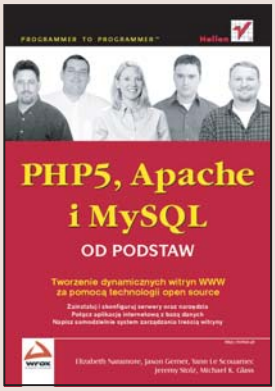
PHP5, Apache i MySQL. Od podstaw



Autor: Zespół autorów
Tłumaczenie: Rafał Jońca
ISBN: 83-7361-997-6
Tytuł oryginału: Beginning PHP5, Apache, MySQL Web Development
Format: B5, stron: 784

Tworzenie dynamicznych witryn WWW za pomocą technologii open source

- Zainstaluj i skonfiguruj serwery oraz narzędzia
- Połącz aplikację internetową z bazą danych
- Napisz samodzielnie system zarządzania treścią witryny

PHP, Apache i MySQL to najpopularniejsze obecnie technologie służące do tworzenia i udostępniania w sieci dynamicznych witryn WWW. Łączy je jedna wspólna cecha — wszystkie dostępne są na licencji open source, co oznacza, że korzystanie z nich nie wiąże się z żadnymi opłatami. PHP to język programowania służący do pisania aplikacji internetowych, MySQL to baza danych, która przechowuje informacje wyświetlane na stronach, a Apache to serwer WWW, dzięki któremu witryny te dostępne są w internecie. Za pomocą tych narzędzi tworzone są portale, sklepy internetowe, fora dyskusyjne i inne strony, z których codziennie korzystają dziesiątki tysięcy użytkowników sieci.

„PHP5, Apache i MySQL. Od podstaw" to podręcznik dla tych użytkowników, którzy chcą poznać sposoby tworzenia dynamicznych witryn WWW w oparciu o te technologie. Przedstawia zasady pisania aplikacji internetowych, stosowania w nich formularzy z mechanizmem weryfikacji poprawności wpisanych danych, łączenia aplikacji z bazą danych i modyfikowania elementów graficznych za pomocą PHP. Opisano tu także sposoby przechowywania danych użytkownika w plikach cookie, mechanizmy wysyłania poczty elektronicznej, obsługi błędów i ograniczania dostępu do określonych obszarów witryny za pomocą nazwy użytkownika i hasła. Każde zagadnienie zaprezentowano na rzeczywistych przykładach dynamicznych witryn WWW.

- Instalacja i konfiguracja Apache, PHP i MySQL-a
- Podstawowe elementy języka PHP
- Komunikacja z bazą danych
- Wyświetlanie oraz modyfikowanie danych
- Obsługa formularzy i weryfikacja poprawności wpisów
- Korzystanie z elementów graficznych i zastosowanie biblioteki GD
- Obsługa błędów
- Komunikacja za pomocą poczty elektronicznej
- Mechanizmy logowania i personalizacji serwisu
- System zarządzania treścią serwisu WWW
- Lista mailingowa i forum dyskusyjne
- Elementy sklepu internetowego
- Analiza ruchu w witrynie WWW za pomocą statystyk

Poznaj najnowsze technologie tworzenia dynamicznych witryn WWW

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

Spis treści

2

Tworzenie stron w PHP5

Niniejszy rozdział dotyczy podstaw języka PHP i jednocześnie jest początkiem opisu tworzenia pierwszej witryny internetowej związanej z recenzjami filmów. Po zakończeniu prac nad witryną jej użytkownicy będą mogli odnaleźć informacje na temat konkretnego filmu, a Czytelnik będzie potrafił pisać programy w języku PHP.

Nawet jeśli dobrze zna się wcześniejsze wersje PHP, polecamy przeczytanie tego rozdziału i zwrócenie szczególnej uwagi na podrozdział o programowaniu obiektowym, które stanowi nowy element, znacznie udoskonalony w PHP5.

Niniejszy rozdział omawia podstawowe polecenia i struktury języka PHP:

- wykorzystanie `echo` do wyświetlania tekstu,
- formatowanie tekstu przy użyciu PHP i HTML,
- stałe i zmienne,
- wykorzystanie adresów URL do przekazywania zmiennych,
- sesje i *cookies* (tzw. ciasteczka),
- formularze HTML,
- konstrukcje `if-else`,
- dołączenia innych plików,
- funkcje,
- tablice i konstrukcję `foreach`,
- pętle `while` i `do-while`,
- wykorzystanie klas i metod w programowaniu obiektowym.

