

SYSTEMY ZARZĄDZANIA TREŚCIĄ CMS

LABORATORIUM 1

Zastosowanie PHP

Praca nad przykładami przedstawionymi w tym rozdziale oraz w pozostałej części książki wymaga dostępu do serwera WWW z zainstalowanym PHP. Aby maksymalnie wykorzystać przykłady i opisy konkretnych przypadków, należy je uruchomić i podjąć próbę modyfikacji. Żeby to zrobić, potrzebne jest miejsce do przeprowadzania testów.

Tworzenie przykładowej aplikacji: „Części samochodowe Janka”

Do najbardziej popularnych zadań każdego obsługiwanego przez serwer języka skryptowego należy przetwarzanie formularzy HTML. Poznawanie PHP rozpoczynamy od implementacji formularza dla „Części samochodowych Janka”, fikcyjnej firmy zajmującej się sprzedażą części.

Formularz zamówienia

Programista HTML pracujący dla firmy Janka stworzył formularz zamówienia na części, które Janek sprzedaje. Ten stosunkowo prosty formularz widoczny na rysunku 1.1 jest podobny do wielu innych istniejących w sieci WWW. Na początku Janek chciałby wiedzieć, co zamówił jego klient, obliczyć sumę tego zamówienia oraz ustalić kwotę podatku VAT od tego zamówienia.

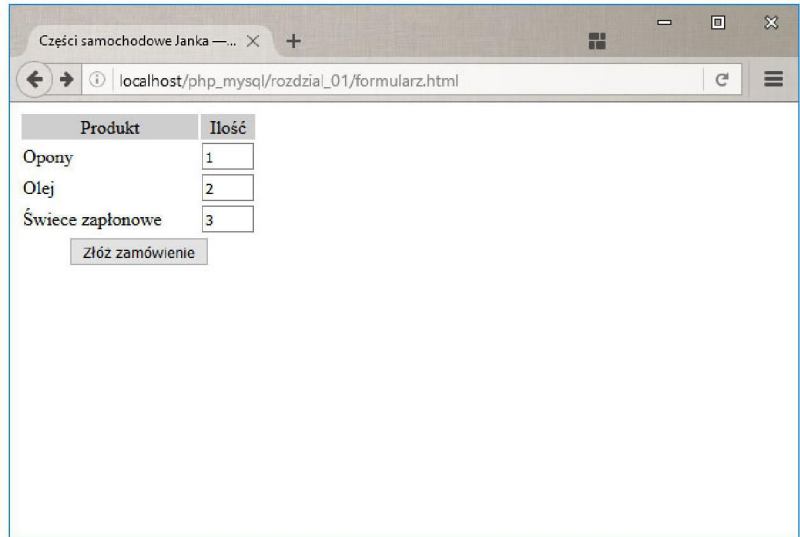
Część kodu HTML dla tego formularza jest przedstawiona na listingu 1.1.

Listing 1.1. formularz.html — kod HTML dla podstawowego formularza zamówienia Janka

```
<form action="przetworzamowienie.php" method="post">
<table style="border: 0px;">
<tr style="background: #cccccc;">
  <td style="width: 150px; text-align: center;">Produkt</td>
  <td style="width: 15px; text-align: center;">Ilość</td>
</tr>
<tr>
  <td>0pony</td>
  <td><input type="text" name="iloscoxon" size="3" maxlength="3" /></td>
</tr>
```

Rysunek 1.1.

Na początkowym formularzu zamówienia Janka zapisano jedynie produkty i ich ilość



Produkt	Ilość
Opony	1
Olej	2
Świece zapłonowe	3

Złóż zamówienie

```
<tr>
  <td>Olej</td>
  <td><input type="text" name="iloscolegu" size="3" maxlength="3" /></td>
</tr>
<tr>
  <td>Świece Zapłonowe</td>
  <td><input type="text" name="iloscswiec" size="3" maxlength="3" /></td>
</tr>
<tr>
  <td colspan="2" style="align: center"><input type="submit" value="Złóż zamówienie" /></td>
</tr>
</table>
</form>
```

Należy zwrócić uwagę, że akcja formularza została ustawiona na nazwę skryptu PHP, który będzie przetwarzał zamówienie klienta (zostanie on napisany później). Ogólnie rzecz biorąc, wartość atrybutu ACTION to URL, który zostanie załadowany, kiedy użytkownik naciśnie przycisk *Złóż zamówienie*. Dane, które wprowadził użytkownik do formularza, zostaną przesłane do tego URL-a za pomocą metody protokołu HTTP podanej w atrybucie method — czy będzie to get (dodane na końcu URL-a), czy też post (wysłane jako osobny komunikat).

