
imagerotate($obraz_źródłowy, kąt_obrotu, kolor_tła, ignoruj_przezroczyste);
```

Obraca ona obraz wskazywany przez *$obraz_źródłowy* o kąt *kąt_obrotu*, wypełniając ewentualne powstałe przy tym puste obszary kolorem *kolor_tła*. Ustawienie argumentu *ignoruj_przezroczyste* na wartość inną niż 0 powoduje, że fragmenty z oznaczoną przezroczystością będą ignorowane (wartość domyślna to 0; argument wprowadzony w PHP 5.1.0). Funkcja zwraca przetworzony obraz. Aby zatem obrócić obraz wskazywany przez zmienną `$img` o 45 stopni (wypełniając puste obszary kolorem białym), a obraz wynikowy przypisać zmiennej `$obraz`, należałoby wykonać instrukcje:

```
$bialy = $imagecolorallocate($img, 255, 255, 255);
$obraz = imagerotate($img, 45, $bialy);
```

Ćwiczenia do samodzielnego wykonania

Ćwiczenie 20.1. Zmodyfikuj skrypt generujący galerię z listingu 6.2 w taki sposób, aby zachowywał się prawidłowo, gdy w katalogu przeznaczonym do przechowywania obrazów:

a) nie ma żadnego pliku,

b) zapisane są pliki o rozszerzeniach innych niż *jpg*, *gif*, *png*,

c) znajdują się podkatalogi.

Ćwiczenie 20.2. Napisz taki skrypt realizujący galerię obrazów, żeby każdy obraz mógł zawierać podpis pobierany z pliku tekstowego o nazwie zgodnej z nazwą pliku z obrazem, ale o rozszerzeniu *txt*.

Ćwiczenie 20.3. Napisz skrypt, który będzie wykonywał skalowanie obrazu z pliku graficznego typu JPG (lub innego) do zadanej rozdzielczości. Nazwa pliku oraz rozdzielczość powinny być podawane w postaci argumentu.

Ćwiczenie 20.4. Napisz skrypt skalujący wszystkie obrazy typu JPG (lub innego) zapisane w danym katalogu o zadaną wartość procentową.

Ćwiczenie 20.5. Napisz skrypt wykonujący konwersje pomiędzy różnymi formatami plików graficznych.

Skorowidz

A

B

C

D

E

F

G

H

I

K

L

Ł

M

N

O

P

Q

R

S

Ś

T

U

V

W

X

Y

Z

PHP5 Praktyczny kurs

Wydanie II

Znajomość języka skryptowego PHP, szczególnie w wersji PHP 5, to w dzisiejszych czasach standard, bez którego nie może się obyć żaden szanujący się twórca stron internetowych. Środowisko PHP 5 pozwala tworzyć dynamiczne witryny, efektywnie komunikujące się z bazami danych. Zapewnia też możliwość współpracy Twojej strony z różnymi rodzajami danych, a ponadto nadaje się do tworzenia samodzielnie działających aplikacji. Jeśli marzy Ci się kariera webmastera, a nie masz jeszcze odpowiednich umiejętności w tej dziedzinie lub chcesz odświeżyć wiedzę, ta książka umożliwi Ci szybkie wejście na grunt praktycznego zastosowania możliwości języka PHP w Twoich własnych projektach.

W podręczniku „PHP 5. Praktyczny kurs. Wydanie II" znajdziesz kompletne informacje o podstawach PHP — od kwestii związanych z nazewnictwem, instalacją i konfiguracją niezbędnych narzędzi, przez omówienie zasad budowy skryptów, aż po programowanie obiektowe i obsługę sieci. Dowiesz się, jak uruchomić działającą (i atrakcyjną) witrynę internetową, jak wykorzystać możliwości grafiki i o czym należy pamiętać, aby uniknąć kłopotów z wyświetlaniem witryny w przeglądarce. Nauczysz się obsługiwać protokoły sieciowe i zrozumiesz, na czym polega obsługa sesji. Sprawdź, jak wiele możesz osiągnąć, programując w PHP!

- **Instalacja i konfiguracja narzędzi**
- **Pierwszy skrypt**
- **Zmienne, typy danych i operatory**
- **Instrukcje sterujące i funkcje**
- **Obsługa daty i czasu, ciągi znaków, tablice**
- **Operacje na strukturze systemu plików**
- **Operacje na plikach i praktyczne wykorzystanie plików**
- **Odbieranie danych z przeglądarki**
- **Wysyłanie danych do przeglądarki**
- **Obsługa cookies, sesje i wyjątki**
- **Programowanie obiektowe**
- **Obsługa grafiki**
- **Połączenia sieciowe, poczta i FTP**
- **PHP i popularne bazy danych**
- **Obiektowa współpraca z MySQL i SQLite**
- **Bazy danych w praktyce**

PHP 5 — po prostu musisz to znać!

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-3393-7

Cena 69,00 zł

Informatyka w najlepszym wydaniu