Pod koniec rozdziału, po wykonaniu wszystkich ćwiczeń „Wypróbuj”, nikt nie powinien mieć najmniejszych problemów z utworzeniem prostego formularza uwierzytelniania, dania użytkownikom możliwości zobaczenia recenzji ulubionego filmu lub listy 10 najlepszych obrazów oraz zaoferowania ponumerowanej listy filmów bazującej na tym, ile będą oni chcieli zobaczyć. Po drobnych modyfikacjach będzie mogła to być alfabetyczna lista filmów.

Omówienie struktury i składni języka PHP

Programy PHP pisze się w edytorach tekstów, takich jak Notatnik, podobnie jak strony HTML. Niemniej, w odróżnieniu od tradycyjnych stron WWW, strony PHP posiadają rozszerzenie *.php*. Rozszerzenie to informuje serwer WWW, aby dokonał przetworzenia kodu PHP przed wysłaniem wynikowej strony do przeglądarki internetowej użytkownika.

W pięciogwiazdkowej restauracji klienci widzą jedynie talerze pełne wspaniałych potraw, które zostały dla nich przygotowane. Nie wiedzą, skąd pochodzi jedzenie ani jak zostało przygotowane. W podobny sposób PHP przygotowuje kod HTML — użytkownik nie widzi, jak jest on przygotowywany.

Jak kod PHP umieszcza się w kodzie HTML?

Zakładamy, że Czytelnik zna choć trochę język HTML i zapewne nieraz widział, w jaki sposób wstawiany jest kod języka JavaScript lub innego wewnątrz kodu strony WWW. PHP jest inny — nie tylko umożliwia tworzenie stron internetowych „w locie", ale jest również niewidoczny dla odwiedzających. Użytkownicy widzą jedynie wynik wykonania napisanego skryptu, czyli kod HTML. Zapewnia to większe bezpieczeństwo kodu PHP i większą elastyczność przy jego pisaniu.

Nic nie stoi na przeszkodzie, aby kod HTML pojawił się w sekcji kodu PHP strony — ułatwia to formatowanie tekstu przy jednoczesnym zachowaniu zwartej budowy bloków kodu. Programista pisze w ten sposób wydajniejszy i lepiej zorganizowany kod, a przeglądarka internetowa (i, co najważniejsze, użytkownik) nie zauważa różnicy.

PHP może zostać napisany jako samowystarczalny program bez żadnych wstawek HTML. Takie pliki stosuje się na ogół do przechowywania zmiennych połączeń z bazą danych, przekierowania użytkownika na inną stronę lub wykonania innych funkcji omówionych w książce.

Zasady składni języka PHP

Jedną z podstawowych zalet PHP jest jego prostota. Podobnie jak w każdym języku programowania istnieje kilka sposobów na wykonanie tego samego zadania. Po dobrym zaznajomieniu się z podstawami i niektórymi programami, warto rozpocząć poszukiwania skrótów, które uczynią kod wydajniejszym. Aby nie utrudniać początku znajomości, omówimy jedynie najpopularniejsze techniki, zasady i funkcje języka PHP.

Zawsze należy pamiętać o dwóch podstawowych zasadach PHP.

- Kod PHP umieszcza się na stronie pomiędzy odpowiednimi znacznikami przedstawionymi poniżej:

```
<?php
?>
```

■ Ogólnie rzecz biorąc, wiersze kodu PHP kończą się znakiem średnika:

```
<?php
// Pierwszy wiersz kodu;
// Drugi wiersz kodu;
// Trzeci wiersz kodu;
?>
```

Komentarze w kodzie programu umieszcza się, poprzedzając je dwoma ukośnikami // (dotyczy to komentarzy jednowierszowych) lub umieszczając między konstrukcjami /* i */ (komentarze wielowierszowe). Wcięcia nie mają żadnego znaczenia, podobnie jak przejścia do nowych wierszy. Daje to programiście ogromną swobodę, która, niestety, czasem jest nadużywana, co wkrótce przedstawimy na przykładzie.

