PHP5. Praktyczny kurs

Autor: Marcin Lis
ISBN: 83-246-0307-7
Format: B5, stron: 432

Znajomość języka PHP wykorzystywanego do tworzenia dynamicznych stron WWW to dziś jedna z podstawowych umiejętności wymaganych od webmastera. Era statycznych stron tworzonych za pomocą języka HTML przeminęła już dawno, a dynamiczne generowanie treści stało się obecnie standardem. Język PHP i baza danych MySQL stanowią platformę o wielkich możliwościach, stosowaną do tworzenia zarówno wielkich portali i witryn e-commerce, jak i prostych stron prywatnych.

Książka „PHP5. Praktyczny kurs" jest wprowadzeniem w tajniki nowoczesnego tworzenia stron WWW. Przedstawia proces instalacji języka PHP wraz z serwerami WWW, podstawowe konstrukcje języka i sposoby wykorzystania ich w procesie budowania witryny. Opisuje takie zagadnienia, jak współpraca z systemem plików, obsługa sesji, możliwości programowania obiektowego oraz współpraca z bazami danych. Dowiesz się z niej także, jak za pomocą PHP generować i przetwarzać elementy graficzne oraz wykorzystywać protokoły sieciowe.

- Instalacja i konfiguracja PHP w różnych systemach operacyjnych
- Zmienne, typy danych i operatory
- Instrukcje sterujące
- Komunikacja skryptów z przeglądarką
- Obsługa plików cookie i sesji
- Operacje na plikach
- Programowanie obiektowe w PHP
- Wykorzystywanie danych zgromadzonych w bazie

Poznaj i wykorzystaj w swoich projektach możliwości języka PHP

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 3.

Współpraca z przeglądarką

Lekcja 10. Odbieranie danych z przeglądarki

Formularze HTML

Skrypty PHP bardzo często są wykorzystywane do odbierania i przetwarzania danych pochodzących z przeglądarki użytkownika. Najczęściej polega to na tym, że osoba przeglądająca stronę WWW wprowadza różne informacje do formularza HTML, które następnie, zazwyczaj po kliknięciu przycisku, są wysyłane do serwera, gdzie zajmuje się nimi kod PHP. Aby taka procedura mogła zostać wykonana, formularz musi zawierać parametry `action` oraz `method`, zatem jego ogólna postać będzie następująca:

```
<form  name = "nazwa"
       target = "okno"
       action = "url"
       method = "metoda"
       enctype = "typ kodowania">
<!--tu definicja obiektów składowych-->
</form>
```

Parametr `method` wskazuje metodę, która zostanie użyta do przesłania danych do serwera, może to być `GET` lub `POST`. Natomiast `action` określa adres skryptu, który będzie odbierał dane. Może on być adresem bezwzględnym, np. *http://www.mojadomena.com/skrypt.php*, lub względnym, np. */skrypty/skrypt.php*. W pierwszym przypadku dane zostaną dostarczone do skryptu *skrypt.php* znajdującego się w głównym katalogu serwera o adresie *http://www.mojadomena.com*, natomiast w drugim — do skryptu o nazwie *skrypt.php* znajdującego się w podkatalogu *skrypty* na serwerze ze stroną WWW zawierającą formularz.

Elementami składowymi formularza mogą być:

- `button` — klasyczny przycisk,
- `checkbox` — pole wyboru,
- `hidden` — element ukryty,
- `password` — pole tekstowe do wpisywania haseł,
- `radio` — pole wyboru,
- `reset` — przycisk *reset*,
- `select` — lista wyboru,
- `submit` — przycisk *submit*,
- `text` — pole tekstowe,
- `textarea` — rozszerzone pole tekstowe.

Każdy z tych elementów powinien mieć określony parametr `name`, dzięki któremu będzie możliwa jego identyfikacja w skrypcie PHP.

Wysyłanie metodą GET

Metoda `GET` służy do przesyłania stosunkowo niewielkich ilości danych, np. krótkich formularzy tekstowych. Wynika to z faktu, że są one przesyłane w adresie URL, który będzie miał wtedy schematyczną postać:

```
http://adres.serwera/skrypt.php?parametr1=wartość1&parametr2=wartość2
```

Jeśli weźmiemy pod uwagę, że adres URL ma zwykle ograniczoną długość (zależy to od przeglądarki oraz serwera), a także że może zawierać jedynie znaki alfanumeryczne, jasnym jest, dlaczego ta metoda jest wykorzystywana jedynie do wymiany prostych danych tekstowych.

