

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

PHP w mgnieniu oka

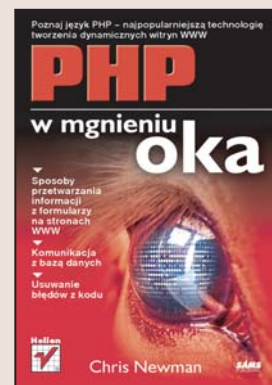
Autor: Chris Newman

Tłumaczenie: Radosław Meryk

ISBN: 83-246-0043-4

Tytuł oryginału: [Teach Yourself PHP in 10 Minutes](#)

Format: A5, stron: 272



Poznaj język PHP — najpopularniejszą technologię tworzenia dynamicznych witryn WWW

- Sposoby przetwarzania informacji z formularzy na stronach WWW
- Komunikacja z bazą danych
- Usuwanie błędów z kodu

Dynamiczne witryny WWW stopniowo wypierają takie, które opierają się wyłącznie na statycznych treściach. Możliwość połączenia stron z bazą danych pozwoliła na tworzenie portali, sklepów internetowych, katalogów, forów dyskusyjnych i wielu innych witryn, bez których trudno sobie wyobrazić oblicze współczesnego internetu. Spośród technologii wykorzystywanych do tworzenia dynamicznych witryn WWW największą popularnością cieszy się PHP. Tę dostępną nieodpłatnie platformę można uruchomić w każdym systemie operacyjnym. PHP jest językiem stosunkowo prostym. Pisane w nim skrypty są osadzone w kodzie strony WWW i interpretowane podczas jej wyświetlania w przeglądarce.

„PHP w mgnieniu oka” to książka dla tych, którzy chcą szybko poznać najważniejsze elementy programowania w języku PHP. Opisuje podstawowe zagadnienia związane ze zmiennymi, słowami kluczowymi i organizacją kodu. Czytając ją, dowiesz się, jak za pomocą PHP przetwarzać różne typy danych, jak osadzać kod PHP wewnątrz kodu HTML i jak tworzyć elementy stron WWW. Nauczysz się również łączyć skrypty PHP z zewnętrznymi programami, serwerami i usługami, przeczytasz o komunikacji z bazą danych, obsłudze systemu plików i usuwaniu błędów ze skryptów.

- Zmienne i stałe w PHP
- Sterowanie przebiegiem programu
- Stosowanie funkcji
- Przetwarzanie różnych typów danych
- Obsługa formularzy HTML, sesji i plików cookie
- Uwierzytelnianie użytkowników
- Komunikacja z serwerem WWW
- Łączenie z bazą danych
- Obsługa błędów
- Klasy z repozytorium PEAR

Jeśli chcesz nauczyć się PHP, ale nie masz na to zbyt wiele czasu, koniecznie przeczytaj tę książkę.



Spis treści

O Autorze	7
Wprowadzenie: PHP wita!	9
Część I Podstawy PHP	15
Rozdział 1. Wprowadzenie w tematykę PHP	17
Podstawy PHP	17
Pierwszy skrypt	20
Podsumowanie	24
Rozdział 2. Zmienne	25
Podstawowe wiadomości o zmiennych	25
Typy danych	28
Podsumowanie	31
Rozdział 3. Przepływ sterowania	33
Instrukcje warunkowe	33
Pętle	39
Podsumowanie	41
Rozdział 4. Funkcje	43
Zastosowanie funkcji	43
Argumenty i zwracane wartości	45
Wykorzystanie plików bibliotecznych	49
Podsumowanie	50
Część II Przetwarzanie danych	51
Rozdział 5. Przetwarzanie liczb	53
Operacje arytmetyczne	53
Liczbowe typy danych	56
Funkcje przetwarzające liczby	57
Podsumowanie	60

