

1.2 1.2 Teoria

Step 1

Treść skryptu PHP musi być oddzielona specjalnymi znacznikami. Poniższy skrypt jest przykładem pierwszego skryptu w PHP:

```
<?php  
echo("Hello PHP!!!");  
?>
```

Oprócz powyższego znacznika `<?php ... ?>` wyróżniamy również następujące typy:

- Skrócony `<? ... ?>`
- Skryptowy `<script language = "php"> ... </script>`
- ASP `<% ... %>`

Komentarze w skryptach PHP są poprzedzone znakami `//` lub `#` (w przypadku komentarzy jednoliniowych) lub `/* ... */` (w przypadku komentarzy wieloliniowych).

```
<?php  
echo ("Hello PHP!!!")  
//This is a comment  
# This is another comment  
?>
```

Zmienna to obszar w pamięci komputera, w którym programista może umieścić jakieś wartości a następnie je odczytywać. Każdej zmiennej przypisujemy własną, unikalną nazwę, która jednoznacznie ją identyfikuje. Język PHP wymaga, aby zaczynała się ona od znaku

dolara, a następnie od litery (ew. podkreślenia). Dalsza część nazwy może już zawierać cyfry. Stosując wielkie litery trzeba uważać, ponieważ dla interpretera są one rozróżnialne od małych, co ma istotne znaczenie. `$zmienna` i `$Zmienna` to dwie różne zmienne. Przykłady poprawnych nazw zmiennych: `$a`, `$b`, `$foo`, `$_50`, `$_Foo`, `$moja_zmienna`, `$mojaZmienna3`. Przykłady niepoprawnych nazw zmiennych: `$5a`, `$'a'`, `$.`.

Wartość i typ zmiennej możemy sprawdzić za pomocą funkcji `var_dump()`, co pokazuje poniższy przykład:

```
<?php  
$length = 1258;  
var_dump($length);  
?>
```

W powyższym kodzie zmienna `$length` przechowuje liczbę całkowitą o wartości `1258`. Typ zmiennej wynika z wykonanego podstawienia. Jeśli do zmiennej wstawimy liczbę całkowitą `67`, to typem zmiennej będzie `int`, jeśli wstawimy napis `'Ann'`, to typem zmiennej będzie `string`. Po wstawieniu wartości zmiennopozycyjnej `56.8` zmienna będzie typu `float`. Zaś po wstawieniu wartości logicznej (`true` lub `false`), zmienna będzie miała typ `bool`.

```
<?php  
$a = 1258;  
$b = 'Ann';  
$c = 56.8;  
$d = true;  
var_dump($a);  
var_dump($b);  
var_dump($c);  
var_dump($d);  
?>
```

Do tej samej zmiennej możemy wstawiać wartości różnych typów. W PHP występują trzy funkcje umożliwiające zbadanie typu i wartości zmiennej.

- `var_dump()` - drukuje informacje wyłącznie o publicznych składowych
- `var_export()` - drukuje informacje o publicznych, chronionych oraz prywatnych składowych
- `print_r()` - drukuje to samo co `var_export()`, z różnicą taką, że generuje on poprawny kod PHP oraz nie drukuje informacji o typie zmiennej, o ile typem jest `string`, `int` lub `float`.

```

<?php
    $a = 1258;
    $b = 'Ann';
    $c = 56.8;
    var_dump($a);
    var_export($b);
    print_r($c);
    $c = false;
    print_r($c);
?>

```

Step 2

Operatory są to najprościej mówiąc symbole, które służą do operacji na zmiennych. Operatory dzielą się na operatory arytmetyczne, które służą do operacji na liczbach, operatory przypisania służące do przypisywania zmiennym wartości, operatory porównania niezbędne do instrukcji warunkowych, operatory inkrementacji i dekrementacji.

Operatory arytmetyczne

Nazwa	Wynik	Przykład
Dodawanie	Suma $\$a$ i $\$b$	$\$a + \b
Odejmowanie	Różnica $\$a$ i $\$b$	$\$a - \b
Mnożenie	Iloczyn $\$a$ i $\$b$	$\$a * \b
Dzielenie	Iloraz $\$a$ i $\$b$ (bez reszty)	$\$a / \b
Modulo	Reszta z dzielenia $\$a$ przez $\$b$	$\$a \% \b

```

<?php
    $a = 1258;
    $b = 56.8;
    $c = 3;
    echo($a + $b);
    echo("\n");
    echo($a - $b);
    echo("\n");
    echo($a * $b);
    echo("\n");
    echo($a / $c);
    echo("\n");
    echo($a % $c);
    echo("\n");
?>

```

Podstawowym operatorem przypisania jest symbol '='. Oczywiście nie oznacza on 'jest równe'. Wyrażenie $\$b = 5$ oznacza, że zmienna $\$b$ przyjmuje wartość równą 5. Zmiennej można przypisać także wartość innej zmiennej: $\$b = 5$; $\$a = \b ; – zmienią $\$a$ przyjmie wartość 5.

Zmiennym można przypisywać nie tylko konkretne wartości, ale też wartości innych zmiennych. Wartości te można przypisywać kaskadowo, przy czym wartości przypisywane będą od prawej do lewej, np.:

```

<?php
    $nazwa = $inna_nazwa = $trzecia_nazwa = 5;
?>

```

W tym wypadku wszystkim zmiennym zostanie przypisana wartość 5. Operator przypisania można łączyć z operatorami arytmetycznymi i operatorem łączenia ciągów, co pokazuje poniższa tabela.

Przykład	Wynik
$\$a += 2$	Do zmiennej $\$a$ dodane zostanie 2
$\$a -= 2$	Od zmiennej $\$a$ odjęte zostanie 2
$\$a *= 2$	Zmienna $\$a$ zostaje pomnożona przez 2
$\$a /= 2$	Zmienna $\$a$ zostaje podzielona przez 2
$\$a %= 2$	Zmienna $\$a$ przyjmuje wartość z dzielenia przez 2
$\$a .= "ciag tekstowy"$	Do zmiennej $\$a$ na końcu dodany zostanie ciąg "ciag tekstowy"

```

<?php
$a = 2048;
$b = "aLa ma kota";
echo($a += 2);
echo("\n");
echo($a -= 2);
echo("\n");
echo($a *= 2);
echo("\n");
echo($a /= 2);
echo("\n");
echo($a %= 2);
echo("\n");
echo($b .= " kot ma ale\n");
?>

```

Operatory porównania są niezbędne do korzystania z instrukcji warunkowych (jeśli coś to zrób coś). Zwracają one wartość TRUE (prawda – 1) lub FALSE (fałsz – 0).

Nazwa	Wynik	Przykład
<code>\$a == \$b</code>	Równy	Prawda jeśli <code>\$a</code> jest równe <code>\$b</code>
<code>\$a === \$b</code>	Identyczny	Prawda jeśli <code>\$a</code> jest równe <code>\$b</code> i są tego samego typu (Tylko w PHP4)
<code>\$a != \$b</code>	Różne	Prawda jeśli <code>\$a</code> nie jest równe <code>\$b</code>
<code>\$a !==\$b</code>	Nie identyczny	Prawda, jeśli <code>\$a</code> nie jest równe <code>\$b</code> lub nie są tego samego typu. (Tylko w PHP4)
<code>\$a < \$b</code>	Mniejsze	Prawda jeśli <code>\$a</code> jest mniejsze niż <code>\$b</code>
<code>\$a > \$b</code>	Większe	Prawda jeśli <code>\$a</code> jest większe niż <code>\$b</code>
<code>\$a <= \$b</code>	Mniejsze lub równe	Prawda jeśli <code>\$a</code> jest mniejsze lub równe <code>\$b</code>
<code>\$a >= \$b</code>	Większe lub równe	Prawda jeśli <code>\$a</code> jest większe lub równe <code>\$b</code>

```

<?php
echo(2 == 2);
echo(2 === 2);
echo(2 != 4);
echo(2 < 4);
echo(2 > 4);
echo(2 <= 2);
echo(2 >= 2);
?>

```

Operatory inkrementacji oraz dekrementacji występują w większości języków programowania. Służą one do zmniejszenia lub zwiększenia wartości danej zmiennej o 1. Każdy operator można stosować na 2 sposoby: preinkrementacja/predekrementacja – najpierw wartość zmiennej zostanie zmieniona, a później zwrócona, lub postinkrementacji/postdekrementacji – najpierw zostanie zwrócona wartość zmiennej, a następnie wartość zmiennej zostanie zmieniona.

Nazwa	Wynik	Przykład
Preinkrementacja	Zwiększa <code>\$a</code> o jeden, a następnie zwraca <code>\$a</code>	<code>++\$a</code>
Postinkrementacja	Zwraca <code>\$a</code> , a następnie zwiększa <code>\$a</code> o jeden	<code>\$a++</code>
Predekrementacja	Zmniejsza <code>\$a</code> o jeden, a następnie zwraca <code>\$a</code>	<code>--\$a</code>
Postdekrementacja	Zwraca <code>\$a</code> , a następnie zmniejsza <code>\$a</code> o jeden	<code>\$a--</code>

```
<?php
echo "Postinkrementacja\n";
$a = 5;
echo "Powinno być 5: " . $a++ . "\n";
echo "Powinno być 6: " . $a . "\n";
echo "Preinkrementacja\n";
$a = 5;
echo "Powinno być 6: " . ++$a . "\n";
echo "Powinno być 6: " . $a . "\n";
echo "Postdekrementacja\n";
$a = 5;
echo "Powinno być 5: " . $a-- . "\n";
echo "Powinno być 4: " . $a . "\n";
echo "Predekrementacja\n";
$a = 5;
echo "Powinno być 4: " . --$a . "\n";
echo "Powinno być 4: " . $a . "\n";
?>
```

Operatory logiczne służą do budowania bardziej skomplikowanych instrukcji warunkowych – do łączenia kilku warunków w jednej instrukcji.

Nazwa	Wynik	Przykład
AND	Prawda, jeśli <code>\$a</code> i <code>\$b</code> są prawdą	<code>\$a && \$b</code>
OR	Prawda, jeśli <code>\$a</code> lub <code>\$b</code> są prawdą	<code>\$a \$b</code>
NOT	Prawda, jeśli <code>\$a</code> nie jest prawdą	<code>! \$a</code>

```
<?php
$a = 1;
$b = 2;
$c = 3;
$w1 = ($a < $b) && ($c > $b);
$w2 = ($a == $b) || ($b < $c);
$w3 = ($a > $b) || ( ($a < $b) && ($b > $c) );
var_dump($w1);
var_dump( $w2);
var_dump($w3);
var_dump(!$w3);
?>
```

Step 3

Czasem chcemy, żeby fragment kodu wykonał się tylko pod jakimś warunkiem. Stosuje się wtedy instrukcję `if`:

```
<?php
$a = 2;
$b = 1;
if($a>$b)
    echo("$a jest większe od $b");
?>
```

Jeżeli wartość `$a` jest większa od `$b` wyświetli się napis "`$a jest większe od $b`". Warunek jest spełniony, jeżeli wyrażenie w nawiasie ma wartość różną od zera. Jeżeli będzie to np. pusta zmienna warunek nie jest spełniony. Polecenie w następnej linii zostanie wykonane, jeżeli warunek jest spełniony.

Aby warunek objął kilka poleceń, stosuje się nawiasy klamrowe:

```
<?php
$a = 2;
$b = 1;
if($a>$b) {
    echo("$a jest większe od $b\n");
    $a++;
    echo "Wartość \$a == " . $a;
}
?>
```

Jeżeli chcemy wykonać inny fragment kodu gdy warunek nie jest spełniony, stosujemy instrukcję `else`:

```
<?php
$a = 1;
$b = 2;
if($a>$b) {
    echo("$a jest większe od $b\n");
}
else{
    echo("$b jest większe od $a\n");
}
?>
```

Aby uzyskać bardziej złożony warunek można zastosować operatory logiczne i (`&&`), lub (`||`).

```
<?php
$a = 1;
$b = 2;
if($a>$b || $a<2)
    echo("$a jest większe od $b, lub mniejsze od 2\n");
?>
```

Step 4

Aby fragment kodu wykonać wiele razy stosuje się pętle. PHP obsługuje 3 rodzaje pętli: `while`, `do..while` i `for`. Najprostszą z nich jest pętla `while`:

```
<?php
while (warunek) {
    //instrukcje do wykonania
}
?>
```

```
<?php
$a=0;
while($a<5)
{
    echo("$a ");
    $a++;
}
?>
```

Powyższy skrypt wyświetli liczby od 0 do 4. W pętli while najpierw sprawdzany jest warunek (w tym wypadku `$a<5`). Jeżeli jest spełniony, pętla wykonuje się i wraca do sprawdzenia warunku. Jeżeli warunek nie jest spełniony, wykonanie pętli kończy się.