Zupełne podstawy mamy już za sobą. Reszta to w zasadzie tylko doskonalenie swoich zdolności programistycznych.

Techniki kodowania są niezmiernie ważne

Zanim przejdziemy dalej, warto wskazać, w jaki sposób struktura kodu potrafi wpłynąć na skrypt. Dla serwera WWW zajmującego się przetwarzaniem kodu PHP nie ma ona większego znaczenia — serwer widzi cały kod jako jeden bardzo długi wiersz pozbawiony wszystkich znaków tabulacji, wcięć i przejść do nowych wierszy. Dla człowieka stosowanie odpowiednich wcięć i organizacji kodu ma przeogromne znaczenie.

Przyjrzyjmy się przykładom:

Przykład 1.

```
<?php
if ($_POST["fname"] == "Jan") {
  echo "<p>Witaj $_POST['fname']</p>;
}
else {
  echo "<h2>Nie masz na imię Jan, więc nie możesz wejść na tę witrynę.</h2>"
}
?>
```

Przykład 2.

```
<?php
// sprawdź, czy użytkownik ma na imię Jan, zanim wyświetlisz zawartość witryny
        if ($_POST["fname"] == "Jan")
            {
            echo "<p>";
            echo "Witaj ";
            echo $_POST['fname'];
            echo "</p>";
            }
        else
            {
            echo "<h2>";
            echo "Nie masz na imię Jan, więc nie możesz wejść na tę witrynę.";
            echo "</h2>";
            }
?>
```

Chociaż drugi przykład wymaga więcej pisania, łatwiej w nim zlokalizować błędy w składni lub wyłączyć dany fragment kodu w trakcie przeprowadzania testów. Jest to szczególnie ważne, gdy dopiero zaczyna się przygodę z językiem. Po nabraniu wprawy można pisać bardziej zwięzły kod z przykładu 1.

Kiedy program jest dobry?

Naprawdę profesjonalny kod powinien wzorować się na wykorzystaniu trzech wskazówek.

- **Spójność**. Bloki dobrze napisanego kodu zawsze wyglądają tak samo i mają te same wcięcia, skróty itp. Znacznie ułatwia to późniejsze czytanie kodu. Wspaniałą cechą języka PHP jest to, że nie troszczy się o wcięcia i znaki tabulacji, więc można stosować własny styl, o ile jest on jednolity i wygodny.

  Ponieważ często istnieje więcej niż jedna składnia związana z wybranym zagadnieniem, dobrzy programiści wybierają jedną składnię i stosują się do niej przez cały czas. Z punktu widzenia interpretera kodu PHP dwa poniższe fragmenty są sobie równoważne:

```
<?php
// tutaj znajduje się kod PHP;
?>

<?
// tutaj znajduje się kod PHP;
?>
```

  Najlepiej wybrać jedno z rozwiązań i trzymać się go cały czas przy pisaniu kodu.

- **Częste komentarze**. Im więcej komentarzy zawiera kod, tym lepiej. Choć nie jest to aż tak istotne w krótkich, niewielkich programach, wraz ze wzrostem ich złożoności coraz trudniej zapamiętać, co zostało zrobione, dlaczego zostało napisane i dlaczego dane rozwiązanie jest najlepsze. Szczegółowe komentarze pozwalają przypomnieć sobie dawne motywy. Gdy nad projektem pracuje wielu programistów, komentarze znacząco ułatwiają zrozumienie kodu napisanego przez inną osobę.

- Numerowanie wierszy**. Niektóre edytory tekstu numerują wiersze, ale niestety nie wszystkie**. Edytory tekstu omawiamy dokładniej w dalszej części programu. Jeśli edytor nie zapewnia numerowania wierszy, warto zagwarantować to sobie samemu, ponieważ gdy skrypt zawiera błędy, interpreter wyświetla informację, w którym wierszu wykrył nieprawidłowość. Jeśli trzeba będzie ręcznie liczyć wiersze po napotkaniu każdego błędu, testowanie kodu i usuwanie z niego błędów będzie zajmowało niezmiernie dużo czasu.

Dlaczego warto troszczyć się o to, jak wygląda kod?