Rozdział 6. Przetwarzanie ciągów znaków	61
Anatomia ciągów znaków	61
Formatowanie ciągów znaków	64
Funkcje przetwarzania ciągów znaków	68
Podsumowanie	70
Rozdział 7. Tablice	71
Czym są tablice?	71
Funkcje przetwarzania tablic	75
Tablice wielowymiarowe	78
Podsumowanie	80
Rozdział 8. Wyrażenia regularne	81
Wprowadzenie do wyrażeń regularnych	81
Zastosowanie funkcji ereg	82
Podsumowanie	89
Rozdział 9. Przetwarzanie godzin i dat	91
Formaty daty	91
Przetwarzanie znaczników czasu	93
Podsumowanie	97
Rozdział 10. Wykorzystanie klas	99
Obiektowe własności języka PHP	99
Czym jest klasa?	100
Tworzenie i wykorzystywanie obiektów	101
Podsumowanie	105
Część III Środowisko webowe	107
Rozdział 11. Przetwarzanie formularzy HTML	109
Przesyłanie formularzy do PHP	109
Obsługa formularzy za pomocą PHP	114
Skrypt do przesyłania wiadomości e-mail za pomocą formularza	116
Podsumowanie	118
Rozdział 12. Generowanie dynamicznego HTML	119
Ustawianie wartości domyślnych	119
Tworzenie elementów formularza	123
Podsumowanie	127
Rozdział 13. Weryfikacja poprawności danych w formularzach	129
Zapewnienie wypełnienia pól obowiązkowych	129
Wyświetlanie ostrzeżeń dotyczących poprawności danych	131
Wymuszanie reguł dotyczących danych	133
Wyróżnianie pól wymagających uwagi	134
Podsumowanie	136
Rozdział 14. Pliki cookie i sesje	137
Pliki cookie	137
Sesje	141
Podsumowanie	143

Rozdział 15. Uwierzytelnianie użytkowników	145
Typy uwierzytelniania	145
Tworzenie systemu uwierzytelniania	148
Podsumowanie	153
Rozdział 16. Komunikacja z serwerem WWW	155
Nagłówki HTTP	155
Zmienne środowiskowe serwera	160
Podsumowanie	162
Część IV Wykorzystanie innych usług z poziomu PHP	163
Rozdział 17. Dostęp do systemu plików	165
Zarządzanie plikami	165
Odczytywanie i zapisywanie plików	168
Podsumowanie	174
Rozdział 18. Wykonywanie programów na serwerze WWW	175
Wykonywanie programów na hoście	175
Środowisko hosta	178
Zagadnienia bezpieczeństwa	181
Podsumowanie	182
Rozdział 19. Wykorzystanie bazy danych MySQL	183
Wykorzystanie MySQL	183
Wykonywanie instrukcji SQL	185
Uruchamianie diagnostyczne instrukcji SQL	189
Podsumowanie	191
Rozdział 20. Abstrakcja bazy danych	193
Klasa PEAR DB	193
Zagadnienia związane z przenośnością baz danych	199
Podsumowanie	202
Rozdział 21. Uruchamianie skryptów PHP z wiersza polecenia	203
Środowisko wiersza polecenia	203
Pisanie skryptów przeznaczonych do uruchamiania w wierszu polecenia	207
Podsumowanie	210
Rozdział 22. Obsługa błędów	211
Zgłaszanie błędów	211
Podsumowanie	218
Część V Konfigurowanie i rozszerzenia PHP	219
Rozdział 23. Konfiguracja PHP	221
Ustawienia konfiguracji	221
Dyrektywy konfiguracji	224
Ładowalne moduły	229
Podsumowanie	230

Rozdział 24. Bezpieczeństwo PHP	231
Tryb bezpieczny	231
Inne mechanizmy zabezpieczeń	234
Podsumowanie	238
Rozdział 25. Repozytorium PEAR	239
Co to jest PEAR?	239
Korzystanie z repozytorium PEAR	241
Podsumowanie	245
Dodatki	247
Dodatek A Instalacja PHP	249
Instalacje na platformie Linux (Unix)	249
Instalacja w systemie Windows	253
Rozwiązywanie problemów	254
Skorowidz	255

Rozdział 2.

Zmienne

W tym rozdziale nauczymy się przypisywać wartości do zmiennych w PHP oraz wykorzystywać je w prostych wyrażeniach.

Podstawowe wiadomości o zmiennych

Zmienne — kontenery, w których można zapisywać wartości do późniejszego wykorzystania, są zasadniczymi komponentami każdego języka programowania.

Na przykład w skrypcie może występować zmienna o nazwie `number`, w której zapisano wartość `5`, albo zmienna o nazwie `name`, w której zapisano wartość `Krzysztof`. Zmienne o podanych nazwach i wartościach zadeklarowano w następującym kodzie:

```
$number = 5;  
$name = "Krzysztof";
```

Zmienne w PHP zawsze poprzedza się znakiem dolara. Jeśli się o tym pamięta, zadeklarowanie nowej zmiennej jest bardzo proste: wystarczy po lewej stronie znaku równości umieścić nazwę zmiennej, a po prawej wartość, którą chcemy do niej przypisać.