Pętla `do..while` to specyficzna odmiana pętli `while`, bo jeśli we `while` warunek jest na starcie fałszywy, to pętla ani razu nie wykona bloku instrukcji. W wypadku użycia `do..while` pętla wykona blok instrukcji przynajmniej raz (nawet wtedy, gdy warunek jest od początku fałszywy). Pętla `do..while` różni się od pętli `while` tym, że najpierw wykonuje się pętla, a dopiero potem sprawdzany jest warunek. Oznacza to, że pętla zawsze wykonana będzie co najmniej 1 raz.

```
<?php
do {
    //instrukcje do wykonania
} while (warunek);
?>
```

```
<?php
$a=6;
do
{
    echo("$a ");
    $a++;
}while($a<5); /* ta pętla wykona się 1 raz */
$a=6;
while($a<5)
{
    echo("$a ");
    $a++;
} /* instrukcje w tej pętli nie zostaną wykonane */
?>
```

Pętla `for` jest używana tylko wtedy, gdy zachodzi potrzeba wykonania jakiegoś kodu określoną liczbę razy (założoną z góry przez autora lub pochodzącą ze zmiennej).

```
<?php
for (inicjalizacja zmiennych; sprawdzenie warunku; modyfikacja zmiennych) {
    //instrukcje do wykonania
}
?>
```

Wykonanie pętli for:

```
<?php
for($a=0;$a<5;$a++)
{
    echo("$a ");
}
?>
```

Odpowiada wykonaniu pętli:

```
<?php
$a=0;
while($a<5)
{
    echo($a. ' ');
    $a++;
}
?>
```

Wykonanie pętli można w każdym momencie zakończyć. Służy do tego instrukcja `break`:

```
<?php
$a=0;
while($a<10)
{
    echo "$a ";
    $a++;
    if($a==3)
        break;
}
?>
```

Powyższa pętla nie wykona się 10 razy - gdy `$a` osiągnie wartość 3, wykonanie pętli zostanie przerwane.

Funkcja `continue` powoduje przerwanie aktualnej iteracji (przebiegu) pętli i wykonanie jej od nowa.

```
<?php
$a=0;
while($a<10)
{
    echo "$a\n";
    $a++;
    if($a==3)
        continue;
    echo("aaa"); /* ta instrukcja wykona
się tylko gdy $a nie jest równe 3 */
}
?>
```

Instrukcja `switch` jest rodzajem skondensowanej instrukcji warunkowej, którą zazwyczaj zastępujemy rozbudowane i wielokrotne użycia `else if`.

```
<?php
$miasto = "hel";
switch ($miasto) {
    case 'warszawa': echo "Pochodzisz ze stolicy?"; break;
    case 'hel': echo "Mieszkaś nad morzem?"; break;
    case 'sanok': echo "A może w Bieszczadach?"; break;
    default: echo "Miasto nierozniane." ;
}
?>
```

Poniższy kod:

```
<?php
$a = 1;
if($a==1)
{
    echo("a jest równe 1");
}
if($a==3)
{
    echo("a jest równe 3");
}
if($a==11)
{
    echo("a jest równe 11");
}
?>
```

Można zastąpić przez:

```

<?php
$a = 1;
switch($a) {
    case 1:
        echo("a jest równe 1");
        break;
    case 3:
        echo("a jest równe 3");
        break;
    case 11:
        echo("a jest równe 11");
        break;
}
?>

```

Step 5

Instrukcja `switch` jest rodzajem skondensowanej instrukcji warunkowej, którą zazwyczaj zastępujemy rozbudowane i wielokrotne użycia `else if`.

```

<?php
$miasto = "hel";
switch ($miasto) {
    case 'warszawa': echo "Pochodzisz ze stolicy?"; break;
    case 'hel': echo "Mieszkaś nad morzem?"; break;
    case 'sanok': echo "A może w Bieszczadach?"; break;
    default: echo "Miasto nierozpoznane.";
}
?>

```

Poniższy kod:

```

<?php
$a = 1;
if($a==1)
{
    echo("a jest równe 1");
}
if($a==3)
{
    echo("a jest równe 3");
}
if($a==11)
{
    echo("a jest równe 11");
}
?>

```

Można zastąpić przez:

```

<?php
$a = 1;
switch($a) {
    case 1:
        echo("a jest równe 1");
        break;
    case 3:
        echo("a jest równe 3");
        break;
    case 11:
        echo("a jest równe 11");
        break;
}
?>

```

Step 6

Tablice są bardzo specyficzny typem zmiennych – są to, najprościej mówiąc, zmienne zawierające w sobie uporządkowany zbiór zmiennych. Do zmiennych tych uzyskuje się dostęp przez liczbę w nawiasie kwadratowym podane bezpośrednio po nazwie zmiennej – tablicy. Liczba ta to tak zwany indeks – numer kolejny zmiennej w tablicy. Tak samo przypisuje się wartość do tablicy.

```

<?php
$tablica[0] = "Wpis numer 0";
$tablica[1] = "Wpis numer 1";
$tablica[2] = "Wpis numer 2";
echo $tablica[2]; // Wyświetlony zostanie napis "Wpis numer 2";
?>

```

Aby po prostu dodać kolejny wpis na końcu tabeli wystarczy przy przypisywaniu wartości nie wpisywać indeksu do nawiasów kwadratowych. Jeśli w ten sposób dodawane są wpisy do nowej tablicy, to pierwszy wpis ma indeks 0. Indeks można też podawać ze zmiennej, z innej tablicy czy funkcji – z dowolnego wyrażenia zwracającego wartość.

```
<?php
$tab1[] = 1;
$tab1[] = 0;
$tab1[] = 3;
$tab1[] = 2;
$tab2[] = "Pierwszy";
$tab2[] = "Drugi";
$tab2[] = "Trzeci";
$tab2[] = "Czwarty";
echo $tab2[$tab1[2]];
?>
```

Elementem tablicy może być każdy typ zmiennej (z innymi tablicami i obiektami włącznie). Innym ze sposobów deklaracji tablic jest użycie słowa kluczowego `Array`:

```
<?php
$woce = array ("mango", "papaja", "banan", "aronia");
$woce2 = array ("d"=>"mango", "a"=>"papaja", "b"=>"banan", "c"=>"aronia");
var_dump( $woce );
var_dump( $woce2 );
?>
```

W PHP występuje też inny rodzaj tablic, tak zwane tablice asocjacyjne (zwane też czasem haszami – hash table). Są to tablice, w których zamiast indeksów liczbowych używa się identyfikatorów znakowych (kluczy):

```
<?php
$tablica["imie"] = "Jan";
$tablica["nazwisko"] = "Kowalski";
$tablica["adres"] = "Polna 1";
echo $tablica["imie"]." ".$tablica["nazwisko"].", ul. ".$tablica["adres"]."\n";
?>
```

Pętla `foreach` ułatwia obsługę tablic i tablic asocjacyjnych. Poniższy przykład pokazuje, jak łatwo zmienić pętlę `for` na `foreach` w wypadku korzystania z tablic.

```
<?php
// tworzymy tablicę asocjacyjną
$tablica = array("imie" => "Jan", "nazwisko" => "Kowalski", "email" => "jankowal@gmail.com");
// Wyświetlamy
foreach ($tablica as $klucz => $dana)
{
    echo 'Klucz to ' . $klucz. ' a jego wartość to ' . $dana. "\n";
}
?>
```

Step 7

PHP umożliwia także deklaracje tzw. tablic wielowymiarowych. Polega to na tworzeniu kolejnych węzłów elementu dając przy tym wrażenie drzewa elementów. Tablice wielowymiarowe można deklarować podając kolejne indeksy w nawiasach kwadratowych:

```
<?php
$dane[0]['imię'] = 'Jan';
$dane[0]['nazwisko'] = 'Kowalski';
$dane[0]['ulica'] = 'Kowalewska';
$dane[1]['imię'] = 'Maciej';
$dane[1]['nazwisko'] = 'Nowak';
$dane[1]['ulica'] = 'Nowakowska';
print_r($dane);
?>
```

PHP umożliwia zamianę ciągów na tablice i odwrotnie. Zamiana ciągu na tablicę jest bardzo przydatna jeśli zachodzi potrzeba wyciągnięcie jakiegoś fragmentu danych z ciągu. Założmy że w odczytaliśmy z pliku z danymi (o odczytanie z plików w jednym z kolejnych rozdziałów) linię z logu zapisanego przez licznik WWW: `"12/11/2000;19:23:33;Netscape Navigator;192.168.1.1"`. Jak widać dane rozdzielone są średnikami. Do rozdzielania ciągów na tablicę służy funkcja `explode()`. Jako pierwszy parametr trzeba do niej podać znak lub dłuższy ciąg który oddziela kolejne pola, jako drugi ciąg do rozdzielenia. Opcjonalnie można podać trzeci argument, który oznacza maksymalną liczbę pól – jeśli jest ich więcej niż ta liczba, to ostatnie pole będzie zawierało wszystkie pozostałe pola. Funkcja zwraca tablicę zawierającą kolejne pola.

```
<?php
$dane = "12/11/2000;19:23:33;Netscape Navigator;192.168.1.1";
$tablica = explode(";", $dane);
print_r($tablica);
?>
```

Jest także rozszerzona wersja funkcji `explode`: `split()`. Różni się ona tym, że zamiast prostego ciągu znaków rozdzielających pola, akceptuje ona wyrażenia regularne. Czasem potrzebne jest działanie w drugą stronę: złączenie pól tablicy w jeden ciąg, w którym pola oddzielone są jakimś znakiem (lub kilkoma). Do tego służy funkcja `implode()`. Jako pierwszy parametr podawany jest ciąg za pomocą którego "sklejane" są elementy tablicy, a jako drugi właśnie tablica do posklejania. Zwracany jest ciąg zawierający posklejane elementy. Jako przykład zastosowania może posłużyć właśnie zapisywanie danych o użytkowniku w aplikacji licznika odwiedzin – tablica zawiera dane o odwiedzającym, a potrzebny jest ciąg pooddzielany średnikami.

```
<?php
$dane = "12/11/2000;19:23:33;Netscape Navigator;192.168.1.1";
$tablica = explode(";", $dane);
$dane = implode(";", $tablica);
print_r($dane);
?>
```

Step 8

Funkcje w PHP są częściowo podobne do funkcji matematycznych. Mogą przyjmować argumenty oraz zwracać wartości. Nie jest to jednak wymogiem. Funkcją nazywamy napisany przez nas kod, zamknięty w nawiasy klamrowe, poprzedzony słowem kluczowym `function` oraz unikatową nazwą.

```
<?php
function wyswietl_powitanie() // deklaracja funkcji
{
    echo "Witam serdecznie!"; // ciało funkcji, czyli
    echo "Proszę się zarejestrować."; // instrukcje do wykonania
}
wyswietl_powitanie();
?>
```

Przeanalizujmy napisany kod. Przedstawia funkcję, której zadaniem jest wyświetlenie na ekranie dwóch komunikatów. Jest to jedynie deklaracja. Żeby użyć tak napisanej funkcji należy ją wywołać. Jak widać, żeby użyć napisanej funkcji, wystarczy napisać jej nazwę. Jak każde polecenie w PHP, tak również wywołanie napisanej przez nas funkcji, musi być zakończone średnikiem. Co znaczą nawiasy między nazwą, a średnikiem? Jest to miejsce na podanie argumentów funkcji. Skoro nasza funkcja nie przyjmuje żadnych, są one puste. Nie należy jednak zapominać o nich umieszczeniu, nawet gdy funkcja jest bezargumentowa!