Aby zobaczyć, jak to działa w praktyce, napiszmy kod prostego formularza zawierającego jedno pole tekstowe, który będzie wysyłał dane do skryptu o nazwie *skrypt.php* umieszczonego w katalogu głównym naszego serwera WWW. Kod HTML zawierający taki formularz jest widoczny na listingu 3.1.

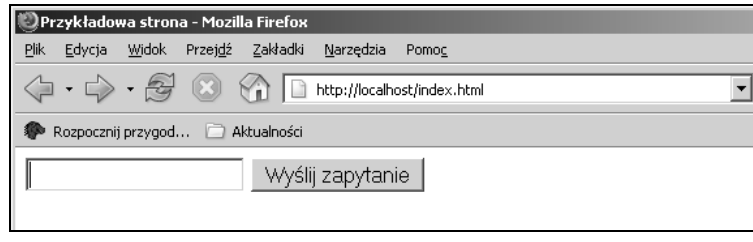
Listing 3.1. *Kod przykładowego formularza HTML*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Przykładowa strona</title>
</head>
<body>
<form method="GET"
      action="http://127.0.0.1/skrypt.php">
  <input type="text" name="pole1">
```

```
  <input type="submit">
</form>
</body>
</html>
```

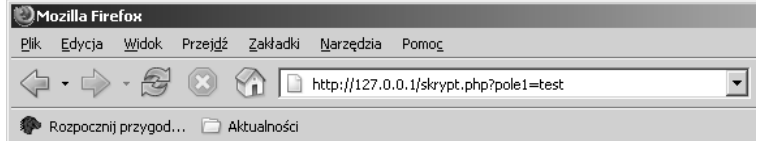
Formularz (widoczny na rysunku 3.1) zawiera jedno pole tekstowe o nazwie `pole1` oraz przycisk typu *Submit*, którego kliknięcie powoduje przesłanie danych do serwera. Wartość parametru `method` to `get`, a zatem dane zostaną przesłane do serwera za pomocą metody *GET*. Nazwę skryptu oraz adres serwera wskazuje parametr `action`, wywołany zostanie więc skrypt o nazwie *skrypt.php* znajdujący się na serwerze o lokalnym adresie *127.0.0.1* (prawidłowe byłoby również użycie adresu *http://localhost/skrypt.php*). Ponieważ zarówno kod HTML, jak i kod skryptu będą się znajdowały na tym samym serwerze w tym samym katalogu, można również pominąć w parametrze `action` dane dotyczące serwera. Tym samym mógłby mieć on postać:

```
action="skrypt.php"
```



Rysunek 3.1. *Formularz generowany przez kod z listingu 3.1*

Jeśli do pola tekstowego wprowadzimy przykładowy ciąg znaków, np. `test`, oraz klikniemy przycisk *Wyślij zapytanie* (napis na przycisku może być różny w zależności od zastosowanej przeglądarki), to powstanie URL w postaci *http://127.0.0.1/skrypt.php?pole1=test*, tak jak na rysunku 3.2. Przetworzony zostanie zatem skrypt znajdujący się na serwerze lokalnym *127.0.0.1*, w pliku *skrypt.php* i zostaną mu przekazane wartości znajdujące się za znakiem `?` odnośnika. W skrypcie możemy te wartości odczytać i wykorzystać do własnych celów. Dowiedzmy się zatem, jak to zrobić.



Rysunek 3.2. *Odnośnik generowany w metodzie GET*

Dostęp do danych z formularza jest możliwy na trzy sposoby. Sposobem najstarszym i obecnie niezalecanym jest wykorzystanie globalnej tablicy `$HTTP_GET_VARS`. Jako indeks tablicy należy podać nazwę pola formularza, z którego chcemy odczytać dane. Schematycznie taka konstrukcja będzie miała postać:

```
$zmienna = $HTTP_GET_VARS['nazwa_pola'];
```

Tego typu odwołanie należy stosować jedynie wtedy, gdy konieczna jest kompatybilność ze starszymi wersjami PHP (PHP3), zazwyczaj jednak nie ma takiej potrzeby. Aby skorzystać z tego typu odwołań w PHP5, należy w pliku konfiguracyjnym *php.ini* włączyć domyślnie wyłączoną opcję `register_long_arrays` (`register_long_arrays = On`).

Drugim sposobem dostępu do danych z formularza jest wykorzystanie globalnej tablicy `$_GET`. Jako indeks tablicy należy zastosować, podobnie jak w poprzednim przypadku, nazwę pola, z którego dane chcemy odczytać. Jest to też zalecany sposób odczytu i będziemy go stosować w dalszej części książki. Schematycznie konstrukcja taka ma postać:

```
$zmienna = $_GET ['nazwa_pola'];
```

Sposób trzeci to dostęp bezpośredni. Jeżeli w pliku konfiguracyjnym *php.ini* włączymy opcję `register_globals`[1], dostęp do pól formularza będzie odbywał się tak samo jak do zwykłych zmiennych. Oznacza to, że jeśli w formularzu będzie występowało pole o nazwie `pole1`, to w skrypcie będzie można się odwołać do niego tak, jak do zmiennej `$pole1`. Ten sposób, choć wydaje się najwygodniejszy, może jednak prowadzić do powstawania błędów w skryptach, gdyż zmienne związane z formularzem nie odróżniają się od zmiennych lokalnych skryptu. Stosujmy ten sposób jedynie wtedy, gdy jest to naprawdę potrzebne i dokładnie wiemy, jakie mogą być konsekwencje.

Na listingu 3.2 został przedstawiony skrypt odczytujący wartość pola tekstowego z formularza z listingu 3.1 z wykorzystaniem wszystkich trzech sposobów.

Listing 3.2. *Różne sposoby odczytu danych z formularza*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Przykładowa strona</title>
</head>
<body>
<p> Dane wprowadzone do formularza: </p>
<?php
//sposób pierwszy
$zmienna = $HTTP_GET_VARS['pole1'];
echo("1. Wartość pola pole1 to $zmienna <br />");

//sposób drugi
$zmienna = $_GET['pole1'];
echo("2. Wartość pola pole1 to $zmienna <br />");

//sposób trzeci
$zmienna = $pole1;
echo("3. Wartość pola pole1 to $zmienna");
?>
</body>
</html>
```

Przy przetwarzaniu formularzy bardzo przydaje się funkcja `isSet`, która pozwala stwierdzić, czy dane pole formularza zostało ustawione, innymi słowy czy została przekazana do skryptu wartość odpowiadająca temu polu. Aby pokazać sposób wykorzystania tej funkcji, utwórzmy formularz składający się z trzech pól wyboru typu `radio`. Odpowiedni kod HTML został zaprezentowany na listingu 3.3, a jego wygląd widoczny jest na ry-

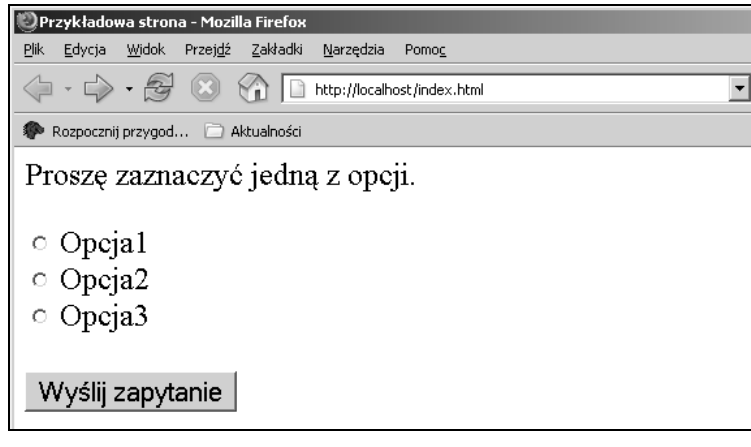
[1] Począwszy od PHP w wersji 4.2.0, a więc także we wszystkich wersjach PHP5, opcja ta jest domyślnie wyłączona.

sunku 3.3. Każde z pól ma taką samą wartość parametru `name`, a zatem tworzą one jedną grupę. Dzięki temu na raz może być zaznaczone tylko jedno pole, są to więc opcje wykluczające. Każde pole ma jednak inną wartość parametru `value`, dzięki czemu w skrypcie będziemy je mogli łatwo zidentyfikować. Parametr `action` formularza wskazuje, podobnie jak we wcześniejszych przykładach, na skrypt znajdujący się w pliku *skrypt.php* na serwerze lokalnym o adresie *127.0.0.1*.