Deklarowanie zmiennych. Inaczej niż w niektórych językach programowania, w PHP nie trzeba deklarować zmiennych przed ich wykorzystaniem. Aby zadeklarować zmienną, wystarczy przypisać do niej wartość. Można to zrobić w dowolnym momencie.

Zmienne można wykorzystywać zamiast literalów w dowolnym miejscu kodu PHP. W zaprezentowanym tu kolejnym przykładzie wykorzystano instrukcję `echo` w celu wyświetlenia wartości zapisanej w zmiennej. Robi się to identycznie jak w przypadku wyświetlania literału tekstowego:

```
$name = "Krzysztof";  
echo "Cześć, ";  
echo $name;
```

Wynik działania tego kodu jest następujący:

```
Cześć, Krzysztof
```

Nazwy zmiennych

Im bardziej opisowe są nazwy zmiennych, tym łatwiej rozpoznać, do czego są przeznaczone. Jest to szczególnie ważne jeśli sięgamy do skryptu kilka miesięcy po jego napisaniu.

Ogólnie rzecz biorąc, nie należy do dobrego stylu stosowanie takich nazw zmiennych jak `$a`, `$b` itd. Prawdopodobnie niezbyt długo będziemy pamiętać, czego dotyczą określone litery. Dobra nazwa zmiennej zawiera informacje o wartościach, jakich dotyczy (na przykład `$cena` lub `$nazwisko`).



Wielkość liter. W PHP w nazwach zmiennych rozróżniane są wielkie i małe litery — na przykład zmienna `$nazwisko` to co innego niż `$Nazwisko`. W tym samym skrypcie w obu zmiennych mogą być zapisane inne wartości.

W nazwach zmiennych mogą występować tylko litery, cyfry i znaki podkreślenia. Nazwa zmiennej może rozpoczynać się literą bądź znakiem podkreślenia. Przykłady prawidłowych i nieprawidłowych nazw zmiennych zestawiono w tabeli 2.1.



Zastosowanie znaków podkreślenia. Wykorzystanie znaków podkreślenia to dobry sposób nadania zmiennym nazw składających się z dwóch lub większej liczby słów. Na przykład nazwy zmiennych `$imie_nazwisko` oraz `$data_urodzenia` są bardziej czytelne właśnie dzięki zastosowaniu znaków podkreślenia.

Inną popularną konwencją stosowaną do nazw zmiennych składających się z większej liczby słów jest rozpoczynanie każdego kolejnego wyrazu w nazwie wielką literą — na przykład `$ImieNazwisko` lub `$DataUrodzenia`. Jeśli ktoś preferuje taki styl, może z powodzeniem go stosować w swoich skryptach, musi jednak pamiętać, że wielkość liter ma znaczenie.

Tabela 2.1. Przykłady prawidłowych i nieprawidłowych nazw zmiennych

Prawidłowe nazwy zmiennych	Nieprawidłowe nazwy zmiennych
\$procent	\$pct%
\$imie_nazwisko	\$imie-nazwisko
\$wiersz_2	\$2gi_wiersz

Wyrażenia

Podczas przypisywania wartości do zmiennej podawana wartość nie musi być literalem. Równie dobrze może to być *wyrażenie* — połączone za pomocą *operatora* dwie (lub więcej) wartości, które wspólnie tworzą wynik. Zrozumienie działania zaprezentowanego tu przykładu nie powinno nastęrczyć trudności. Dokładny jego opis zamieszczono pod przykładem:

```
$sum = 16 + 30;
echo $sum;
```

Zmienna `$sum` pobiera wartość wyrażenia znajdującego się po prawej stronie znaku równości. Wartości `16` i `30` zostały połączone za pomocą operatora dodawania — symbolu plus (+), dlatego wyrażenie zwraca wynik dodawania tych dwóch liczb. Jak łatwo przewidzieć, wykonanie tego kodu spowoduje wyświetlenie liczby 46.