Poprzednio omawialiśmy funkcje, których wykonywane zadanie zamykało się wewnątrz struktury. Znaczy to tyle, że nie miały wpływu na wykonanie dalszego kodu. Tym razem zajmiemy się funkcjami, które coś wnoszą do programu. Żeby funkcja zwróciła wartość do programu głównego, musimy umieścić zwracane wyrażenie po słowie `return`.

```
<?php
function tresc_powitania() // deklaracja funkcji
{
    return "Witam wszystkich!";
}
$powitanie = tresc_powitania();
echo $powitanie;
?>
```

Żeby zrozumieć zasadę działania funkcji zwracającej wartość, wyobraźmy sobie, że funkcja to taka zmienna, której wartość zmienia się dynamicznie. Co za tym idzie, funkcję możemy przypisać zmiennej. Dodatkowo możemy sprawdzić, czy funkcja jest mniejsza lub większa od pewnej liczby. Oczywiście, mówiąc funkcja, mamy na myśli wartość zwracaną przez daną funkcję.

```
<?php
function oblicz()
{
    $zm1 = 3;
    $zm1 += 5;
    $zm1++;
    return $zm1;
}
if (oblicz() > 5)
    echo "Funkcja zwraca wartość większą od 5";
else
    echo "Wartość zwracana przez funkcję jest mniejsza od 6";
?>
```

Co nazywamy argumentami? Wszystkie wartości przekazywane w nawiasie, zaraz po nazwie funkcji. Jest to bardzo przydatna rzecz, która w dużej mierze usprawnia programowanie. Zmienna przekazana jako argument może być używana i modyfikowana wewnątrz funkcji, bez wpływu na jej wartość w programie głównym.

```
<?php
function przywitaj($zmienna_z_imieniem)
{
    echo 'Witaj '.$zmienna_z_imieniem.'!';
}
$imie = "Marcin";
przywitaj($imie);
?>
```

Kilka słów wyjaśnień. Tworząc deklarację funkcji, podajemy jako argumenty fikcyjne nazwy zmiennych. Następnie, wywołując funkcję, podajemy istniejącą zmienną, która jest podstawiana pod zmienną fikcyjną. Podsumowując, działa to tak, że wszędzie, gdzie użыта była `$zmienna_z_imieniem`, użta zostanie `$imie`. Funkcja wyświetli "Witaj Marcin!" .

Argumenty funkcji mogą być przekazywane na dwa sposoby: przez wartość oraz za pomocą referencji. W prezentowanych do tej pory przykładach wykorzystywane było domyślne przekazywanie argumentów przez wartość. Oznacza to, że w rzeczywistości funkcja otrzymuje kopię argumentów źródłowych i wszelkie operacje wykonuje na tych kopiiach. Nie jest więc w stanie dokonać żadnej modyfikacji oryginału.

```
<?php
function dodajJeden($liczba){
    $liczba = $liczba + 1;
}
$liczba = 1;
echo "Przed wywołaniem funkcji: $liczba\n";
dodajJeden($liczba);
echo "Po wywołaniu funkcji: $liczba\n";
?>
```

Jeśli jednak chcemy mieć możliwość modyfikowania w funkcji argumentu oryginalnego, trzeba zastosować sposób drugi – przekazywanie przez referencję. W tym celu przed nazwą argumentu należy umieścić znak `&` (ampersand).

```
<?php
function dodajJeden(&$liczba){
    $liczba = $liczba + 1;
}
$liczba = 1;
echo "Przed wywołaniem funkcji: $liczba\n";
dodajJeden($liczba);
echo "Po wywołaniu funkcji: $liczba\n";
?>
```

PHP pozwala na konstruowanie funkcji o domyślnych wartościach argumentów. Dzięki temu część parametrów lub nawet wszystkie będą mogły być pominięte w wywołaniu, a w ich miejsce zostaną wstawione wartości zdefiniowane podczas tworzenia funkcji.

```
<?php
function dodaj($wartosc, $ile = 1){
    return $wartosc + $ile;
}
$liczba1 = 10;
$liczba2 = dodaj($liczba1, 5);
echo "\$liczba2 == $liczba2\n";
$liczba2 = dodaj($liczba1);
echo "\$liczba2 == $liczba2\n";
?>
```

Funkcją rekurencyjną nazywamy funkcję odwołującą się do siebie samej. Przykład powinien nieco rozjaśnić wątpliwości.

```
<?php
function silnia($liczba)
{
    if($liczba < 2)
        return 1;
    else
        return $liczba*silnia($liczba-1);
}
echo silnia(5);
?>
```

Przeanalizujmy działanie kodu. Jeżeli liczba jest mniejsza od 2, czyli 0 lub 1, zwrócona zostanie wartość 1 (z definicji funkcji). Jeśli natomiast liczba jest większa, wywołujemy funkcję ponownie z argumentem pomniejszonym o 1. Wynika to z faktu, że $4! = 4 * 3!$. Robimy tak dopóki nie zejdziemy do jedynki. Podstawową zaletą funkcji rekurencyjnych jest prostota kodu. Na funkcji obliczającej silnię nie widać tego tak znacząco, lecz pisząc bardziej rozbudowane konstrukcje, jest to zauważalne. Programiści jednak odchodzą od stosowania funkcji rekurencyjnych z racji dużej ilości pamięci, zajmowanej podczas kolejnych wywołań. Można sobie to łatwo zobrazować. Licząc silnię z dziesięciu, musimy wywołać funkcję dziesięć razy (z argumentem: 10, 9, 8 itd.). Dodatkowo system musi zapamiętać wynik zwracany przez każdą z funkcji. Z tego powodu bardzo zachęcam do stosowania klasycznych funkcji.

Wiedząc co to jest funkcja zrobimy teraz krótki przegląd funkcji, mających miejsce w tablicach:

- `int array_push (array &tablica, mixed wartość [, mixed ...])` - `array_push()` traktuje zmienną `tablica` jako stos i wstawia przekazane parametry na koniec podanej tablicy. Długość parametru `tablica` zwiększa się o liczbę przekazanych wartości.

```
<?php
$stos = array("pomarańcza", "banan");
array_push($stos, "jabłko", "malina");
print_r($stos);
?>
```

- `int array_unshift (array &tablica, mixed wartość [, mixed ...])` - `array_unshift()` wstawia jeden lub więcej przekazanych jako parametry elementów na początek tablicy `tablica`. Zauważ, że lista elementów wstawiana jako całość, więc elementy zostają w takim samym porządku. Wszystkie klucze liczbowe zostaną zmodyfikowane tak, aby ich wartości zaczynały się od zera, podczas gdy klucze znakowe nie zostaną zmienione. Funkcja zwraca nową liczbę elementów w tablicy `tablica`.

```
<?php
$kolejka = array ("pomarańcza", "banan");
array_unshift ($kolejka, "jabłko", "malina");
print_r($kolejka);
?>
```

- `mixed array_pop (array &tablica)` - `array_pop()` zdejmuje i zwraca ostatnią wartość tablicy `tablica`, skracając tą tablicę o jeden element. Jeśli tablica jest pusta (lub nie jest tablicą), zwracana jest wartość `NULL`.

```
<?php
$stos = array("pomarańcza", "banan", "jabłko", "malina");
$owoc = array_pop($stos);
print_r($stos);
?>
```

- `mixed array_shift (array &tablica)` - `array_shift()` usuwa pierwszą wartość parametru `tablica` i zwraca go skracając tą tablicę o jeden element przesuwając wszystkie pozostałe elementy w dół. Wszystkie klucze liczbowe zostaną zmodyfikowane tak, aby ich wartości zaczynały się od zera, podczas gdy klucze znakowe nie zostaną zmienione. Jeśli tablica jest pusta (lub nie jest tablicą), zwracana jest wartość `NULL`.

```
<?php
$stos = array ("pomarańcza", "banan", "jabłko", "malina");
$owoc = array_shift ($stos);
print_r($stos);
print_r($owoc);
?>
```

- Funkcja `unset();` - pozwala usunąć pojedynczą, lub więcej zmiennych, lub element tablicy.

```
<?php
$owoce = array ("pomarańcza", "banan", "jabłko", "malina");
print_r($owoce);
unset($owoce[1]); // usunięcie pojedynczej zmiennej
print_r($owoce);
unset($owoce[0], $owoce[3], $owoce[5]); // usunięcie wielu zmiennych
print_r($owoce);
?>
```

- `int rand ([int $min [, int $max]])` - Jeśli wywołana bez opcjonalnych argumentów `min` i `max`, funkcja `rand()` zwraca pseudolosową liczbę stałoprzecinkową z przedziału pomiędzy 0 a `RAND_MAX`. Dla uzyskania liczby losowej z przedziału np. od 5 do 15 (włącznie), należy wywołać `rand(5, 15)`.

```
<?php
echo rand() . "\n";
echo rand() . "\n";
echo rand(5, 15);
?>
```

- `sort()` - sortuje zwykłe tablice (nie asocjacyjne) w kolejności alfabetycznej.

```
<?php
$owoce = array ("pomarańcza", "banan", "jabłko", "malina");
print_r($owoce);
sort($owoce);
print_r($owoce);
?>
```

- `rsort()` - sortuje zwykłe tablice (nie asocjacyjne) w odwróconej kolejności.

```
<?php
$woce = array ("pomarańcza", "banan", "jabłko", "malina");
print_r($woce);
rsort($woce);
print_r($woce);
?>
```

- `asort()` - sortuje tablice asocjacyjne zachowując przypisanie kluczy do wartości.

```
<?php
$woce = array ("d"=>"mango", "a"=>"papaja", "b"=>"banan", "c"=>"aronia");
print_r($woce);
asort($woce);
print_r($woce);
?>
```

- `arsort()` - sortuje w odwrotnej kolejności tablice asocjacyjne zachowując przypisanie kluczy do wartości.

```
<?php
$woce = array ("d"=>"mango", "a"=>"papaja", "b"=>"banan", "c"=>"aronia");
print_r($woce);
arsort($woce);
print_r($woce);
?>
```

- `ksort()` - sortuje tablice asocjacyjne według kluczy.

```
<?php
$woce = array ("d"=>"mango", "a"=>"papaja", "b"=>"banan", "c"=>"aronia");
print_r($woce);
ksort($woce);
print_r($woce);
?>
```

- `current` - Zwraca bieżący element tablicy. Funkcja `current()` po prostu zwraca element tablicy, na który aktualnie wskazuje wewnętrzny wskaźnik. Nie przesuwa ona wskaźnika. Jeśli wewnętrzny wskaźnik jest poza końcem listy elementów, `current()` zwraca `false`.

```
<?php
$tablica = array('krok pierwszy', 'krok drugi', 'krok trzeci', 'krok czwarty');
echo current($tablica);
?>
```

- `next` - Przesuń do przodu wewnętrzny wskaźnik tablicy. Przesuwa wewnętrzny wskaźnik tablicy i jedną pozycję do przodu i zwraca element tablicy aktualnie wskazywany przez wskaźnik, lub `false` jeśli nie ma już więcej elementów.

```
<?php
$tablica = array('krok pierwszy', 'krok drugi', 'krok trzeci', 'krok czwarty');
echo current($tablica) ."\n";
next($tablica);
echo current($tablica) ."\n";
next($tablica) ;
echo current($tablica) ."\n";
next($tablica);
echo current($tablica) ."\n";
?>
```

- `prev` - Cofnij wewnętrzny wskaźnik tablicy. Zwraca wartość z tablicy z miejsca poprzedniego od tego na które wskazywał wewnętrzny wskaźnik pliku, lub `false` jeśli nie ma już więcej elementów.

```
<?php
$tablica = array('krok pierwszy', 'krok drugi', 'krok trzeci', 'krok czwarty');
echo current($tablica) ."\n";
next($tablica);
echo current($tablica) ."\n";
prev($tablica) ;
echo current($tablica) ."\n";
?>
```

- `reset` - Ustaw wewnętrzny wskaźnik tablicy na jej pierwszy element. `reset()` przewija wewnętrzny wskaźnik tablicy parametru tablica na jego pierwszy element i zwraca jego wartość, lub `false` jeśli tablica jest pusta.

```
<?php
    $tablica = array('krok pierwszy', 'krok drugi', 'krok trzeci', 'krok czwarty');
    echo current($tablica) ."\n";
    next($tablica);
    echo current($tablica) ."\n";
    next($tablica) ;
    echo current($tablica) ."\n";
    reset($tablica);
    echo current($tablica) ."\n";
?>
```

- `bool in_array(mixed $igła , array $stógi_siana [, bool $ścisłe])` - Przeszukuje `$stógi_siana` w poszukiwaniu parametru `igła` i zwraca true jeśli wartość została znaleziona lub false w przeciwnym przypadku.

```
<?php
    $a = array(0,1,2,3,4,5);
    echo (int) in_array("0", $a);
    echo (int) in_array("3", $a);
    echo (int) in_array("8", $a);
?>
```

- `mixed array_search(mixed $igła , array $stógi_siana [, bool $ścisły])` - Przeszukuje `$stógi_siana` w poszukiwaniu parametru `igła` i zwraca odpowiedni klucz jeśli został on znaleziony lub false w przeciwnym przypadku.

```
<?php
    $tablica = array(0 => 'niebieski', 1 => 'czerwony', 2 => 'zielony', 3 => 'czerwony');
    $klucz = array_search('zielony', $tablica);
    echo "$klucz\n";
    $klucz = array_search('czerwony', $tablica);
    echo "$klucz\n";
    $klucz = array_search('hiacyntowy', $tablica);
    echo "$klucz\n";
?>
```