Z trzech powodów warto stosować dobre praktyki pisania kodu:

- **Wydajność**. Im mniej problemów sprawia przeczytanie kodu, tym łatwiej śledzić, co się w nim dzieje, i w razie potrzeby dokonywać odpowiednich poprawek (nawet po dłuższej przerwie).

- **Usuwanie błędów**. Wiedza na temat tego, co jest powodem błędu, jest bezcenna. Gdy stosuje się komentarze, łatwiej zrozumieć logikę programu. Jeżeli dodatkowo stosuje się numerację wierszy i jednolite formatowanie kodu, odnalezienie problematycznego fragmentu zajmuje jedynie chwilę.
- **Przyszłe rozszerzenia i modyfikacje**. Stosowanie komentarzy jest niezmiernie ważne, gdy chce się w przyszłości rozbudowywać kod, gdyż niezmiernie trudno przypomnieć sobie logikę skryptu pisanego wiele miesięcy temu. W trakcie pracy w zespole programistycznym komentarze i jednolitość formatowania ułatwiają dokonywanie zmian w kodzie napisanym przez inne osoby.

Na tym zakończymy wstęp do programowania — przejdźmy do praktyki.

Pierwszy program

Chyba nie można napisać prostszego programu od poniższego, ale jego prześledzenie pozwoli zrozumieć wyniki działania skryptów PHP. Funkcja `echo`, którą można zauważyć w przedstawionym poniżej skrypcie, jest jedną z najczęściej stosowanych funkcji języka PHP. Służy do wysyłania tekstu (zawartości zmiennych i nie tylko) do przeglądarki internetowej.

Wypróbuj — funkcja echo

Wypróbuj funkcję `echo`, aby sprawdzić, jakie będą wyniki.

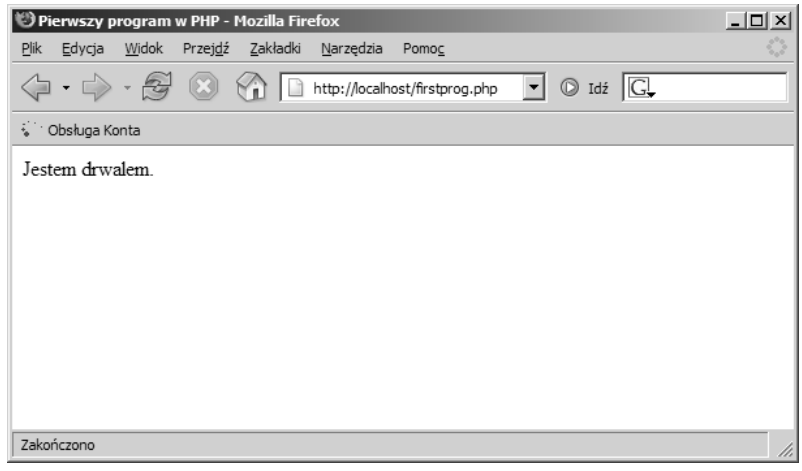
1. Wpisz poniższy program w ulubionym edytorze tekstu (na przykład Notatniku) i zapisz w pliku *firstprog.php*.

   Upewnij się, iż został on zapisany jako „zwykły tekst", aby uniknąć problemów z interpretacją. Jeśli korzysta się z Notatnika, warto sprawdzić, czy plik nie został zapisany pod nazwą *firstprog.php.txt*.

```
<html>
<head>
<title>Pierwszy program w PHP</title>
</head>
<body>
<?php
echo "Jestem drwalem.";
?>
</body>
</html>
```

2. Otwórz program, używając przeglądarki internetowej. Uzyskany wynik powinien wyglądać tak jak na rysunku 2.1.
3. Wyświetl źródło kodu HTML, aby przekonać się, co się tak naprawdę stało ze skryptem PHP. Jak nietrudno zauważyć, kod PHP zniknął i pozostał jedynie kod HTML.
4. Dodaj wyróżniony wiersz z poniższego kodu, aby lepiej zrozumieć istotę działania skryptów PHP.

Rysunek 2.1.




```
<html>
<head>
<title>Pierwszy program w PHP</title>
</head>
<body>
<?php
echo "Jestem drwalem.";
echo "I jest mi dobrze.";
?>
</body>
</html>
```

5. Zapisz zmodyfikowany plik i otwórz go ponownie w przeglądarce. Oba wiersze zostały złączone razem bez żadnych przerw, choć w kodzie PHP znajdowały się w dwóch osobnych wierszach.