Listing 3.3. *Formularz zawierający elementy typu radio*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Przykładowa strona</title>
</head>
<body>
<P>Proszę zaznaczyć jedną z opcji.</P>
<form method="get"
      action="http://127.0.0.1/skrypt.php">
  <input type="radio" name="radio1" value="opcja1">
  Opcja1
  <br />
  <input type="radio" name="radio1" value="opcja2">
  Opcja2
  <br />
  <input type="radio" name="radio1" value="opcja3">
  Opcja3
  <br /><br />
  <input type="submit">
</form>
</body>
</html>
```

Rysunek 3.3. *Wygląd formularza generowanego przez kod z listingu 3.3*



Napiszmy więc teraz skrypt *skrypt.php*. Pozwoli nam on stwierdzić, czy została zaznaczona jakaś opcja i ewentualnie która. W wykonaniu tego zadania bardzo nam pomoże wspomniana przed chwilą funkcja `isSet`. Spójrzmy na kod widoczny na listingu 3.4 — `isSet` jest tu używana do stwierdzenia, czy w tablicy `$_GET` jest ustawiony klucz o nazwie

radio1, a tym samym czy do skryptu została przekazana wartość pola radio o nazwie radio1. Jeśli tak, funkcja zwraca wartość true, jeśli nie — wartość false. Takie zachowanie pozwala na użycie instrukcji warunkowej if do wyświetlenia właściwego komunikatu na ekranie.

Listing 3.4. *Obsługa pól typu radio*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Wynik działania skryptu</title>
</head>
<body>
<p>
<?php
if(!isSet($_GET['radio1'])){
  echo("Proszę zaznaczyć jedną z opcji!");
}
else{
  echo("Zaznaczona opcja to {$_GET['radio1']}.");
}
?>
</p>
</body>
</html>
```

Metoda POST

Metoda POST to drugi sposób przesyłania danych do serwera. Główne różnice dla użytkownika to możliwość przesłania dużo większej ilości danych (np. plików binarnych) oraz to, że nie można ich zobaczyć w polu adresu przeglądarki (co wydaje się całkiem logiczne). Maksymalna ilość danych, jakie mogą być przesłane za pomocą tej metody, jest ograniczona przez znajdującą się w pliku *php.ini* opcję konfiguracyjną post_max_size[2]. Domyślnie jest to 8 MB. Prosty formularz wykorzystujący przesyłanie danych metodą POST został przedstawiony na listingu 3.5. Jak widać, jedyną różnicą w stosunku do metody GET (formularz z listingu 3.1) jest zmiana wartości parametru method znacznika form. Również i w tym przypadku parametr action mógłby mieć postać:

```
action="http://localhost/skrypt.php"
```

lub po prostu:

```
action="skrypt.php"
```

Listing 3.5. *Kod formularza HTML wysyłającego dane metodą POST*

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

[2] Na maksymalną ilość odbieranych danych mogą również mieć wpływ ustawienia serwera WWW.

```
<title>Przykładowa strona</title>
</head>
<body>
<form method="post"
      action="http://127.0.0.1/skrypt.php">
  <input type="text" name="pole1">
  <input type="submit">
</form>
</body>
</html>
```

W skrypcie PHP wartości przesłane z formularza można odczytać, podobnie jak miało to miejsce w przypadku metody `GET`, na trzy sposoby. Sposobem pierwszym (obecnie niezalecanym; niezbędne jest włączenie opcji konfiguracyjnej `register_long_arrays`) jest wykorzystanie globalnej tablicy `$HTTP_POST_VARS`. Jako indeks tablicy należy podać nazwę pola formularza, z którego chcemy odczytać dane. Schematycznie taka konstrukcja ma postać:

```
$zmienna = $HTTP_POST_VARS['nazwa_pola'];
```

Należy jej użyć, jeśli niezbędne jest zachowanie kompatybilności z wersją 3 PHP.