- `array array_intersect (array $tablica1 , array $tablica2 [, array $...])` - `array_intersect()` zwraca tablicę zawierającą wszystkie wartości tablicy `tablica1` które istnieją we wszystkich argumentach. Zauważ, że zachowywane są przypisania kluczy.

```
<?php
    $tablica1 = array ("a" => "zielony", "czerwony", "niebieski");
    $tablica2 = array ("b" => "zielony", "żółty", "czerwony");
    $wynik = array_intersect ($tablica1, $tablica2);
    print_r($wynik);
?>
```

- `array array_diff (array $tablica1 , array $tablica2 [, array $...])` - `array_diff()` zwraca tablicę zawierającą wszystkie wartości tablicy `tablica1` które nie są obecne w innych tablicach-argumentach. Zauważ, że zachowywane są klucze.

```
<?php
    $tablica1 = array ("a" => "zielony", "czerwony", "niebieski", "czerwony");
    $tablica2 = array ("b" => "zielony", "żółty", "czerwony");
    $wynik = array_diff ($tablica1, $tablica2);
    print_r($wynik);
?>
```

Step 10

W PHP konsola jest interfejsem wiersza poleceń, nazywanym także powłoką interaktywną. Możemy uzyskać do niego dostęp, wpisując następujące polecenie w terminalu:

```
php -a
```

Jeśli wpiszemy jakikolwiek kod PHP w powłoce i wciśniemy enter, zostanie on wykonany bezpośrednio i wyświetli dane wyjściowe lub pokaze komunikaty o błędach w przypadku jakiegokolwiek błędu. Przykładowe uruchomienie kodu PHP, który odczytuje dane wejściowe z konsoli PHP wygląda następująco:

Istnieją dwie metody odczytu konsoli lub danych wejściowych użytkownika w PHP: poprzez funkcję `readline` lub `fscanf`. Funkcja `readline()` jest funkcją wbudowaną w PHP. Ta funkcja służy do odczytywania danych wejściowych konsoli. Następujące rzeczy można osiągnąć za pomocą funkcji `readline()`:

```
<?php
$a = readline('Enter a string: ');
echo $a;
?>
```

Domyślnie typ danych zmiennej akceptowany przez funkcję `readline()` to ciąg tekstowy. Tak więc dla każdego innego typu danych musimy go jawnie zeskanować, jak opisano poniżej.

```
<?php
$a = (int)readline('Enter an integer: ');
$b = (float)readline('Enter a floating' . ' point number: ');
echo "Entered integer is " . $a . " and entered float is " . $b;
?>
```

Możemy osiągnąć to samo bez monitowania użytkownika również:

```
$a = readline();
```

W tym przypadku, gdy tylko użytkownik naciśnie enter, wprowadzona wartość jest przechowywana w zmiennej `a`.

Funkcje `readline()` można łączyć z funkcją `explode()`. W poniższym przykładzie separatorem jest spacja. Drugim argumentem jest funkcja `readline()`. Tutaj również typem danych `$var1` i `$var2` będzie ciąg tekstowy. Musimy więc oddziennie rzutować je dla innych typów danych. W poniższym przykładzie rzutowanie typu jest pokazane dla liczb całkowitych.

```
<?php
$array = explode(' ', readline());
$var1 = (int)$array[0];
$var2 = (int)$array[1];
echo "The sum of " . $var1 . " and " . $var2 . " is " . ($var1 + $var2);
print_r($array);
?>
```

Użycie funkcji `fscanf()` działa tak samo jak funkcja `fscanf()` w języku C. Możemy odczytać 2 liczby całkowite z klawiatury (`stdin`), jak poniżej:

```
<?php
fscanf(STDIN, "%d %d", $a, $b);
echo "The sum of " . $a . " and " . $b . " is " . ($a + $b);
?>
```

Nie ma potrzeby używania jawnego rzutowania typów dla funkcji `fscanf()`, ponieważ jest to wykonywane przez specyfikatory formatu, np. `%d`, `%f`, `%c` itd. Ponadto funkcja `fscanf()` jest znacznie szybsza niż funkcja `readline()`.

Ciągi znaków są jednym z najczęściej używanych typów danych, prawdopodobnie niezależnie od języka programowania. Łańcuchy mogą być zakodowane na stałe (określone bezpośrednio przez programistę) lub sformatowane (gdzie określony jest podstawowy szkielet, a końcowy ciąg jest uzyskiwany przez włączenie wartości innych zmiennych). Sformatowane ciągi można zdefiniować jako zestaw segmentów, w których każdy segment może zawierać liczbę całkowitą, zmiennoprzecinkową lub nawet inny ciąg. Sformatowane ciągi używają **specyfikatorów formatu** do tworzenia podstawowej struktury ciągu. Specyfikatory formatu to predefiniowana sekwencja znaków, której można użyć do zdefiniowania typu danych, który ma być przechowywany lub wyświetlany, a także sposobu formatowania dowolnej wartości, tj. Precyzyj, dopełnienia itp. Specyfikatory formatu, ogólnie rzecz biorąc, rozpoczynają się od symbolu centyla lub „%” po którym następuje sekwencja znaków definiująca typ danych i żądany format. Podczas iteracji po formacie, jeśli napotkany zostanie jakikolwiek specyfikator formatu, kompilator / interpreter rozumie, że istnieje odpowiednia dyrektywa, której wartość ma zostać sformatowana i użyta. W związku z tym ciąg może w ogóle nie zawierać specyfikatora formatu, ale jeśli zawiera co najmniej taką samą liczbę dyrektyw, powinien również zostać ponownie wysłany. W przypadku nadmiernej liczby dyrektyw, niektóre języki po prostu ignorują niepotrzebne i pozwalają na wykonanie z ostrzeżeniem.

W języku PHP wyróżniamy następujące specyfikatory formatów:

- **%** - Wyświetla znak %. Nie jest wymagana żadna dyrektywa.
- **b** - Odwołuje się do liczby całkowitej i jest wyświetlana jako liczba binarna.
- **c** - Odwołuje się do liczby całkowitej i jest wyświetlana jako odpowiadający jej znak ASCII.
- **d** - Odwołuje się do liczby całkowitej i jest wyświetlana jako liczba dziesiętna.
- **e** - Odwołuje się do notacji naukowej (np. 2.12e + 3).
- **E** - synonim „e”.
- **f** - Odwołuje się do zmiennej zmiennoprzecinkowej i jest wyświetlana jako liczba rzeczywista (z uwzględnieniem ustawień regionalnych).
- **F** - Odwołuje się do zmiennej zmiennoprzecinkowej i jest wyświetlana jako liczba rzeczywista (bez rozpoznawania ustawień regionalnych).
- **o** - Odwołuje się do liczby całkowitej i jest wyświetlana jako liczba ósemkowa.
- **s** - Jest traktowane i wyświetlane jako ciąg tekstowy.
- **u** - Odwołuje się do liczby całkowitej i jest wyświetlana jako liczba dziesiętna bez znaku.
- **x** - Odwołuje się do liczby całkowitej i jest wyświetlana jako liczba szesnastkowa (z małymi literami).
- **X** - Odwołuje się do liczby całkowitej i jest wyświetlana jako liczba szesnastkowa (z wielkimi literami).

Ponadto do nich można użyć następującego formatowania:

- Specyfikator znaku może służyć do wymuszonego wyświetlania znaku (- lub +), który ma być użyty na liczbie. Domyślnie tylko znak - jest wyświetlany na liczbach ujemnych. Używając tego specyfikatora, liczby dodatnie są wyświetlane z poprzedzającym znakiem +. Można to osiągnąć za pomocą symbolu + i można to zaimplementować tylko na wartościach liczbowych.

```
%+d
```

- Specyfikator dopełnienia może służyć do określenia, jaki znak zostanie użyty do wypełnienia wyników do dowolnego zdefiniowanego rozmiaru ciągu. Domyślnie jako wypełnienie używane są spacje. Alternatywny znak wypełniający można określić, poprzedzając go pojedynczym cudzysłowem lub '.

```
%'0d
```

- Specyfikator wyrównania może służyć do określenia wyrównania wyniku, tj. Czy wyrównanie do lewej czy do prawej. Domyślnie jest wyjustowany do prawej. Użycie znaku - powoduje wyrównanie do lewej.

```
%-s
```

- Specyfikator szerokości może służyć do określenia minimalnej liczby znaków, które mają być obecne w samym wyniku. Można ją określić za pomocą dowolnej liczby oznaczającej minimalną szerokość. Jest najczęściej używany z specyfikatorem dopełnienia.

```
%'05d
```

- Specyfikator dokładności może służyć do określania dokładności podczas pracy z liczbami rzeczywistymi. Kropka lub '.', Po którym następuje opcjonalny ciąg cyfry dziesiętnej, który odnosi się do cyfr dziesiętnych wyświetlanych po przecinku. Używając tego specyfikatora w ciągu, określa on maksymalny limit znaków w ciągu.

```
.5f  
.2s
```

```
<?php  
$numValue = 5;  
$strValue = "GeeksForGeeks";  
printf("Signed Number: %+d\n", $numValue);  
printf("Padding and Width\n%'03d\n%'03d\n", $numValue, $numValue+10);  
printf("Precision: %.5f %.5s\n", $numValue, $strValue);  
printf("Percentage: %%%\n", $numValue);  
printf("Binary: %b Octal: %o Hexadecimal: %x\n", $numValue+10, $numValue+10, $numValue+10);  
printf("Character: %c\n", $numValue+60);  
printf("String: %s\n", $strValue);  
printf("RealNumber: %f\n", 1/$numValue);  
printf("Scientific Representation:%e\n", $numValue+100);  
?>
```

2.1 2.1 Teoria

Step 1

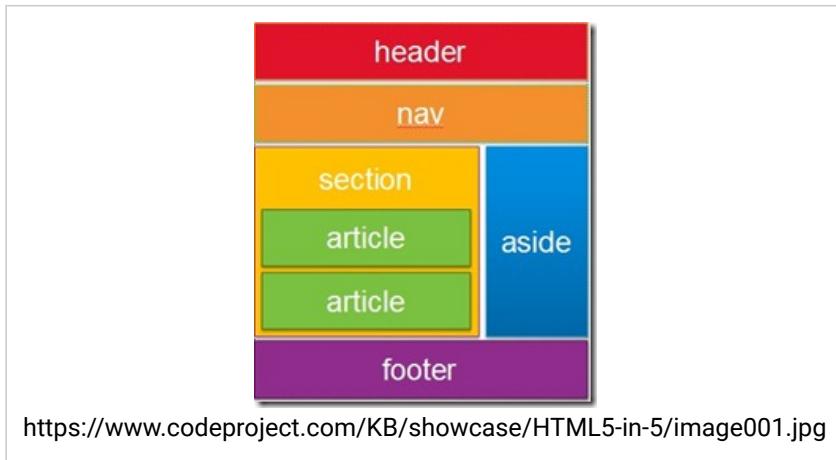
Język [HTML 4](https://www.kurshtml.edu.pl/html/html.html) (<https://www.kurshtml.edu.pl/html/html.html>) stał się oficjalną rekomendacją [W3C \(World Wide Web Consortium\)](#), w 1997 roku. Pierwsza wersja specyfikacji [XHTML 1](#) (<https://www.kurshtml.edu.pl/html/xhtml.html>) została ogłoszona w roku 2000. Celem [XHTML](#) 1.0 ani 1.1 nie było wprowadzanie nowych znaczników ani atrybutów, ale zbliżenie składni języka [HTML](#) do [XML \(Extensible Markup Language\)](#). Miało to umożliwić twórcom oraz administratorom serwisów internetowych korzystanie z narzędzi i bibliotek dostępnych dla języka [XML](#) oraz pozwolić na większą rozszerzalność [HTML](#) - poprzez możliwość osadzania w dokumencie [XHTML](#) fragmentów pochodzących z innych specyfikacji [XML](#) ([SVG \(Scalable Vector Graphics\)](#), [MathML \(Mathematical Markup Language\)](#), [RDF \(Resource Description Framework\)](#)). Jednak brak nowych możliwości języka, w ciągle rozwijającej się sieci, zaczął mocno doskwierać twórcom oraz producentom przeglądarek.