Drugą kwestią wartą uwagi są nazwy pól formularza: `iloscopon`, `iloscolegu`, `iloscswiec`, użyte ponownie w skrypcie PHP. Z tego powodu ważne jest nadawanie polom formularza nazw, które coś znaczą i mogą być łatwo zapamiętane po rozpoczęciu pisania skryptu PHP. Niektóre edytory HTML domyślnie tworzą nazwy pól, takie jak `pole23`, które niełatwo zapamiętać. Zadanie programisty PHP jest o wiele łatwiejsze, jeżeli nazwy opisują wpisywane w te pola dane.

Możliwe jest takie zaadaptowanie standardu kodowania nazw pól, aby wszystkie nazwy na stronach witryny używały identycznego formatu. Ułatwia to zapamiętanie na przykład skrótów wyrazów w nazwie pola formularza lub stosowanie znaku podkreślenia jako spacji.

Przetwarzanie formularza

Aby przetworzyć formularz, trzeba utworzyć skrypt wymieniony w atrybucie ACTION znacznika form, o nazwie *przetworzzamowienie.php*. Należy otworzyć edytor tekstów, utworzyć ten plik i wpisać następujący kod:

```
<!DOCTYPE html>
<html>
<head>
  <title>Części samochodowe Janka – wyniki zamówienia</title>
</head>
<body>
<h1>Części samochodowe Janka</h1>
<h2>Wyniki zamówienia</h2>
</body>
</html>
```

Warto podkreślić, że wszystko, co zostało napisane do tej pory, to czysty kod HTML. Teraz nadszedł czas, aby dodać prosty kod PHP do skryptu.

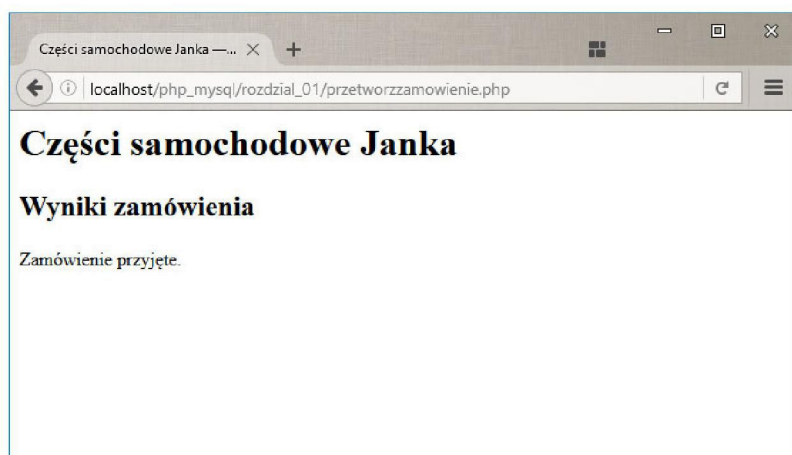
Osadzanie PHP w HTML

Pod nagłówkiem <h2> w powyższym pliku należy dodać następujące wiersze:

```
<?php
  echo '<p>Zamówienie przyjęte.</p>';
?>
```

Następnie trzeba zapisać plik, załadować go w przeglądarce poprzez wypełnienie formularza Janka i kliknąć przycisk *Złóż zamówienie*. Rezultat powinien być zbliżony do wyniku pokazanego na rysunku 1.2.