Jak to działa?

Gdy przeglądarka internetowa wywołuje program PHP, najpierw przeszukuje się cały kod wiersz po wierszu w poszukiwaniu sekcji PHP (kodu PHP umieszczonego w znacznikach), a następnie przetwarza się je po kolei. Serwer traktuje cały kod PHP jak jeden długi wiersz, więc dwa powyższe wiersze zostały złączone na ekranie. Po przetworzeniu kodu PHP serwer wraca do pozostałego jeszcze kodu i wysyła go do przeglądarki wraz z kodem sekcji PHP.

Wykorzystanie HTML-a do uatrakcyjnienia stron

Jak łatwo zauważyć, wykorzystanie jedynie samego kodu PHP spowodowałoby powstanie bardzo nudnych stron WWW. Wystarczy jednak zastosować trochę kodu HTML, aby stały się one znacznie ciekawsze. Kod HTML umieszcza się w bloku PHP, używając funkcji `echo`. Skrypt PHP może korzystać z wszystkich elementów HTML, włączając w to ramki, tabele, różne czcionki itp.

Integracja HTML-a z PHP

Kolejny praktyczny przykład przedstawia, jak łatwo zastosować kod HTML w programie PHP.

Wypróbuj — użycie kodu HTML w skrypcie PHP

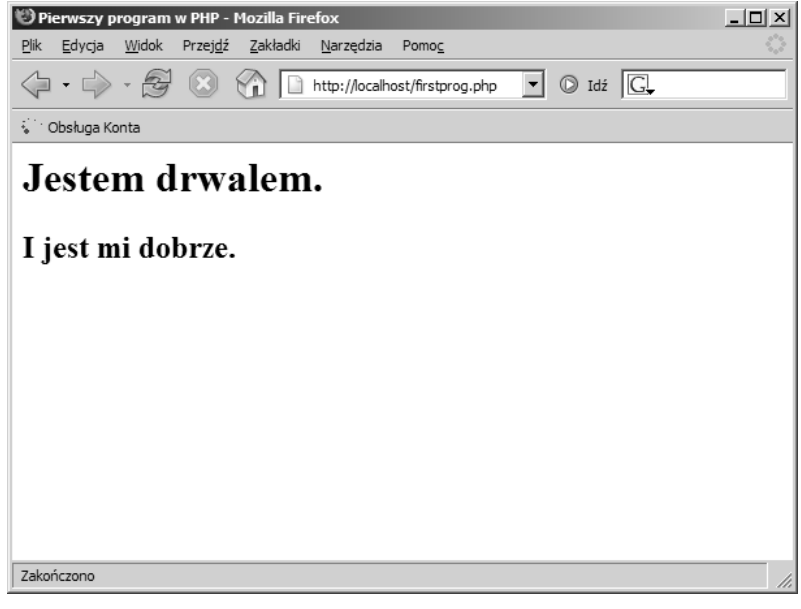
W niniejszym przykładzie łączymy kod PHP i HTML.

1. Zmodyfikuj wyróżnione wiersze programu *firstprog.php*.

```
<html>
<head>
<title>Pierwszy program w PHP</title>
</head>
<body>
<?php
echo "<h1>Jestem drwalem.</h1>";
echo "<h2>I jest mi dobrze.</h2>";
?>
</body>
</html>
```

2. Zapisz plik i przeładuj stronę WWW. Uzyskany wynik powinien wyglądać tak jak na rysunku 2.2.

Rysunek 2.2.



Jak to działa?

Funkcja `echo` po prostu wysyła do przeglądarki to, co zostało jej przekazane, niezależnie od tego, czy jest to zwykły tekst, kod HTML, zmienna itp. Aby to udowodnić, w przedstawionym przykładzie po prostu przekazaliśmy kod HTML, modyfikując następujące wiersze:

```
echo "<h1>Jestem drwalem.</h1>";
echo "<h2>I jest mi dobrze.</h2>";
```

Wstawiając kod HTML do sekcji PHP, uzyskujemy dwie rzeczy:

- Poprawiamy wygląd strony WWW.
- Utrzymujemy skrypt PHP zwartym, gdyż nie przeskakujemy cały czas między kodem PHP i HTML.