Problem jednak nadal pozostał. Dzisiaj Internet nie taki sam, jak w 1997 roku, kiedy dominowały statyczne dokumenty tekstowe, a multimedialność polegała na wstawieniu do artykułu kilku ilustracji. Dlatego producenci popularnych przeglądarek postanowili rozpocząć prace nad nową wersję znanego wszystkim języka znaczników - HTML tym razem już w odsłonie nr 5. Potem dla ostatecznego opracowania nowej specyfikacji powołano grupę roboczą W3C. Głównymi celami [HTML5](#) (<http://www.w3.org/TR/html5/>) są:

- Wprowadzenie nowych elementów dla zwiększenia interaktywności i multimedialności stron internetowych.
- Wprowadzenie nowych [znaczników semantycznych](#) (https://www.kurshtml.edu.pl/kod_poprawny_semantycznie,tekst.html), aby uczynić sieć bardziej dostępną dla wszystkich.
- Oficjalne załączenie do specyfikacji rozszerzeń, dodanych w przeszłości na własną rękę przez producentów przeglądarek, które i tak stały się już wcześniej *standardem de facto*.
- Bardziej szczegółowe określenie sposobu obsługi błędów, tak aby dokumenty napisane przez niedouczonych webmasterów wyświetlały się tak samo w każdej przeglądarce.
- Zachowanie kompatybilności wstecz, tak by użytkownicy starszych przeglądarek również mogli korzystać ze stron napisanych w nowym języku.

Obecny standard języka znaczników **HTML5** dodaje bardzo wiele znaczników, które mają na celu ułatwienie deweloperom przeznaczenie danego elementu na danej stronie internetowej. Wcześniej do wszystkich rzeczy używało się znacznika [`<div>`](#). Dodanymi najpopularniejszymi znacznikami w **HTML5** są:

- `article`
- `aside`
- `figcaption`
- `figure`
- `footer`
- `header`
- `hgroup`
- `mark`
- `nav`
- `section`
- `time`



Header odpowiada za nagłówek strony, natomiast **footer** za jego stopkę. **Nav** służy do tworzenia nawigacji po naszej stronie **sections** oraz **articles** służą do grupowania zawartości naszej strony internetowej.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<header>
    <hgroup>
        <h1>Header in h1</h1>
        <h2>Subheader in h2</h2>
    </hgroup>
</header>
<nav>
    <ul>
        <li><a href="#">Menu Option 1</a></li>
        <li><a href="#">Menu Option 2</a></li>
        <li><a href="#">Menu Option 3</a></li>
    </ul>
</nav>
<section>
    <article>
        <header>
            <h1>Article #1</h1>
        </header>
        <section>
            This is the first article. This is <mark>highlighted</mark>.
        </section>
    </article>
    <article>
        <header>
            <h1>Article #2</h1>
        </header>
        <section>
            This is the second article. These articles could be blog posts, etc.
        </section>
    </article>
</section>
<aside>
    <h1>Links</h1>
    <ul>
        <li><a href="#">Link 1</a></li>
        <li><a href="#">Link 2</a></li>
        <li><a href="#">Link 3</a></li>
    </ul>
    <figure>
        
        <figcaption>Mateusz Miotk</figcaption>
    </figure>
</aside>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

Jedną z największych zmian w HTML5 jest dodanie wsparcia dla plików audio oraz video. Jednakże trzeba tutaj pamiętać o sterownikach i kodękach obsługujących ten typ danych. Każda przeglądarka internetowa może używać inne rodzaje kodeków do odtwarzania multimedialów.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
    <header>
        <hgroup>
            <h1>Header in h1</h1>
            <h2>Subheader in h2</h2>
        </hgroup>
    </header>
    <nav>
        <ul>
            <li><a href="#">Menu Option 1</a></li>
            <li><a href="#">Menu Option 2</a></li>
            <li><a href="#">Menu Option 3</a></li>
        </ul>
    </nav>
    <section>
        <article>
            <header>
                <h1>Article #1</h1>
            </header>
            <section>
                This is the first article. This is <mark>highlighted</mark>.
            </section>
        </article>
        <article>
            <header>
                <h1>Article #2</h1>
            </header>
            <section>
                This is the second article. These articles could be blog posts, etc.
            </section>
        </article>
    </section>
    <aside>
        <section>
            <h1>Links</h1>
            <ul>
                <li><a href="#">Link 1</a></li>
                <li><a href="#">Link 2</a></li>
                <li><a href="#">Link 3</a></li>
            </ul>
        </section>
        <figure>
            
            <figcaption>Mateusz Miotk</figcaption>
        </figure>
    </aside>
    <audio controls="controls">
        <source src="http://greenmp3.pl/dzwonki/10236.mp3" type="audio/mp3" />
        Your browser does not support the audio element.
    </audio>
    <video width="320" height="240" controls="controls">
        <source src="http://dl5.webmfiles.org/big-buck-bunny_trailer.webm" type="video/webm" />
        Your browser does not support the video tag.
    </video>
    <footer>Footer - Copyright 2020</footer>
</body>
</html>

```

Step 2

Skrypty w języku PHP mają główne zastosowanie w serwisach WWW. Ogólny schemat pliku .html z użyciem skryptu w PHP wygląda następująco:

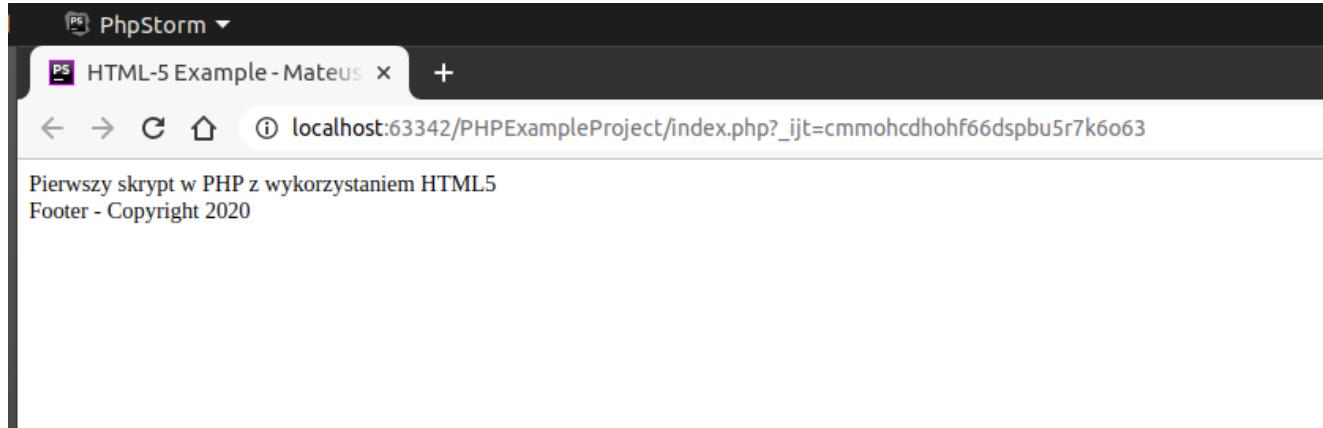
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Moja strona WWW</title>
</head>
<body>
  <p>
    <?php
      echo("Pierwszy skrypt w PHP z użyciem HTML");
    ?>
  </p>
</body>
</html>
```

Większa część kodu to standardowe znaczniki `HTML` tworzące strukturę najprostszej strony zgodnej ze standardem `XHTML`. W sekcji `<body>` wewnątrz znacznika `<p>` jest umieszczona treść skryptu PHP. Znajduje się ona między znacznikami `<?php` oraz `?>`. Wszystko co znajduje się między nimi, stanowi kod PHP i jest przetwarzane przez aparat wykonawczy PHP. Zobaczmy teraz, jak działa taki skrypt po wczytaniu do przeglądarki. Całą teść kodu zapisujemy w pliku o nazwie `index.php`, po czym umieszczać go w katalogu `public_html`. Odwołujemy się następnie do pliku `index.php`, wpisując na pasku adresu: `http://szuflandia.pjwstk.edu.pl/~nazwa_konta/`

Przy użyciu HTML5 wyglądałoby to następująco:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<?php
  echo "Pierwszy skrypt w PHP z wykorzystaniem HTML5";
?>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

Tak wygenerowana powyżej strona wyglądałaby następująco:



Natomiast kod źródłowy powyższej strony wygląda następująco:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
  Pierwszy skrypt w PHP z wykorzystaniem HTML5<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

Widać wyraźnie, że zniknęły wszystkie znaczniki PHP i pozostał jedynie czysty kod HTML. Taka jest bowiem istota działania języków skryptowych pracujących po stronie serwera. Zadaniem skryptu PHP jest wygenerowanie takiego kodu, który będzie mógł być zrozumiały dla przeglądarki.

Step 3

Szczególnym elementem bardzo często używanym na stronach www są formularze. Definicja formularza wygląda następująco:

```
<form>
    ciało formularza
</form>
```

Znacznik `<form>` może przyjmować również następujące parametry:

- `name` - określa nazwę formularza
- `action` - określa adres programu (skryptu), który ma przejąć dane z formularza
- `method = POST/GET` - sposób przesłania danych

Przykład formularza w `HTML5`:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<form name="formularz">
    To jest ciało formularza
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

Formularze przyjmują różne tzw. kontrolki, odpowiadające za obsługę danych. Do najpopularniejszych kontrolek należą:

- `<input type=text>` - Pole wprowadzania danych. Przyjmuje on też następujące parametry:

- `name` - nazwa pola
- `value` - wstępna wartość pola
- `size` - szerokość pola
- `maxlength` - maksymalna liczba znaków

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

FORMULARZ

Footer - Copyright 2020



- `<input type="radio">` - Pole jednokrotnego wyboru. Przyjmuje on też następujące parametry:

- `name` - nazwa pola
- `value` - wstępna wartość pola
- `checked` - określa czy przycisk jest zaznaczony

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <input type="radio" name="radio1" value="Przycisk nie zaznaczony">
    <input type="radio" name="radio2" checked value="Przycisk zaznaczony">
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

FORMULARZ

Footer - Copyright 2020



- `<input type="checkbox">` - Pole wielokrotnego wyboru. Przyjmuje on następujące parametry:

- `name` - nazwa pola
- `value` - wstępna wartość pola
- `checked` - określa czy przycisk jest zaznaczony

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <input type="radio" name="radio1" value="Przycisk nie zaznaczony">
    <input type="radio" name="radio2" checked value="Przycisk zaznaczony"> <br/>
    <input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
    <input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

localhost:63342/PHPExampleProject/index.php?_

FORMULARZ

Footer - Copyright 2020

- `<input type="submit">` - Przycisk wysyłania danych. Przyjmuje on następujące parametry:
 - `name` - nazwa pola
 - `value` - wyświetlna wartość w przycisku

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <input type="radio" name="radio1" value="Przycisk nie zaznaczony">
    <input type="radio" name="radio2" checked value="Przycisk zaznaczony"> <br/>
    <input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
    <input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    <input type="submit" name="sub1" value="Prześlij formularz">
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

localhost:63342/PHPExampleProject/index.php?_

FORMULARZ

Prześlij formularz

Footer - Copyright 2020

- <input type="reset"> - Przycisk kasowania danych z formularza (przywracania ich do ustawień domyślnych). Przyjmuje on następujące parametry:

- name - nazwa pola
- value - wyświetlana wartość w przycisku

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <input type="radio" name="radio1" value="Przycisk nie zaznaczony">
    <input type="radio" name="radio2" checked value="Przycisk zaznaczony"> <br/>
    <input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
    <input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    <input type="submit" name="sub1" value="Prześlij formularz">
    <input type="reset" name="res1" value="Zresetuj formularz">
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

localhost:63342/PHPExampleProject/index.php?_

FORMULARZ

Wypełnij mnie max 15 znaków

 Footer - Copyright 2020

- <input type="button"> - Przycisk polecenia. Przyjmuje on następujące parametry:

- name - nazwa pola
- value - Wyświetlana wartość

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <input type="radio" name="radio1" value="Przycisk nie zaznaczony">
    <input type="radio" name="radio2" checked value="Przycisk zaznaczony"> <br/>
    <input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
    <input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    <input type="submit" name="sub1" value="Prześlij formularz">
    <input type="reset" name="res1" value="Zresetuj formularz"><br/>
    <input type="button" name="but1" value="Kolejny przycisk co nic nie robi :-)">
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

FORMULARZ

Footer - Copyright 2020



- <textarea> zawartość </textarea> - Pole wprowadzania tekstów. Przyjmuje on następujące parametry:

- `name` - nazwa pola
- `cols` - ilość kolumn
- `rows` - ilość znaków w wierszu
- `wrap=off/soft/hard` - sposób przejścia do następnej linii

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <input type="radio" name="radio1" value="Przycisk nie zaznaczony">
    <input type="radio" name="radio2" checked="" value="Przycisk zaznaczony"> <br/>
    <input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
    <input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    <input type="submit" name="sub1" value="Prześlij formularz">
    <input type="reset" name="res1" value="Zresetuj formularz"><br/>
    <input type="button" name="but1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    <textarea name="text1" cols="2" rows="10" wrap="hard">Tutaj wpisz tekst</textarea>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

FORMULARZ

Tuta
j
wpis
z
tekst
t

Footer - Copyright 2020

- <select>opcje</select> - Pole listy rozwijanej. Przyjmuje ona następujące parametry:

- `name` - nazwa pola

- `size` - liczba wierszy
- `multiple` - możliwość wielokrotnego wyboru
- Opcje w poleceniu `select` definiujemy za pomocą znacznika `<option>tekst wyświetlny</option>`. Przyjmuje on następujące parametry:
 - `value` - wartość opcji
 - `selected` - opcja wybrana

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h5>FORMULARZ</h5>
<form name="formularz">
  <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
  <input type="radio" name="radio1" value="Przycisk nie zaznaczony">
  <input type="radio" name="radio2" checked value="Przycisk zaznaczony"> <br/>
  <input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
  <input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
  <input type="submit" name="sub1" value="Prześlij formularz">
  <input type="reset" name="res1" value="Zresetuj formularz"><br/>
  <input type="button" name="butt1" value="Kolejny przycisk co nic nie robi :-)"><br/>
  <textarea name="text1" cols="2" rows="10" wrap="hard">Tutaj wpisz tekst</textarea>
<br/>
  <select name="nationality" size="2">
    <option value="PL" selected>Polska</option>
    <option value="DE">Niemcy</option>
    <option value="GB">Wielka Brytania</option>
  </select>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

localhost:63342/PHPExampleProject/Index.php?_ijt=a9ioZm4

FORMULARZ

Wypełnij mnie max 15 znaków

Prześlij formularz Zresetuj formularz

Kolejny przycisk co nic nie robi :-)

Tutaj wpisz tekst

Polska
Niemcy

Footer - Copyright 2020

Step 4

Znacznik `<fieldset>` pozwala zgrupować tematycznie kilka pól formularza, dzięki czemu zostają one objęte ramką.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Radiobuttony</legend>
        <label>TAK: </label><input type="radio" name="radio1" value="Przycisk nie zaznaczony">
        <label>NIE: </label><input type="radio" name="radio1" checked value="Przycisk zaznaczony"> <br/>
    </fieldset>
    <fieldset>
        <legend>Checkboxy</legend>
        <label>Kawa: </label><input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
        <label>Herbata: </label><input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="butt1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

localhost:63342/PHPExampleProject/Index.php?input=Wype

FORMULARZ z użyciem fieldset

Wypełnij mnie max 15 znaków

Radiobuttony

TAK: NIE:

Checkboxy

Kawa: Herbata:

Footer - Copyright 2020

HTML5 wprowadził nowe kontrolki do formularza są to między innymi:

- <input type="email"> - Pole email. Pole, które służy do wprowadzania adresu e-mail. Przeglądarka przy wprowadzaniu maila sprawdzi, czy wygląda on na poprawny (musi zawierać znak @, nie może kończyć, ani zaczynać się kropką itp.). **Uwaga:** Może przechwytywać niepoprawne adresy email, dlatego z uwagą należy podchodzić do validacji tego pola.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Radiobuttony</legend>
        <label>TAK: </label><input type="radio" name="radio1" value="Przycisk nie zaznaczony">
        <label>NIE: </label><input type="radio" name="radio1" checked value="Przycisk zaznaczony">
    <br/>
    </fieldset>
    <fieldset>
        <legend>Checkboxy</legend>
        <label>Kawa: </label><input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
        <label>Herbata: </label><input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    </fieldset>
    <fieldset>
        <legend>Pola HTML5</legend>
        <label>Podaj email: </label><input type="email" name="mail" value="Podaj email">
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="but1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

← → ⌛ ⓘ localhost:63342/PHPExampleProject/index.php?_Jt=bo3tb6c

FORMULARZ z użyciem fieldset

Radiobuttony

TAK: NIE:

Checkboxy

Kawa: Herbata:

Pola HTML5

Podaj email:

Footer - Copyright 2020



- <input type="number"> - Pole number. Wymusza na użytkowniku wprowadzenie liczby. Dozwolony zakres liczb można określić za pomocą atrybutów max i min.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Radiobuttony</legend>
        <label>TAK: </label><input type="radio" name="radio1" value="Przycisk nie zaznaczony">
        <label>NIE: </label><input type="radio" name="radio1" checked value="Przycisk zaznaczony">
    <br/>
    </fieldset>
    <fieldset>
        <legend>Checkboxy</legend>
        <label>Kawa: </label><input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
        <label>Herbata: </label><input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    </fieldset>
    <fieldset>
        <legend>Pola HTML5</legend>
        <label>Podaj email: </label><input type="email" name="mail" value="Podaj email">
        <label>Podaj liczbę z zakresu od 1 do 10: </label><input type="number" value="1" min="1" max="10">
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="but1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

← → ⌂ ⓘ localhost:63342/PHPExampleProject/index.php?input=Wype

FORMULARZ z użyciem fieldset

Wypełnij mnie max 15 znaków

Radiobuttony

TAK: NIE:

Checkboxy

Kawa: Herbata:

Pola HTML5

Podaj email: Podaj liczbę z zakresu od 1 do 10:

Footer - Copyright 2020

- <input type="date"> - Pole data. Pole służące do wprowadzania daty. Również bardzo przydatne, chociażby podczas podawania daty urodzenia.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Radiobuttony</legend>
        <label>TAK: </label><input type="radio" name="radio1" value="Przycisk nie zaznaczony">
        <label>NIE: </label><input type="radio" name="radio1" checked value="Przycisk zaznaczony">
    <br/>
    </fieldset>
    <fieldset>
        <legend>Checkboxy</legend>
        <label>Kawa: </label><input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
        <label>Herbata: </label><input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    </fieldset>
    <fieldset>
        <legend>Pola HTML5</legend>
        <label>Podaj email: </label><input type="email" name="mail" value="Podaj email"><br/>
        <label>Podaj liczbę z zakresu od 1 do 10: </label><input type="number" value="1" min="1" max="10"><br/>
        <label>Data urodzenia: </label><input type="date" name="birth">
    </fieldset>
    <input type="submit" name="sub1" value="Prześlij formularz">
    <input type="reset" name="res1" value="Zresetuj formularz">
    <input type="button" name="butt1" value="Kolejny przycisk co nic nie robi :-)"><br/>
</fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

← → ⌂ ⓘ localhost:63342/PHPExampleProject/index.php?input=Wype

FORMULARZ z użyciem fieldset

Wypełnij mnie max 15 znaków

Radiobuttony

TAK: NIE:

Checkboxy

Kawa: Herbata:

Pola HTML5

Podaj email:

Podaj liczbę z zakresu od 1 do 10:

Data urodzenia:

Footer - Copyright 2020

- <input type="time"> - Pole time. Służy do podawania godziny.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Radiobuttony</legend>
        <label>TAK: </label><input type="radio" name="radio1" value="Przycisk nie zaznaczony">
        <label>NIE: </label><input type="radio" name="radio1" checked value="Przycisk zaznaczony">
    <br/>
    </fieldset>
    <fieldset>
        <legend>Checkboxy</legend>
        <label>Kawa: </label><input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
        <label>Herbata: </label><input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    </fieldset>
    <fieldset>
        <legend>Pola HTML5</legend>
        <label>Podaj email: </label><input type="email" name="mail" value="Podaj email"><br/>
        <label>Podaj liczbę z zakresu od 1 do 10: </label><input type="number" value="1" min="1" max="10"><br/>
        <label>Data urodzenia: </label><input type="date" name="birth"><br/>
        <label>Czas zatrudnienia: </label><input type="time" name="timeEmployee">
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="but1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

← → ⌂ ⌃ ⓘ localhost:63342/PHPExampleProject/index.php?input=Wypełnij+mnie+max+15+znaków

FORMULARZ z użyciem fieldset

Wypełnij mnie max 15 znaków

Radiobuttony

TAK: NIE:

Checkboxy

Kawa: Herbata:

Pola HTML5

Podaj email:

Podaj liczbę z zakresu od 1 do 10:

Data urodzenia:

Czas zatrudnienia:

Footer - Copyright 2020

- <input type="color"> - Pole color. Pole służące do wybierania koloru. W przeglądarkach, po użyciu tego pola zobaczymy klasyczny systemowy widget do wyboru koloru.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Radiobuttony</legend>
        <label>TAK: </label><input type="radio" name="radio1" value="Przycisk nie zaznaczony">
        <label>NIE: </label><input type="radio" name="radio1" checked value="Przycisk zaznaczony">
    <br/>
    </fieldset>
    <fieldset>
        <legend>Checkboxy</legend>
        <label>Kawa: </label><input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
        <label>Herbata: </label><input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    </fieldset>
    <fieldset>
        <legend>Pola HTML5</legend>
        <label>Podaj email: </label><input type="email" name="mail" value="Podaj email"><br/>
        <label>Podaj liczbę z zakresu od 1 do 10: </label><input type="number" value="1" min="1" max="10"><br/>
        <label>Data urodzenia: </label><input type="date" name="birth"><br/>
        <label>Czas zatrudnienia: </label><input type="time" name="timeEmployee"><br/>
        <label>Kolor: </label><input type="color" name="col">
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="butt1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

The screenshot shows a web page with the following structure:

- Header:** <title>HTML-5 Example - Mateusz Miotk</title>
- Section:** <h3>FORMULARZ z użyciem fieldset</h3>
- Form:** <form name="formularz">

 - Text Input:** <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
 - Fieldset 1 (Radiobuttons):** <fieldset> with legend "Radiobuttony". It contains two radio buttons: "TAK" (unchecked) and "NIE" (checked).
 - Fieldset 2 (Checkboxes):** <fieldset> with legend "Checkboxy". It contains two checkbox inputs: "Kawa" (unchecked) and "Herbata" (checked).
 - Fieldset 3 (HTML5 Fields):** <fieldset> with legend "Pola HTML5". It contains four inputs:
 - <label>Podaj email:</label> <input type="email" name="mail" value="Podaj email">
 - <label>Podaj liczbę z zakresu od 1 do 10:</label> <input type="number" value="1" min="1" max="10">
 - <label>Data urodzenia:</label> <input type="date" name="birth">
 - <label>Czas zatrudnienia:</label> <input type="time" name="timeEmployee">
 - Buttons:** <input type="submit" name="sub1" value="Prześlij formularz">, <input type="reset" name="res1" value="Zresetuj formularz">, <input type="button" name="butt1" value="Kolejny przycisk co nic nie robi :-)">

- Footer:** <footer>Footer - Copyright 2020</footer>

- <input type="range"> - Pole range. Pole, które służy do wybierania liczby z podanego zakresu. Wybór odbywa się za pomocą suwaka, a zakres, podobnie jak w polu do wyboru liczby, możemy ustalić za pomocą atrybutów min i max.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Radiobuttony</legend>
        <label>TAK: </label><input type="radio" name="radio1" value="Przycisk nie zaznaczony">
        <label>NIE: </label><input type="radio" name="radio1" checked value="Przycisk zaznaczony">
    <br/>
    </fieldset>
    <fieldset>
        <legend>Checkboxy</legend>
        <label>Kawa: </label><input type="checkbox" name="check1" value="Przycisk check niezaznaczony">
        <label>Herbata: </label><input type="checkbox" name="check2" value="Przycisk check zaznaczony" checked><br/>
    </fieldset>
    <fieldset>
        <legend>Pola HTML5</legend>
        <label>Podaj email: </label><input type="email" name="mail" value="Podaj email"><br/>
        <label>Podaj liczbę z zakresu od 1 do 10: </label><input type="number" value="1" min="1" max="10"><br/>
        <label>Data urodzenia: </label><input type="date" name="birth"><br/>
        <label>Czas zatrudnienia: </label><input type="time" name="timeEmployee"><br/>
        <label>Kolor: </label><input type="color" name="col"><br/>
        <label>Zakres: </label><input type="range" min="1" max="10" value="5">
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="but1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

← → ⌛ ⓘ localhost:63342/PHPExampleProject/index.php?_ijt=h3jq0pb3q2rr/vm5l2immtddq

FORMULARZ z użyciem fieldset

Wypełnij mnie max 15 znaków

Radiobuttony

TAK: NIE:

Checkboxy

Kawa: Herbata:

Pola HTML5

Podaj email:

Podaj liczbę z zakresu od 1 do 10:

Data urodzenia:

Czas zatrudnienia:

Kolor:

Zakres:

Footer - Copyright 2020

Do tej pory nie mieliśmy praktycznie żadnych narzędzi dostępnych w języku HTML, których mogliśmy użyć, aby sprawdzić poprawność wprowadzonych w formularzu danych. Najbardziej przydatnym nowym narzędziem w walce o dobrze wypełniony formularz jest atrybut `required`. Dzięki niemu możemy wymusić na użytkowniku wypełnienie pola. Tworząc więc pole `<input type="email" required>` upewniamy się, że użytkownik poda nam adres e-mail. To samo dotyczy się pozostałych pól.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Wymagane dane</legend>
        <label>Imię: </label><input type="text" required minlength="2">
        <label>Adres e-mail</label><input type="email" required>
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="butt1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

FORMULARZ z użyciem fieldset

Wypełnij mnie max 15 znaków

Wymagane dane

Imię: _____ Adres e-mail: _____

Prześlij formularz Zresetuj formularz Kolejny przycisk co nic nie robi :-)

Footer - Copyright 2020

Nowy element `datalist` daje nam możliwość stworzenia podpowiedzi do wypełnianego pola. Dzięki temu możemy łatwo zasugerować użytkownikowi jedną z najczęściej wybieranych opcji.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form name="formularz">
    <input type="text" name="input" value="Wypełnij mnie max 15 znaków" maxlength="15">
    <fieldset>
        <legend>Wymagane dane</legend>
        <label>Imię: </label><input type="text" required minlength="2">
        <label>Adres e-mail</label><input type="email" required>
        <label>Stan cywilny: </label><input type="text" required list="stany">
        <datalist id="stany">
            <option value="Zamężny/Zamężna"></option>
            <option value="Kawaler/Panna"></option>
            <option value="Rozwodnik"></option>
        </datalist>
    </fieldset>
    <fieldset>
        <input type="submit" name="sub1" value="Prześlij formularz">
        <input type="reset" name="res1" value="Zresetuj formularz">
        <input type="button" name="butt1" value="Kolejny przycisk co nic nie robi :-)"><br/>
    </fieldset>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

localhost:63342/PHPExampleProject/index.php?input=Wypełnij+mnie+max+15+znaków&su

FORMULARZ z użyciem fieldset

Wypełnij mnie max 15 znaków

Wymagane dane

Imię: Adres e-mail: Stan cywilny:

Footer - Copyright 2020

Więcej informacji o formularzach znajdziemy pod adresem: https://www.w3schools.com/html/html_forms.asp (https://www.w3schools.com/html/html_forms.asp).