Tę samą operację dodawania można wykonać z wykorzystaniem dwóch zmiennych:

```
$a = 16;
$b = 30;
$sum = $a + $b;
echo $sum;
```

Wykonanie tego kodu powoduje dodanie wartości zmiennych `$a` i `$b`. Tak jak poprzednio, skrypt wyświetla liczbę 46.

Zmienne w ciągach znaków

Jak już powiedziano, ciągi znaków muszą być ujęte w cudzysłów lub apostrofy. Wiemy też, że istnieje różnica pomiędzy zastosowaniem apostrofów i cudzysłówów.

Na czym polega ta różnica? Znak dolara występujący w ciągu znaków ujętym w cudzysłów oznacza, że bieżąca wartość zmiennej powinna stać się częścią ciągu znaków. Jeśli natomiast występuje w ciągu znaków ujętym w apostrofy, znak dolara jest interpretowany literalnie i nie powoduje odwołania do zmiennych.

Różnicę tę zaprezentowano w zamieszczonych niżej przykładach. W pierwszym z nich w ciągu znaków wprowadzono wartość zmiennej `$name`:

```
$name = "Krzysztof";  
echo "Cześć, $name";
```

Wykonanie kodu spowoduje wyświetlenie ciągu `Cześć, Krzysztof`.

W drugim przykładzie znak dolara zostanie zinterpretowany literalnie i nie nastąpi odwołanie do zmiennej:

```
$name = 'Krzysztof';  
echo 'Cześć, $name';
```

Wykonanie tego kodu spowoduje wyświetlenie ciągu `Cześć, $name`.

Czasami w kodzie PHP trzeba jawnie oznaczyć początek i koniec zmiennej. W tym celu należy użyć nawiasów klamrowych — `{}`. Aby wyświetlić wagę z przyrostkiem oznaczającym kilogramy lub funty, odpowiednia instrukcja powinna przyjąć następującą postać:

```
echo "Całkowita waga wynosi {$weight}kg";
```

Gdyby nie zostały zastosowane nawiasy klamrowe wokół zmiennej `$weight`, interpreter PHP próbowałby znaleźć zmienną `$weightkg`, która najprawdopodobniej nie występuje w skrypcie.

Ten sam efekt można osiągnąć dzięki zastosowaniu operatora *konkatenacji* — symbolu kropki. Można go wykorzystać do połączenia ze sobą dwóch lub większej liczby ciągów znaków, tak jak pokazano w następującym przykładzie:

```
echo 'Całkowita waga wynosi ' . $weight . 'kg';
```

Trzy wartości — dwa literały znakowe i jedna zmienna `$weight` — są ze sobą łączone w kolejności, w jakiej występują w instrukcji. Warto zwrócić uwagę na to, że na końcu pierwszego ciągu znaków dodano spację, aby oddzielić słowo *wynosi* od wartości wagi.

Gdyby zmienna `$weight` miała wartość `99`, wykonanie przedstawionej wyżej instrukcji spowodowałoby wyświetlenie następującego wyniku:

```
Całkowita waga wynosi 99kg
```

Typy danych

Każda zmienna, w której zapisano wartość, posiada typ danych. Typ danych definiuje rodzaje wartości, które mogą być zapisane w zmiennej. Podstawowe typy danych dostępne w PHP zestawiono w tabeli 2.2.

Tabela 2.2. *Typy danych w PHP*

Typ danych	Opis
Boolean	Ocena logiczna; może mieć wartość TRUE (prawda) lub FALSE (fałsz).
Integer	Wartość numeryczna — dodatnia lub ujemna liczba całkowita.
Double (lub float)	Liczba zmiennoprzecinkowa — dowolna liczba dziesiętna.
String	Wartość alfanumeryczna — może zawierać dowolną liczbę znaków ASCII.

W momencie przypisania wartości do zmiennej następuje ustalenie typu danych. PHP określa typ danych automatycznie, na podstawie przypisanej wartości. Aby sprawdzić typ danych zmiennej, można skorzystać z funkcji `gettype`.

Wykonanie poniższego kodu pozwoli przekonać się, że typ danych liczby dziesiętnej to `double`:

```
$value = 7.2;
echo gettype($value);
```

Działanie odwrotne do `gettype` ma funkcja `settype`, która umożliwi przesłonięcie typu danych zmiennej. Jeśli zapisana w zmiennej wartość nie może być zapisana przy użyciu nowego typu, jest modyfikowana do najbliższej możliwej wartości.