Jeśli zajrzy się do kodu źródłowego strony WWW, da się zauważyć, że kod HTML został wstawiony zgodnie z oczekiwaniami.

Rozważania na temat kodu HTML w sekcjach PHP

Poniżej przedstawiamy kilka wskazówek związanych z typowymi błędami wstawiania kodu HTML w sekcjach PHP.

- **Sprawdzaj cudzysłowy**. Łatwo wywnioskować z poprzedniego przykładu, iż funkcja echo wymaga zastosowania cudzysłowów. Ponieważ kod HTML także korzysta z cudzysłowów, warto zastosować jedną z dwóch sztuczek, by uniknąć problemów:
  - Zastosować apostrofy w kodzie HTML.
  - Zastosować znak unikowy (lewy ukośnik) przed każdym cudzysłowem kodu HTML. Oto przykład:

    ```
    echo "<font size=\"2\">";
    ```

    Jest to szczególnie przydatne, gdy chce się umieścić cudzysłowy w tekście.

    ```
    echo "Marek powiedział \"Tak\".";
    ```

- **Pamiętaj, że zasady kodu PHP obowiązują także wtedy, gdy stosuje się kod HTML**. Czasami zapomina się, pisząc dłuższy fragment kodu HTML w sekcji PHP, że na końcu instrukcji echo trzeba umieszczać cudzysłowy zamykające i średniki.
- **Nie umieszczaj zbyt dużo kodu HTML w sekcjach PHP**. Jeśli jest środek sekcji PHP, a kod HTML staje się coraz dłuższy, warto pomyśleć o zakończeniu takiej sekcji i rozpoczęciu pisania wyłącznie w kodzie HTML. Rozważmy dwa poniższe przykłady:

  Przykład 1.

  ```
  <?php
    echo "<table width='100%' border='2' bgcolor='#FFFFFF'>";
    echo "<tr>";
    echo "<td width='50%'>";
    echo "<font face='Verdana, Arial' size='2'>";
    echo "Imię:";
    echo "</font></td>";
    echo "<td width='50%'>";
    echo "<font face='Verdana, Arial' size='2'>";
    echo $_POST['fname'];
    echo "</font></td>";
    echo "</tr>";
    echo "</table>";?>
  ```

Przykład 2.

```
<table width='100%' border='2' bgcolor='#FFFFFF'>
  <tr>
    <td width='50%'>
      <font face='Verdana, Arial' size='2'>
        Imię:
      </font>
    </td>
    <td width='50%'>
      <font face='Verdana, Arial' size='2'>
        <?php
          echo $_POST['fname'];
        ?>
      </font>
    </td>
  </tr>
</table>
```

Chociaż nie omówiliśmy jeszcze zmiennych, nietrudno zauważyć, iż tak naprawdę w całym fragmencie jedynym dynamicznie wstawianym elementem była zmienna `fname`. Pozostała treść to po prostu kod HTML. Z tego względu lepiej pozostać w języku HTML i wykorzystać PHP jedynie wtedy, gdy jest naprawdę potrzebny, niż stosować kod HTML w PHP. Choć nie ma to dużego znaczenia dla serwera, upraszcza formatowanie, ułatwia testowanie i nie wymaga tyle pisania (co zawsze jest dobrą rzeczą). Zawsze trzeba umieć zachować równowagę między kodem HTML i PHP w trakcie pisania dowolnej witryny internetowej.

Wykorzystanie zmiennych i stałych

Omówiliśmy podstawy wykorzystywania funkcji echo do wyświetlania dowolnego tekstu. Na razie nie różni się to niczym od pisania standardowej strony WWW w HTML-u. Dopiero zastosowanie stałych i zmiennych pozwala językowi PHP pokazać całą swoją moc.

Stałe

Stała to symbol zastępczy dla wartości, z której korzysta się w kodzie. Na ogół stałe pisze się wielkimi literami (aby łatwiej odnaleźć je w kodzie), a ich wartości definiuje się przed użyciem symbolu zastępczego. Nazwy stałych muszą zaczynać się od litery lub podkreślenia i nie mogą zaczynać się od cyfry. Interpreter skryptów zwraca uwagę na wielkość liter.