Step 6

Formularz `html` definiujemy stosując element `form`. Wewnątrz, po między znacznikami `<form>` oraz `</form>` umieszczamy zawartość formularza, na którą składają się kontrolki (np. `<input>`) oraz elementy formatujące (np. `<table>`). Typowy formularz składa się z elementu `form` zawierającego tabelę, wewnątrz której umieszczone kilka kontrolek. Ważne jest pole `action`, w którym zapisujemy skrypt, jaki ma być uruchomiony podczas przesyłania formularza.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form action="jakis-skrypt.php">
    <table>
        <tr>
            <td>Imię:</td>
            <td><input name="imie"></td>
        </tr>
        <tr>
            <td>Nazwisko:</td>
            <td><input name="nazwisko"></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="submit" value="Wyślij"></td>
        </tr>
    </table>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

FORMULARZ z użyciem fieldset

Imię:

Nazwisko:

Footer - Copyright 2020

Powyższy formularz zawiera dwa pola do wprowadzania danych (pola te nazywano Imię i Nazwisko) oraz przycisk Wyślij. Osoba odwiedzająca witrynę może umieścić cursor wewnątrz pól formularza, wypełnić je, wpisując napisy Mateusz Miotk, po czym przesłać formularz, naciskając przycisk Wyślij. Treść wprowadzona przez użytkownika zostanie przesłana do skryptu o nazwie `jakis-skrypt.php`. Nazwę skryptu przetwarzającego formularz podajemy jako wartość atrybutu `action` elementu `form`. Po naciśnięciu przycisku Wyślij, wizyta zostanie przeniesiona pod adres `jakis-skrypt.php`. W skrypcie tym będą dostępne dane wprowadzone przez użytkownika w formularzu. Skrypt przetwarzający formularz zawarty w pliku `jakis-skrypt.php` może być napisany w dowolnym języku programowania dynamicznych stron WWW. Może to być PHP, Perl, ASP, JavaServerPages, skrypty CGI w bashu, C, czy nawet Pascalu. Jednakże trzeba od początku jasno podkreślić, że nie ma możliwości przetworzenia formularza w języku HTML. Do przetwarzania formularza musimy użyć jednego z języków skryptowych, służących do programowania dynamicznych stron WWW. Formularze tworzymy w języku HTML, stosując między innymi elementy `form` oraz `input`. Natomiast przetwarzanie formularza wykonuje skrypt napisany na przykład w jednym z języków PHP, ASP lub Perl i umieszczony na serwerze. Zatem korzystanie z formularzy wymaga znajomości zarówno języka HTML jak i języka skryptowego. Język HTML zajmuje się jedynie wyglądem zewnętrznym formularza. Stosując elementy HTML układamy zawartość formularza na stronie oraz ustalamy adres URL skryptu, który będzie zajmował się przetworzeniem danych pochodzących z formularza. Jeśli formularz przedstawiony powyżej zapiszemy do pliku `formularz.html`, wówczas cały przykład będzie się składał z dwóch plików. Pierwszym plikiem jest plik `formularz.html` zawierający kod HTML formularza, zaś drugim plikiem będzie `jakis-skrypt.php`. Pamiętajmy, że nazwa pliku zawierającego skrypt musi być dokładnie taka, jak wartość atrybutu `action` formularza.

Formularz jest przedstawiany w oknie przeglądarki w postaci szeregu kontrolek. Układ graficzny kontrolek nie wpływa na sposób zakodowania danych. Dane wprowadzone do formularza są kodowane przez przeglądarkę. O sposobie kodowania decyduje atrybut `enctype` elementu `form`. Domyślnym kodowaniem formularzy jest `application/x-www-form-urlencoded`. Kodowanie to polega na utworzeniu par postaci: `nazwakontrolki=wartosc` i połączeniu ich separatorem `&`. Wszystkie znaki specjalne występujące w nazwach lub wartościach kontrolek zostają przedstawione w postaci kodu szesnastkowego poprzedzonego znakiem procentu. Na przykład spacja jest zamieniana na napis `%20` (kod ASCII znaku spacja - w systemie dziesiętnym - jest równy 32; liczba 32 w systemie

szesnastkowym wynosi `20 HEX`). Nazwy zmiennych są pobierane z kodu HTML formularza. Każda kontrolka posiada atrybut name. Atrybut ten ustala nazwę zmiennej. W formularzu przedstawionym powyżej występują dwie kontrolki o nazwach `imie` oraz `nazwisko`. Po wprowadzeniu do formularza danych Mateusz Miotk, otrzymamy zakodowany napis: `imie=Mateusz&nazwisko=Miotk`. Pierwszy krok interakcji użytkownika z aplikacją internetową polega na wprowadzeniu danych do formularza. Następnie, po naciśnięciu przycisku Wyślij, przeglądarka koduje wprowadzone przez użytkownika dane, po czym wysyła odpowiednie zapytanie.

Wszystkie transakcje WWW - a zatem także wysyłanie zawartości formularza - są realizowane przy użyciu protokołu HTTP. Protokół ten definiuje cztery metody przekazywania danych. Metodami tymi są `POST`, `GET`, `DELETE` oraz `PUT`. W stosunku do formularzy zastosowanie znajdują dwie spośród nich: `GET` oraz `POST`. W metodzie `GET` dane są dołączone do adresu URL i przyjmują postać:

```
http://gdzies.w.sieci/kat/strona.php?imie=Jan&nazwisko=Nowak&plec=M&wiek=35
```

Natomiast w metodzie `POST` dane z formularza są dołączone na końcu zapytania HTTP (za wszystkimi nagłówkami). Metodę przekazywania danych formularza ustalamy atrybutem `method` elementu `form`. Ponieważ wartością domyślną jest `GET`, zatem formularz:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form action="jakis-skrypt.php">
    <table>
        <tr>
            <td>Imię:</td>
            <td><input name="imie"></td>
        </tr>
        <tr>
            <td>Nazwisko:</td>
            <td><input name="nazwisko"></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="submit" value="Wyślij"></td>
        </tr>
    </table>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

Jest równoważny z formularzem:

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form action="jakis-skrypt.php" method="GET">
    <table>
        <tr>
            <td>Imię:</td>
            <td><input name="imie"></td>
        </tr>
        <tr>
            <td>Nazwisko:</td>
            <td><input name="nazwisko"></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="submit" value="Wyślij"></td>
        </tr>
    </table>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

Natomiast formularz przekazywany metodą POST wygląda następująco:

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h3>FORMULARZ z użyciem fieldset</h3>
<form action="jakis-skrypt.php" method="post">
    <table>
        <tr>
            <td>Imię:</td>
            <td><input name="imie"></td>
        </tr>
        <tr>
            <td>Nazwisko:</td>
            <td><input name="nazwisko"></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="submit" value="Wyślij"></td>
        </tr>
    </table>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>

```

W zależności od użytej metody, dane pochodzące z formularza odbieramy na różne sposoby wewnątrz skryptu przetwarzającego formularz.

Step 7

Tablice `$_GET`, `$_POST` oraz `$_REQUEST` zawierające przetworzone dane pochodzące z formularza i dostępne wewnątrz skryptu php są zmiennymi **superglobalnymi**. Oznacza to, że są one widoczne wewnątrz wszystkich funkcji i metod bez konieczności stosowania słowa kluczowego `global`. Tablica `$_GET` zawiera dane przekazane do skryptu metodą `GET`. Tablica `$_POST` zawiera dane przekazane do skryptu metodą `POST`. Natomiast tablica `$_REQUEST` zawiera dane pochodzące z ciasteczek, sesji, oraz przekazane metodami `POST` lub `GET`. Dane pochodzące z formularzy przekazywanych metodą `GET` są dostępne w skrypcie php w tablicy `$_GET`. Jeśli formularz jest przekazany metodą `POST`, to należy użyć tablicy `$_POST`. Wszystkie trzy wymienione

tablice są **tablicami asocjacyjnymi**. Indeksem w powyższych tablicach może być napis. Jakiego indeksu powinniśmy użyć w celu odczytania imienia i nazwiska pochodzących z formularza z poprzedniego slajdu? Napis wprowadzony w polu zatytułowanym **Imię** jest dostępny pod indeksem **imie**, zaś **nazwisko** - pod indeksem **nazwisko**. Indeksy **imie** i **nazwisko** są wartościami atrybutu **name** kontrolek **input**. Jeśli użyto metody **\$_GET**, wówczas imię i nazwisko podane przez internautę w formularzu są dostępne jako:

```
$_GET['imie']
$_GET['nazwisko']
```

Jeśli użyto metody **\$_POST**, to należy użyć:

```
$_POST['imie']
$_POST['nazwisko']
```

Zatem wybór metody przekazywania danych z formularza do skryptu php wpływa na wybór tablicy superglobalnej, z której skrypt będzie pobierał dane. Jeśli stosujemy metodę POST to należy korzystać z tablicy **\$_POST**. Korzystając z metody **GET** dane pobieramy z tablicy

\$_GET. Wiedząc, że są to tablice możemy po prostu już na nich operować ze znanych wcześniej funkcji na przykład **print_r**.

Wszystkie informacje na temat danych pochodzących z formularza i dostępnych wewnątrz skryptu zwraca funkcja **phpinfo()**. Jeśli w skrypcie **jakis-skrypt.php** przetwarzającym formularz z poprzedniego slajdu umieścimy kod:

```
<?php
    phpinfo();
?>
```

To wówczas funkcja **phpinfo()** wyświetli listę wszystkich zmiennych przekazanych z formularza do skryptu. Ponadto wyświetli wiele informacji na temat używanej wersji PHP.

Drugim sposobem sprawdzenia danych przekazanych do skryptu jest użycie jednej z funkcji **var_dump()**, **var_export()** oraz **print_r()**. Możemy również, stosując funkcję **array_keys()**, odczytać wszystkie indeksy tablicy **\$_GET**, po czym w pętli **foreach** wydrukować kolejno wszystkie elementy tablicy:

```
<?php
var_dump($_GET);
echo "\n";
$keys = array_keys($_GET);
foreach ($keys as $key){
    echo "\$_GET['$key'] == {$_GET[$key]}<BR>";
}
?>
```

localhost:63342/PHPExampleProject/index.php?_ijt=4mnp6p6v6o83kc61rtjie04rke

FORMULARZ z użyciem fieldset

Imię:

Nazwisko:

Footer - Copyright 2020

W analogiczny sposób możemy oczywiście użyć powyższych rozwiązań do wyświetlenia zawartości tablicy `$_POST`. Dodajmy jeszcze, że oprócz tablic `$_GET`, `$_POST`, `$_REQUEST` w skrypcie jest dostępna również tablica `$_SERVER`, która zawiera szczegółowe informacje na temat zapytania HTTP. Zmienne: `$_SERVER["REQUEST_METHOD"]` oraz `$_SERVER["REQUEST_URI"]` zawierają informacje na temat metody zapytania HTTP oraz żądanego dokumentu.