Rysunek 1.2.
Tekst przekazany
w konstrukcji PHP
echo został pokazany
przez przeglądarkę



Warto zauważyć, że kod PHP został osadzony w zwyczajnie wyglądającym pliku HTML. W dalszej kolejności należy obejrzeć źródło w przeglądarce. Powinien pojawić się następujący kod:

```
<!DOCTYPE html>
<html>
<head>
  <title>Części samochodowe Janka – wyniki zamówienia</title>
```

```
</head>
<body>
<h1>Części samochodowe Janka</h1>
<h2>Wyniki zamówienia</h2>
<p>Zamówienie przyjęte.</p></body>
</html>
```

Żaden fragment czystego kodu PHP nie jest widoczny, ponieważ interpreter języka PHP przetworzył skrypt i zamienił go na jego wyniki. Oznacza to, że z PHP można utworzyć czysty kod HTML, możliwy do przeglądania w każdej przeglądarce — innymi słowy, przeglądarka użytkownika nie musi rozumieć języka PHP.

Powyższy przykład ukazuje ideę przetwarzania skryptów przez serwer. Kod PHP jest interpretowany i wykonywany przez serwer WWW, natomiast JavaScriptu i innych technologii — przez przeglądarkę WWW na komputerze użytkownika.

Kod znajdujący się w tym pliku składa się z czterech części tekstowych:

- kodu HTML,
- znaczników PHP,
- instrukcji PHP,
- odstępów.

Do kodu można również dodać komentarze.

Większa część wierszy kodu przedstawionych w przykładzie to czysty kod HTML.

Zastosowanie znaczników PHP

Kod PHP w powyższym przykładzie rozpoczynał się znakiem `<?` i kończył `?>`. Jest to konstrukcja podobna do wszystkich znaczników HTML, które rozpoczynają się symbolem „mniejsze niż” (`<`), a kończą symbolem „większe niż” (`>`). Symbole te (`<?php` oraz `?>`), są nazywane *znacznikami PHP* i komunikują serwerowi WWW, gdzie rozpoczyna się i kończy kod PHP. Każdy tekst pomiędzy znacznikami zostanie zinterpretowany jako PHP, a poza nimi — jako normalny kod HTML. Znaczniki PHP pozwalają na *ucieczkę* od HTML.

Można wybierać różne style znaczników. Przyjrzyjmy się im bliżej.

Istnieją dwa różne style znaczników PHP. Wszystkie poniższe fragmenty kodu są równoznaczne.

■ Styl XML

```
<?php echo '<p>Zamówienie przyjęte.</p>'; ?>
```

Takiego stylu znaczników będziemy używać w książce, gdyż jest to preferowany styl znaczników PHP. Administrator serwera nie ma możliwości jego wyłączenia, a więc mamy gwarancję, że będzie on dostępny na każdym serwerze. Nabiera to szczególnego znaczenia w przypadku aplikacji, które prawdopodobnie będą używane w różnych środowiskach. Ten styl znaczników może być używany z dokumentami XML (*Extensible Markup Language* — Rozszerzalny Język Oznaczania). Ogólnie rzecz biorąc, zaleca się stosowanie tego właśnie stylu znaczników PHP.

■ Styl krótki

```
<? echo '<p>Zamówienie przyjęte.</p>'; ?>
```

Ten najprostszy styl znaczników jest skonstruowany według standardów przetwarzania instrukcji SGML (*Standard Generalized Markup Language* — Standardowy Ogólny Język Oznaczania). Aby posługiwać się tym najkrótszym typem znaczników, należy skompilować PHP z włączonymi krótkimi znacznikami bądź włączyć je w pliku konfiguracyjnym. Szczegółowe informacje na ten temat znajdują się w dodatku A. Używanie takiego stylu znaczników nie jest zalecane, ponieważ nie będzie ono rozpoznawane w wielu środowiskach, gdyż jego obsługa jest domyślnie wyłączona.

Instrukcje PHP

Interpreter PHP działa dzięki instrukcjom umieszczonym między znacznikami otwierającymi i zamykającymi. W poprzednim przykładzie wystąpił tylko jeden typ instrukcji:

```
echo '<p>Zamówienie przyjęte.</p>';
```

Jak łatwo się domyślić, zastosowanie konstrukcji `echo` daje bardzo prosty rezultat: wyświetlenie łańcucha znaków przekazanych do przeglądarki. Na rysunku 1.2 przedstawiono wynik — w oknie pojawia się tekst *Zamówienie przyjęte*.