Drugim sposobem dostępu jest wykorzystanie globalnej tablicy `$_POST`. Jako indeks tablicy należy zastosować również nazwę pola formularza, z którego chcemy odczytać dane. Jest to polecany sposób odwoływania się do danych i będzie on stosowany w dalszej części książki, o ile zaistnieje potrzeba zastosowania metody `POST`. Schematycznie konstrukcja taka ma postać:

```
$zmienna = $_POST['nazwa_pola'];
```

Sposób trzeci to dostęp bezpośredni, niezbędne jest w tym przypadku włączenie w pliku konfiguracyjnym *php.ini* opcji `register_globals`. Dostęp do pól formularza może się wtedy odbywać tak samo, jak w przypadku zwykłych zmiennych. Oznacza to, że jeśli w formularzu będzie występowało pole o nazwie `pole1`, to w skrypcie będzie można się odwołać do niego tak, jak do zmiennej `$pole1`. Jak już wiemy, ta metoda też nie jest polecana, gdyż nie pozwala na odróżnienie zwykłych zmiennych od tych powstałych ze względu na otrzymanie danych z formularza, co może powodować trudne do wykrycia błędy.

Na listingu 3.6 został przedstawiony skrypt odczytujący wartość pola tekstowego z formularza z listingu 3.5 z wykorzystaniem wszystkich trzech wymienionych sposobów. Przykładowy efekt działania skryptu w przypadku, kiedy do pola tekstowego formularza został wprowadzony ciąg znaków `test`, został przedstawiony na rysunku 3.4. Jak widać, przy prostej obsłudze formularzy różnice pomiędzy metodami *GET* i *POST* są niewielkie, jednak ta druga pozwoli nam wykonywać bardziej zaawansowane zadania, jak np. przesyłanie plików do serwera, czym zajmiemy się już w kolejnej sekcji.

Listing 3.6. *Skrypt odczytujący dane przekazane z formularza za pomocą metody POST*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Wynik działania skryptu</title>
</head>
<body>
<p>
<?php
//sposób pierwszy
$zmienna = $HTTP_POST_VARS['pole1'];
echo("1. Wartość pola pole1 to $zmienna <br />");

//sposób drugi
$zmienna = $_POST['pole1'];
echo("2. Wartość pola pole1 to $zmienna <br />");

//sposób trzeci
$zmienna = $pole1;
echo("3. Wartość pola pole1 to $zmienna");
?>
</p>
</body>
</html>
```

Rysunek 3.4. *Różne sposoby odczytu przesłanych danych dają taki sam efekt*

Wysyłanie plików do serwera

Aby wysłać plik z komputera użytkownika na serwer, trzeba przygotować odpowiedni formularz HTML umożliwiający wybór pliku oraz skrypt PHP, który go odbierze. Również środowisko PHP musi być odpowiednio skonfigurowane. W pliku *php.ini* musi być włączona opcja `file_uploads`, zmienna `upload_tmp_dir` powinna wskazywać na katalog, w którym będą zapisywane dane tymczasowe podczas ich pobierania, zmienna `upload_max_filesize` powinna określać maksymalny rozmiar pojedynczego pliku (standardowo 2 MB). Jeżeli katalog tymczasowy nie zostanie podany, wykorzystany będzie katalog systemowy. Należy zwrócić uwagę, aby uprawnienia dostępu do katalogu tymczasowego były ustawione tak, aby PHP miało możliwość zapisywania w nim danych. Maksymalna wielkość pliku zależy również od opcji `post_max_size` (standardowo 8 MB) i `memory_limit` (standardowo 8 MB). W większości przypadków ustawienia domyślne są jednak wystarczające i w standardowych zastosowaniach nie ma potrzeby ich modyfikować.

Formularz HTML należy zdefiniować za pomocą znacznika `<form>`, który powinien zawierać następujące parametry:

`name` — określa nazwę formularza.

`enctype` — określa typ kodowania `MIME`, w tym przypadku będzie to `multipart/form-data`.

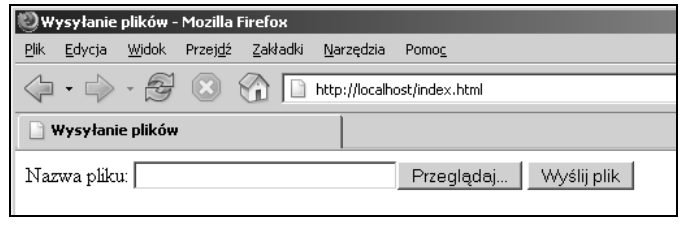
`action` — określa adres skryptu PHP.

`method` — określa metodę wysyłania danych, w tym przypadku będzie to metoda *POST*.

W formularzu należy umieścić dwa pola `input`, jedno typu `file`, a drugie typu `submit`. Polu typu `file`, służącemu do wyboru pliku, nadamy nazwę (parametr `name`) `plik1`. Pozwoli ona na zidentyfikowanie danych z tego pola w skrypcie PHP. Kod przykładowego formularza został zaprezentowany na listingu 3.7. Gdy zostanie wczytany do przeglądarki, będzie miał natomiast postać widoczną na rysunku 3.5.

Listing 3.7. *Formularz HTML służący do wysyłania plików*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Wysyłanie plików</title>
</head>
<body>
<form name    = "formularz1"
      enctype = "multipart/form-data"
      action  = "http://127.0.0.1/upload.php"
      method  = "POST">
Nazwa pliku:
<input type  = "file"
       name  = "plik1"
       size  = "30"
       value = "">
<input type  = "submit"
       name  = "wyslij"
       value = "Wyślij plik">
</form>
</body>
</html>
```