Następujący kod jest próbą przekształcenia ciągu znaków na liczbę całkowitą:

```
$value = "22nd January 2005";
settype($value, "integer");
echo $value;
```

W tym przypadku ciąg znaków rozpoczyna się od cyfr, ale nie reprezentuje liczby całkowitej. W wyniku konwersji przekształcone będą wszystkie znaki — od początku ciągu do pierwszego znaku, który nie jest liczbą. Pozostała część ciągu znaków będzie odrzucona, a zatem wykonanie tego kodu zwróci wartość `22`.



Analiza typów danych. W praktyce funkcji `settype` i `gettype` nie wykorzystuje się zbyt często, ponieważ nie ma zbyt wielu sytuacji, w których trzeba modyfikować typ danych zmiennej. Jak już powiedziano, PHP automatycznie przypisuje typ danych do zmiennej.

Żonglowanie typami

Czasami interpreter PHP przeprowadza niejawną konwersję typów danych. Dzieje się tak w przypadku, gdy spodziewa się wartości określonego typu. Takie działanie określa się jako *żonglowanie typami* (ang. *type juggling*).

Na przykład operator dodawania powinien znajdować się pomiędzy dwoma liczbami. Przed wykonaniem operacji dodawania ciągi znaków są przekształcane na wartości typu `double` lub `integer`. Tak więc zaprezentowana poniżej operacja dodawania zwróci wynik w postaci liczby całkowitej:

```
echo 100 + "10 cali";
```

Wykonanie kodu powoduje dodanie liczb 100 i 10, dlatego w wyniku wyświetlona zostanie liczba 110.

Podobny mechanizm działa w przypadku zastosowania operatora znakowego dla danych liczbowych. W przypadku przeprowadzania operacji znakowej na danych liczbowych, wartości liczbowe są najpierw przekształcane na ciągi znaków. Z taką sytuacją zetknęliśmy się już przy okazji omawiania operatora konkatenaacji — wyświetlana zmienna `$weight` była liczbą.

Wynikiem działania operacji wykonywanych na ciągach znaków zawsze jest ciąg znaków, nawet wtedy, kiedy przypomina liczbę. Wykonanie kodu zaprezentowanego w zamieszczonym niżej przykładzie zwraca wynik 69, ale — jak pokazuje wynik działania funkcji `gettype` — zmienna `$number` zawiera ciąg znaków:

```
$number = 6.9;  
echo $number;  
echo gettype($number);
```

Rozbudowaną listę operatorów liczbowych i tekstowych przedstawimy w rozdziałach 5., „Przetwarzanie liczb”, oraz 6., „Przetwarzanie ciągów znaków”.

Zmienne do przechowywania nazw zmiennych

W PHP wartość zapisaną w zmiennej można wykorzystać jako nazwę innej zmiennej. Jeśli to nie brzmi zbyt zrozumiale, spróbujemy wyjaśnić to za pomocą następującego przykładu:

```
$my_age = 21;  
$varname = "my_age";  
echo "Wartość zmiennej $varname wynosi ${$varname}";
```

Wynik działania powyższego skryptu jest następujący:

```
Wartość zmiennej my_age wynosi 21
```

Ponieważ ciąg znaków jest ujęty w cudzysłów, znak dolara oznacza, że wartość zmiennej stanie się częścią ciągu znaków. Konstrukcja `${$varname}` spowoduje, że wartość zmiennej, której nazwę zapisano w zmiennej `$varname`, stanie się częścią ciągu znaków.

Nawiasy klamrowe wokół nazwy `$varname` są obowiązkowe wewnątrz ciągów znaków, ale w innych wywołaniach ich użycie nie jest konieczne. Kod zastosowany w naszym kolejnym, zamieszczonym poniżej przykładzie, w którym zastosowano operator konkatencji, zwraca dokładnie taki sam wynik jaki uzyskaliśmy poprzednio:

```
echo 'Wartość zmiennej ' . $varname . ' wynosi ' . $$varname;
```

Podsumowanie

W tym rozdziale mogliśmy zobaczyć, jak posługiwać się zmiennymi w PHP. W następnym dowiemy się, w jaki sposób korzystać z instrukcji warunkowych i instrukcji pętli do zarządzania przepływem sterowania w skryptach.