Należy zauważyć średnik pojawiający się na końcu instrukcji `echo`. Jego funkcją jest odseparowanie instrukcji PHP, podobnie jak w języku polskim rozdziela się zdania kropką. Średnik spełnia podobne funkcje w językach Java i C.

Do często popełnianych błędów składni należy opuszczenie średnika. Na szczęście, równie łatwo go odnaleźć i naprawić.

Odstępy

Znaki tworzące przerwy — takie jak nowe wiersze (powrót karetki), spacje i znaki tabulacji — noszą nazwę *odstępów*. Z połączenia akapitów powyższego i poniższego wynika jeden spójny akapit wyjaśniający, w jaki sposób odstępy są ignorowane w PHP i HTML.

Jak wiadomo, przeglądarki ignorują odstępy w HTML-u. Podobnie działa mechanizm PHP. Warto przeanalizować dwa poniższe fragmenty kodu HTML:

```
<h1>Witamy w "Częściach samochodowych Janka"!</h1><p>Co Państwo zechcą zamówić dzisiaj?</p>
```

i

```
<h1>Witamy w          "Częściach samochodowych  
Janka"!</h1>  
<p>Co Państwo zechcą  
zamówić dzisiaj?</p>
```

Przedstawione fragmenty kodu HTML dają identyczne rezultaty, ponieważ są tak samo interpretowane przez przeglądarkę. Można jednak — co jest zalecane — używać odstępów w HTML-u jako pomocy w celu poprawienia czytelności kodu HTML. Podobne zasady obowiązują w PHP. Nie ma potrzeby wstawiania odstępów między instrukcjami PHP, ale kod będzie o wiele bardziej zrozumiały, jeżeli każda z nich zostanie umieszczona w osobnym wierszu. Na przykład:

```
echo 'witaj';  
echo 'świecie';
```

i

```
echo 'witaj ';echo 'świecie';
```

są równoznaczne, lecz wersja pierwsza jest łatwiejsza do odczytania.

Komentarze

Komentarze znaczą dokładnie tyle, co ich nazwa: w kodzie spełniają funkcję notatek dla czytających. Mogą być one użyte w celu wyjaśnienia przeznaczenia skryptu, podania jego autora, wyjaśnienia sposobu zapisu, wskazania daty ostatniej modyfikacji itd. Znajdują się we wszystkich, poza najprostszyimi, skryptach PHP.

Interpretator PHP zignoruje każdy tekst zawarty w komentarzu. Czyni tak zwłaszcza analizator składniowy, który omija komentarze, traktując je jak odstępy.

PHP rozpoznaje komentarze w stylu C, C++ i skryptów powłoki systemowej.

Poniżej znajduje się wielowierszowy komentarz w stylu C, który może pojawić się na początku skryptu:

```
/* Autor: Janek Nowak
   Ostatnia modyfikacja: 10 kwietnia
   Ten skrypt przetwarza zamówienia klientów
*/
```

Komentarze wielowierszowe powinny rozpoczynać się symbolem `/*`, a kończyć `*/`. Tak jak w języku C, nie mogą być one zagnieżdżane.

Można również używać komentarzy jednowierszowych, zarówno w stylu C++:

```
echo '<p>Zamówienie przyjęte.'; // Początek wydruku zamówienia
```

jak i w stylu skryptów powłoki:

```
echo '<p>Zamówienie przyjęte.'; # Początek wydruku zamówienia
```

W obu stylach wszystko, co następuje po symbolu komentarza (`#` lub `//`), jest traktowane jako komentarz. Dzieje się tak, dopóki nie zostanie osiągnięty koniec wiersza bądź kończący znacznik PHP, w zależności od tego, co pojawi się wcześniej.

W poniższym wierszu kodu tekst `oto komentarz` występujący przed znacznikiem zamykającym, stanowi część komentarza. Tekst `a to już nie`, znajdujący się za znacznikiem zamykającym, będzie natomiast traktowany jako zwykły kod HTML, ponieważ nie występuje on między znacznikami:

```
// oto komentarz ?> a to już nie
```

Dodawanie zawartości dynamicznej

PHP nie został dotychczas użyty do zadań, których nie można by wykonać w czystym kodzie HTML.