Rysunek 3.5. *Wygląd formularza do wysyłania plików*

Plik wysłany za pomocą takiego formularza do serwera zostanie zapisany w katalogu służącym do przechowywania plików tymczasowych. Katalog ten możemy ustalić samodzielnie, ustawiając w pliku *php.ini* opcję konfiguracyjną `upload_tmp_dir`. W systemie Linux będzie to np.:

```
upload_tmp_dir = /var/www/upload
```

a w systemie Windows:

```
upload_tmp_dir = c:\www\upload\
```

Skrypt odbierający dane uzyska dostęp do globalnej tablicy `$_FILES` zawierającej informacje niezbędne do ich dalszego przetworzenia. Jest to tablica asocjacyjna, w której plik identyfikowany jest przez nazwę pola `input` (typu `file`) z formularza WWW. W przypadku formularza z listingu 3.7 ta nazwa to `plik1`. Pod tym indeksem znajduje się 5 innych, które pozwalają na odczytanie informacji o pliku. Są to:

- `$_FILES['plik1']['name']` — oryginalna nazwa pliku (którą miał on na komputerze użytkownika).
- `$_FILES['plik1']['type']` — typ MIME pliku (o ile przeglądarka dostarczyła tę informację).
- `$_FILES['plik1']['size']` — wielkość pliku w bajtach.
- `$_FILES['plik1']['tmp_name']` — nazwa tymczasowa, pod jaką plik został zapisany na serwerze.
- `$_FILES['plik1']['error']` — status operacji, kod błędu.

Pole `error` (dostępne w PHP od wersji 4.2.0) może przyjmować jedną z wartości:

- `UPLOAD_ERR_OK` — brak błędu, operacja została zakończona sukcesem.
- `UPLOAD_ERR_INI_SIZE` — wielkość pliku przekracza wielkość maksymalną zdefiniowaną w pliku *php.ini* (zmienna `upload_max_filesize`).
- `UPLOAD_ERR_FORM_SIZE` — rozmiar pliku przekracza wielkość maksymalną zdefiniowaną w formularzu HTML.
- `UPLOAD_ERR_PARTIAL` — została odebrana jedynie część pliku.
- `UPLOAD_ERR_NO_FILE` — plik nie został pobrany.

Plik wysłany do serwera jest umieszczany w katalogu tymczasowym i należy go przenieść do właściwej lokalizacji docelowej (np. katalogu, który przeznaczyliśmy do przechowywania tego typu danych). Wykorzystuje się w tym celu funkcję `move_uploaded_file`, która dodatkowo ze względów bezpieczeństwa oprócz zmiany lokalizacji pliku wykonuje sprawdzenie, czy na pewno został on wysłany do serwera za pomocą metody `HTTP_POST`.

Powyższe informacje w zupełności wystarczą do napisania skryptu odbierającego pliki wysyłane z przeglądarki i zapisującego je w wybranym katalogu. Został on przedstawiony na listingu 3.8 i należy go zapisać pod nazwą `upload.php` w katalogu głównym serwera WWW.