Stałe definiuje się za pomocą funkcji `define()`. Po zdefiniowaniu stałej nie można jej zmienić ani usunąć.

Wypróbuj — wykorzystanie stałych

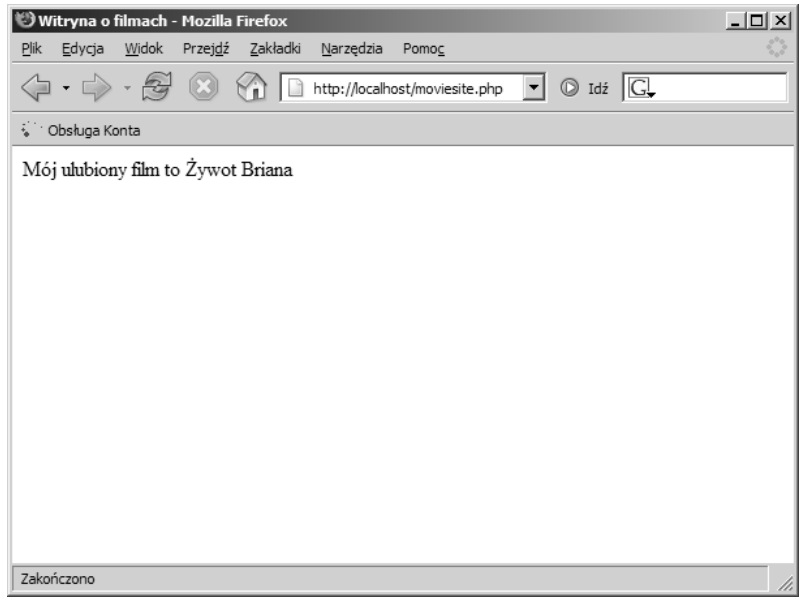
W ćwiczeniu przedstawimy wykorzystanie stałych w programie PHP.

1. Otwórz edytor tekstu i wpisz poniższy program:

```
<html>
<head>
<title>Witryna o filmach</title>
</head>
<body>
<?php
  define ("FAVMOVIE", "Żywot Briana");
  echo "Mój ulubiony film to ";
  echo FAVMOVIE;
?>
</body>
</html>
```

2. Zapisz tekst w pliku *moviesite.php* i otwórz go, używając przeglądarki internetowej. Strona powinna wyglądać tak jak na rysunku 2.3.

Rysunek 2.3.



Jak to działa?

Definiując stałą o nazwie `FAVMOVIE`, przypisaliśmy jej wartość `Żywot Briana`, która może zostać użyta i wyświetlona w przyszłości. Choć stałej nie można zmienić ani usunąć, jest ona dostępna w dowolnym miejscu skryptu.

Zmienne

W odróżnieniu od stałych, zmienne, jak sama nazwa wskazuje, potrafią się zmieniać — różne fragmenty programu mogą modyfikować ich zawartość. Zmienne nie muszą być definiowane lub deklarowane. Wystarczy coś do nich przypisać, gdy jest to konieczne. Zmienne mogą przechowywać dane tekstowe lub numeryczne.

Zmienne poprzedza się znakiem dolara ($), a wielkość liter ma znaczenie (innymi słowy, `$WpisanaData` i `$wpisanaData` to dwie różne zmienne). Pierwszy znak zmiennej musi być znakiem podkreślenia lub literą i nie może być cyfrą.

> W poprzedniej wersji języka PHP (4.) zmienne nie były przekazywane przez referencję, jeśli nie poprzedziło się ich znakiem ampersanda (&). W PHP5 wszystkie zmienne są przekazywane przez referencję i nie jest do tego wymagana żadna dodatkowa składnia. W znaczący sposób poprawia to szybkość i elastyczność programów. Nie należy się przejmować, jeśli to, co tu napisaliśmy, nie jest dla kogoś jasne. Staje się to istotne dopiero w bardziej złożonych aplikacjach.