Podstawowym powodem stosowania przetwarzanego przez serwer języka skryptowego jest możliwość dostarczania użytkownikom stron o dynamicznej treści. Jest to ważna funkcja, ponieważ zawartość dostosowująca się do potrzeb użytkownika bądź ulegająca nieustannym zmianom przysięga na trwale uwagę odwiedzających stronę. PHP pozwala na łatwe zastosowanie tej funkcji.

Na początek prosty przykład. Należy zamienić kod PHP w pliku *przetworzzamowienie.php* następującym kodem:

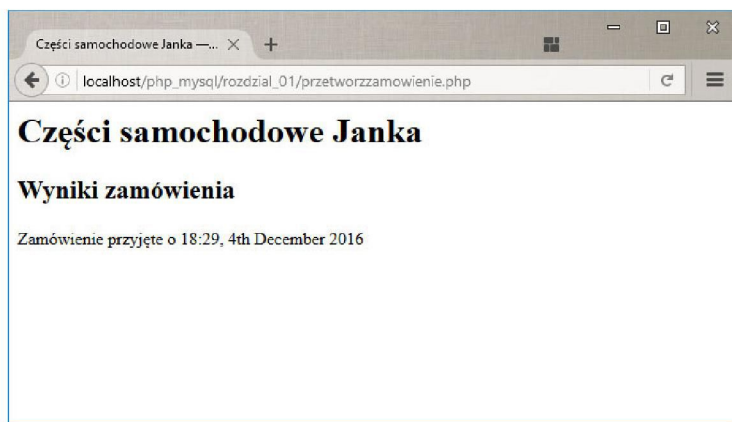
```
<?php
echo '<p>Zamówienie przyjęte o ';
echo date('H:i. jS F Y');
echo '</p>';
?>
```

Ten sam kod można by również zapisać w jednym wierszu, wykorzystując do tego operator konkatenacji (`.`), na przykład:

```
<?php
    echo '<p>Zamówienie przyjęte o '.date('H:i, jS F Y').'\</p>';
?>
```

W powyższym kodzie wbudowana w PHP funkcja `date()` została zastosowana po to, aby przekazać klientowi datę i godzinę przyjęcia jego zamówienia. Dane te ulegną zmianie przy każdym uruchomieniu skryptu. Wynik jednego z uruchomień jest przedstawiony na rysunku 1.3.

Rysunek 1.3.
Funkcja PHP `date()`
zwraca sformatowaną
datę



Wywoływanie funkcji

Warto zwrócić uwagę na ogólny sposób wywoływania funkcji `date()`. PHP posiada bardzo rozbudowaną bibliotekę funkcji używanych do tworzenia aplikacji WWW. Większość z nich wymaga dostarczenia im pewnych danych oraz zwraca inne.

Oto wywołanie funkcji:

```
echo date('H:i, jS F');
```

Należy zauważyć, że w nawiasach znajduje się łańcuch znaków przekazywany funkcji. Element znajdujący się w nawiasach nazywany jest *argumentem* lub *parametrem* funkcji. Argumenty są używane przez funkcję na wejściu, aby zwrócić określone wyniki.

Używanie funkcji `date()`

Argument przekazywany funkcji `date()` powinien być łańcuchem opisującym format, czyli pożądany styl zwracanego wyniku. Każda z liter w łańcuchu przedstawia jedną część zapisu daty i godziny. `H` to godzina w formacie dwudziestoczerogodzinnym, w razie konieczności poprzedzona zerami, `i` — minuty z występującym, kiedy to potrzebne, na pierwszym miejscu zerem, `j` — dzień miesiąca bez poprzedzającego zera, `S` przedstawia przyrostek porządkowy (w tym przypadku `th`), a `F` to miesiąc podany słownie.



Uwaga

Jeśli funkcja `date()` wyświetli ostrzeżenie informujące o tym, że nie została określona strefa czasowa, to do pliku `php.ini` należy dodać dyrektywę konfiguracyjną `date.timezone`. Więcej informacji na ten temat można znaleźć w przykładowym pliku `php.ini` zamieszczonym w dodatku A.