Listing 3.8. *Skrypt odbierający pliki wysłane do serwera*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Wynik odbioru pliku</title>
</head>
<body>
<p>
<?php
$uploaddir = './';

if($_FILES['plik1']['error'] == UPLOAD_ERR_OK){
  $new_name = $uploaddir.$_FILES['plik1']['name'];
  $temp_name = $_FILES['plik1']['tmp_name'];
  if(move_uploaded_file($temp_name, $new_name)){
    echo "Plik został załadowany.\n";
  }
  else{
    echo "Nieprawidłowy plik\n";
  }
}
else{
  echo("Wystąpił błąd: ");
  switch($_FILES['plik1']['error']){
    case UPLOAD_ERR_INI_SIZE  :
    case UPLOAD_ERR_FORM_SIZE :
      echo("Przekroczony maksymalny rozmiar pliku!\n");
      break;
    case UPLOAD_ERR_PARTIAL :
      echo("Odebrano tylko część pliku!\n");
      break;
    case UPLOAD_ERR_NO_FILE :
      echo("Plik nie został pobrany!\n");
      break;
    default :
      echo("Nieznany typ błędu!\n");
  }
}
?>
</p>
</body>
</html>
```

Katalog, w którym mają być zapisywane odbierane pliki, jest wskazywany przez zmienną `$uploaddir`. Oczywiście musi on istnieć w systemie plików serwera. W przykładzie został zastosowany ciąg `./`, co oznacza katalog bieżący (katalog, w którym znajduje się skrypt `upload.php`). Należy również pamiętać, że musi on mieć odpowiednio ustawione prawa dostępu (aparat wykonawczy PHP musi mieć możliwość zapisu).

Pierwszą czynnością wykonywaną w skrypcie jest sprawdzenie, czy pole `error` tablicy `$_FILES` zawiera wartość `UPLOAD_ERR_OK`, a zatem czy plik został odebrany bez problemów. Jeśli tak, z tablicy `$_FILES` jest odczytywana oryginalna nazwa pliku oraz nazwa tymczasowa, pod którą został on zapisany na serwerze. Do nazwy oryginalnej dołączana jest nazwa katalogu zawarta w zmiennej `$uploaddir` i całość jest zapisywana w zmiennej `$new_name`. Nazwa tymczasowa jest z kolei zapisywana w zmiennej `$temp_name`. Zmienne te (zostały one wprowadzone do skryptu w celu zwiększenia przejrzystości kodu) są następnie wykorzystywane jako parametry funkcji `move_uploaded_file`. Jeśli wykonanie tej funkcji zakończy się sukcesem i plik zostanie przeniesiony pod

oryginalną nazwą do katalogu wskazywanego przez `$uploaddir`, funkcja zwróci wartość `true`, a w przypadku przeciwnym — `false`. W zależności od tej wartości na ekranie jest wyświetlany odpowiedni komunikat.

Jeżeli jednak wartość pola `error` tablicy `$_FILES` jest różna od `UPLOAD_ERR_OK`, wykonywana jest instrukcja `switch` pozwalająca na stwierdzenie, jakiego typu błąd wystąpił. Badane są wszystkie pozostałe możliwości stanu pola `error` i wyświetlany jest odpowiedni komunikat. Następnie ma miejsce sprawdzenie, czy plik w ogóle został wysłany, czy jego wielkość nie przekroczyła maksymalnego rozmiaru oraz czy został on załadowany w całości. Klauzula `default` instrukcji `switch` zabezpiecza nas przed sytuacją, kiedy w polu `error` znalazłby się nieznany kod błędu (np. wprowadzony w kolejnych wersjach PHP). Dzięki niej również w takiej sytuacji zostanie wyświetlona odpowiednia informacja.

Ćwiczenia do samodzielnego wykonania

Ćwiczenie 10.1. Napisz skrypt obliczający pierwiastki równania kwadratowego o parametrach wprowadzanych w formularzu HTML.

Ćwiczenie 10.2. Napisz skrypt umożliwiający wykonywanie czterech podstawowych działań arytmetycznych na dwóch argumentach. Wartości argumentów mają być wprowadzane poprzez pola tekstowe formularza, a wybór działania ma następować poprzez pola wyboru typu `radio`.

Ćwiczenie 10.3. Napisz skrypt wykonujący konwersję tekstu wprowadzonego w polu tekstowym formularza ze standardu Windows-1250 do ISO-8859-2.

Ćwiczenie 10.4. Napisz skrypt umożliwiający użytkownikowi wysłanie pliku do serwera, umieszczający ten plik w wybranym katalogu i zwracający odnośnik do niego tak, żeby było możliwe jego pobranie z poziomu przeglądarki.