Wypróbuj — wykorzystanie zmiennych

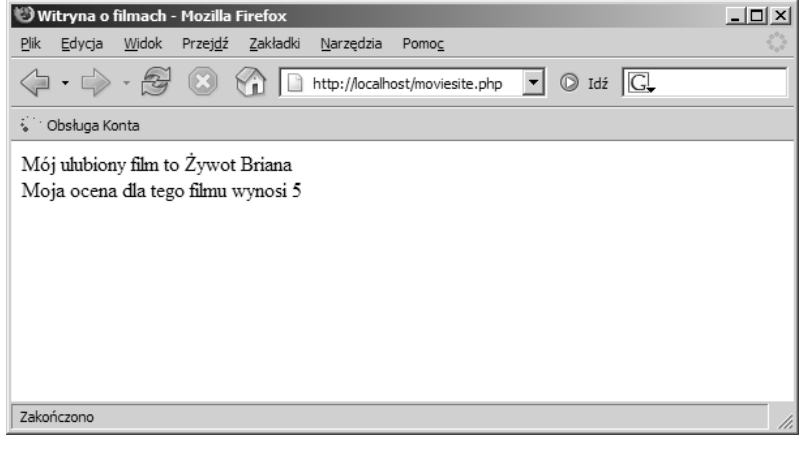
W ćwiczeniu do istniejącego już skryptu dodamy zmienne.

1. Otwórz edytor tekstu i dokonaj w pliku *moviesite.php* odpowiednich zmian (zostały one wyróżnione).

```
<html>
<head>
<title>Witryna o filmach</title>
</head>
<body>
<?php
  define ("FAVMOVIE", "Żywot Briana");
  echo "Mój ulubiony film to ";
  echo FAVMOVIE;
  echo "<br>";
  $movierate = 5;
  echo "Moja ocena dla tego filmu wynosi ";
  echo $movierate;
?>
</body>
</html>
```

2. Zapisz zmiany i otwórz plik w przeglądarce internetowej. Powinna pojawić się strona przedstawiona na rysunku 2.4.

Rysunek 2.4.



Jak to działa?

Wartość „5” jest przypisywana do zmiennej `movierate`. Warto zauważyć, że nie jest to zmienna tekstowa, ale liczbowa. Poniższy wiersz spowodował przypisanie zmiennej jako ciągu znaków.

```
$movierate = "5";
```

Zachowując zmienną jako liczbę, umożliwiamy przeprowadzanie na niej w przyszłości operacji matematycznych (na przykład wyliczenie średniej oceny filmu). Oto przykład:

```
<?php
  $bobsmovierate = 5;
  $joesmovierate = 7;
  $grahamsmovierate = 2;
  $zabbysmovierate = 1;
  $avgmovierate = (($bobsmovierate + $joesmovierate + $grahamsmovierate
                    + $zabbysmovierate) / 4);
  echo "Średnia ocena dla tego filmu wynosi ";
  echo $avgmovierate;
?>
```

Język PHP posiada wiele wbudowanych funkcji matematycznych, z których można korzystać dla zmiennych zawierających liczby. Oto kilka z nich:

- Funkcja `rand([min], [maks])` — generuje losową liczbę całkowitą z podanego przedziału.
- Funkcja `ceil(liczba)` — zaokrągla liczbę w górę do liczby całkowitej.
- Funkcja `floor(liczba)` — zaokrągla liczbę w dół do liczby całkowitej.
- Funkcja `number_format(liczba [, po przecinku] [, znak przecinka] [, znak tysięcy])` — formatuje liczbę zgodnie z podaną liczbą miejsc po przecinku, używając przekazanych znaków jako przecinka i separatora tysięcy. Domyślnie PHP używa kropki jako separatora części całkowitej i ułamkowej oraz przecinka jako separatora kolejnych tysięcy. Jeśli takie formatowanie jest odpowiednie, można pominąć opcjonalne parametry umieszczone w nawiasach kwadratowych. Aby otrzymać formatowanie stosowane w Polsce, trzeba napisać następujący kod:

  ```
  $price = 12345.67;
  number_format($price); // zwraca 12,345.67
  number_format($price, 2, ",", " "); // zwraca 12 345,67
  ```

- Funkcja `max(argument1, argument2, ...)` — zwraca maksymalną wartość z przekazanych argumentów.
- Funkcja `min(argument1, argument2, ...)` — zwraca minimalną wartość z przekazanych argumentów.

Pełna lista funkcji matematycznych języka PHP znajduje się w dodatku C.