```
<?php
var_dump($_GET);
echo "\n";
$keys = array_keys($_GET);
foreach ($keys as $key){
    echo "\$_GET['$key'] == {$_GET[$key]}<br/>";
}
echo "<br/>";
echo $_SERVER['REQUEST_METHOD'] . "<br/>";
echo $_SERVER['REQUEST_URI'];
?>
```

localhost:63342/PHPExampleProject/index.php?_ijt=4mnp6p6v6o83kc61rtjie04rke

FORMULARZ z użyciem fieldset

Imię:

Nazwisko:

Footer - Copyright 2020



Step 8

Droga, jaką odbywają dane wprowadzone do formularza jest następująca:

- użytkownik wypełnia formularz, po czym naciska przycisk Wyślij,
- przeglądarka koduje informacje zawarte w formularzu, a następnie wysyła zapytanie HTTP do serwera,
- oprogramowanie działające na serwerze odbiera zapytanie HTTP,

- zapytanie jest przekazywane przez kolejne warstwy oprogramowania: stos protokołów TCP/IP, przekazuje zapytanie do procesu Apache, Apache uruchamia maszynę PHP i przekazuje jej zapytanie, zaś maszyna PHP przetwarza zapytanie, uruchamia skrypt i przekazuje do skryptu tablice `$_GET`, `$_POST`, itd.
- skrypt przetwarza dane, produkuje wynikowy kod HTML,
- kod zostaje wysłany w odpowiedzi HTTP do przeglądarki.

Jeśli porównamy zapytania wysłane metodą GET i POST, to zauważymy następujące różnice:

- w przypadku metody GET:
 - dane zapytania są zakodowane w adresie URL w postaci:
 - `jakis-skrypt.php?imie=Mateusz&nazwisko=Miotk`
 - adres wyświetlany przez przeglądarkę, zawiera informacje o przekazanych zmiennych;
 - stronę możemy odświeżyć przyciskiem Odśwież.
- w przypadku metody POST:
 - o dane zapytania są dołączone za nagłówkami;
 - o dane nie są widoczne w polu adres przeglądarki;
 - o odświeżanie strony powoduje wyświetlenie komunikatu.
 - Pamiętając o tym, by - ze względów bezpieczeństwa - stosować wyłącznie metodę POST,

Poniżej znajduje się kod formularza, który przetwarza kalkulator.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>HTML-5 Example - Mateusz Miotk</title>
</head>
<body>
<h1>Formularz - kalkulator</h1>
<form action="kalkulator-skrypt.php" method="POST">
    <table>
        <tr>
            <td>Pierwsza liczba:</td>
            <td><input name="liczba1"></td>
        </tr>
        <tr>
            <td>Druga liczba:</td>
            <td><input name="liczba2"></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="submit" value="Wyślij"></td>
        </tr>
    </table>
</form>
<footer>Footer - Copyright 2020</footer>
</body>
</html>
```

```
<?php
if (isset($_POST['liczba1']) && isset($_POST['liczba2'])) {
    if (is_numeric($_POST['liczba1']) && is_numeric($_POST['liczba2'])) {
        echo "W formularzu podano liczby {$_POST['liczba1']} oraz
{$_POST['liczba2']}.<br>";
        echo "Wyniki działań:<br>";
        echo "{$_POST['liczba1']} + {$_POST['liczba2']} = ";
        echo $_POST['liczba1'] + $_POST['liczba2'];
        echo "<br>";
    } else {
        echo "Błędne dane! Jedna lub obie liczby są niepoprawne!<br>";
    } else {
        echo "Brak danych! Jedna lub obie liczby nie zostały podane!<br>";
    }
?>
```

3.1 3.1 Teoria

Step 1

Podstawową funkcją pozwalającą na uzyskanie informacji o dacie jest `date`. Jej wywołanie ma postać:

```
date(format[ , znacznik_czasu])
```

Zwraca ona ciąg znaków we wskazanym formacie. Parametr `format` określa, jakie dane nas interesują, natomiast `znacznik_czasu` to argument opcjonalny zawierający znacznik czasu Uniksa i określający interesującą nas datę.

Znaczniki formatujące dla funkcji `date`:

- `d` - Dzień miesiąca, 2 cyfry z wiodącymi zerami
- `D` - Tekstowy opis angielskiej nazwy dnia, trzy litery
- `j` - Dzień miesiąca bez zer wiodących
- `\l` (mała litera 'L') - Pełen angielski opis dnia tygodnia
- `N` - Liczbowa forma dnia tygodnia, zgodna z normą ISO-8601 (dodana w PHP 5.1.0)
- `S` - Angielski przyrostek porządkowy dla dnia miesiąca, 2 litery
- `w` - Liczbowa forma dnia tygodnia
- `z` - Dzień roku (Zaczynając od 0)
- `W` - Numer tygodnia w roku, zgodny z normą ISO-8601, Tygodnie rozpoczynają Poniedziałki (dostępne od PHP 4.1.0)
- `F` - Pełen angielski opis, dnia miesiąca, taki jak January czy March
- `m` - Liczbowa forma miesiąca, z zerami wiodącymi
- `M` - Krótki, angielski opis miesiąca, trzy litery
- `n` - Liczbowa forma miesiąca, bez zer wiodących
- `t` - Ilość dni w danym miesiącu
- `L` - Informacja o tym, czy rok jest przestępny
- `o` - Numer roku, zgodny z normą ISO-8601. Zwraca to taką samą wartość jak Y, z takim wyjątkiem, że numer tygodnia ISO (W) należy do poprzedniego lub następnego roku, niż rok użyty w tym miejscu. (dodane w PHP 5.1.0)
- `Y` - Pełna liczbowa forma roku, 4 cyfry
- `y` - Dwie cyfry reprezentujące rok
- `a` - Pora dnia - dwie małe litery (przed/po południu) (ang. Ante/Post meridiem)
- `A` - Pora dnia - dwie duże litery (przed/po południu) (ang. Ante/Post meridiem)
- `g` - Godzina, w formacie 12-godzinnym, bez zer wiodących
- `G` - Godzina, w formacie 24-godzinnym, bez zer wiodących
- `h` - Godzina, w formacie 12-godzinnym, z zerami wiodącymi
- `H` - Godzina, w formacie 24-godzinnym, z zerami wiodącymi
- `i` - Minuty z zerami wiodącymi
- `s` - Sekundy, z zerami wiodącymi
- `e` - Identyfikator strefy czasowej (dodano w PHP 5.1.0)
- `I` (duże i) - Informacja o tym, czy czas jest letni
- `O` - Różnica z czasem Greenwich (GMT) w godzinach
- `P` - Różnica z czasem Greenwich (GMT) z dwukropkiem pomiędzy godzinami i minutami (dodano w PHP 5.1.3)
- `T` - Skrót dla strefy czasowej
- `Z` - Różnica dla strefy czasowej w sekundach. Wyrównanie to jest zawsze ujemne dla stref położonych na zachód od południka 0, oraz dodatnie dla tych leżących na wschód od niego.
- `c` - Data w standardzie ISO 8601 (dodana w PHP 5)
- `r` - Data sformatowana zgodnie z RFC 2822
- `U` - Sekundy liczone od ery UNIX-a (1 stycznia 1970 00:00:00 czasu Greenwich - GMT)

Przykład użycia funkcji `date()`:

```
<?php
    echo 'date(\"Y-m-d\") = ' . date("Y-m-d") . '<br \>';
    echo 'date(\"d-m-Y\") = ' . date("d-m-Y") . '<br \>';
    echo 'date(\"j, M Y\") = ' . date("j, M Y") . '<br \>';
    echo 'date(\"jS, M Y\") = ' . date("jS, M Y") . '<br \>';
    echo 'date(\"G:i:s\") = ' . date("G:i:s") . '<br \>';
    echo 'date(\"h:i:sa\") = ' . date("h:i:sa") . '<br \>';
    echo 'date(\"Y-m-d G:i:s\") = ' . date("Y-m-d G:i:s") . '<br \>';
?>
```

Step 2

Drugą funkcją pozwalającą na pobranie informacji dotyczących daty i czasu jest `getdate`. Jej wywołanie ma postać:

```
getdate([znacznik_czasu])
```

Również w tym przypadku parametr `znacznik_czasu` jest opcjonalny, a jego użycie ma takie samo znaczenie jak w przypadku `date`. Wynikiem działania `getdate` nie jest jednak ciąg znaków, ale tablica asocjacyjna zawierająca pobrane dane. Indeksy tej tablicy wraz z ich znaczeniami i przykładowymi wartościami:

Klucz	Opis	Przykłady zwracanych wartości
"seconds"	Ilość sekund	0-59
"minutes"	Ilość minut	0-59
"hours"	Ilość godzin	0-23
"mday"	Liczba będąca dniem miesiąca	1-31
"wday"	Dzień tygodnia w postaci cyfry	0 (dla niedzieli) aż do 6 (sobota)
"mon"	Miesiąc w postaci liczby	1-12
"year"	Pełny rok, w postaci liczby, 4 cyfry	1999, 2021
"yday"	Dzień danego roku, w postaci liczby	0-365
"weekday"	Pełna nazwa dnia tygodnia	Sunday
"month"	Pełna nazwa miesiąca	January
0	Sekundy, które upłynęły od Ery Uniksa	Zależnie od systemu

Wykorzystanie funkcji `getdate`:

```
<?php
$data = getdate();
$dzien = $data["mday"];
$ miesiac = $data["mon"];
$rok = $data["year"];
$dzien_roku = $data["yday"];
$nazwa_miesiac = $data["month"];
$nazwa_dzien = $data["weekday"];
if ($dzien < 10)
    $dzien = "0" . $dzien;
if ($miesiac < 10)
    $miesiac = "0" . $miesiac;
echo "Dziś jest $dzien-$miesiac-$rok ";
echo "Dziś jest $dzien_roku dzień roku ";
echo "Dziś jest $nazwa_dzien miesiąca $nazwa_miesiac ";
?>
```

Step 3

Do uzyskiwania danych opisujących datę i czas w językach narodowych można również wykorzystać funkcję `strftime`, która zwraca ciąg znaków sformatowanym zgodnie z szablonem przekazanym jako argument. Jej wywołanie ma postać:

```
strftime(format[ , timestamp])
```

Argument `format` to ciąg znaków, który może zawierać znaczniki przedstawione poniżej. Parametr `timestamp` jest opcjonalny. Pozwala on na uzyskanie ciągu znaków, który odpowiada konkretnej dacie. Jeśli zostanie pominięty, będzie użyty bieżący czas lokalny.

Znaczniki formatujące dla funkcji `strftime`:

- `%a` - skrócona nazwa dnia tygodnia zgodnie z lokalizacją
- `%A` - pełna nazwa dnia tygodnia zgodnie z lokalizacją
- `%b` - skrócona nazwa miesiąca zgodnie z lokalizacją
- `%B` - pełna nazwa miesiąca zgodnie z lokalizacją
- `%c` - preferowana reprezentacja daty i czasu zgodnie z lokalizacją
- `%C` - numer wieku (rok podzielony przez 100 i skrócony do liczby całkowitej, przedział od 00 do 99)
- `%d` - dzień miesiąca jako liczba dziesiętna (przedział od 01 do 31)
- `%D` - to samo co `%m/%d/%y`
- `%e` - dzień miesiąca jako liczba dziesiętna, przy czym pojedyncza cyfra poprzedzona jest spacją (przedział od " 1" do "31")
- `%g` - tak jak `%G`, ale bez uwzględnienia wieku

- `%G` - rok w zapisie czterocyfrowym, powiązany z numerem tygodnia wg ISO. Symbol ten ma ten sam format i wartość jak `%Y`, z tym wyjątkiem, że jeśli numer tygodnia wg ISO należy do poprzedniego lub następnego roku, to poprzedni lub następny rok jest zwracany przez ten symbol.
- `%h` - tak jak `%b`
- `%H` - godzina jako liczba dziesiętna w systemie 24-godzinnym (przedział od 00 do 23)
- `%I` - godzina jako liczba dziesiętna w systemie 12-godzinnym (przedział od 01 do 12)
- `%j` - dzień roku jako liczba dziesiętna (przedział od 001 do 366)
- `%m` - miesiąc jako liczba dziesiętna (przedział od 01 do 12)
- `%M` - minuty jako liczba dziesiętna
- `%n` - znak nowej linii
- `%p` - albo " `am` " lub " `pm` " zgodnie z podanym czasem, albo łańcuchy znaków odpowiadające lokalizacji
- `%r` - czas w notacji a.m. lub p.m.
- `%R` - czas w notacji 24-godzinnej
- `%S` - sekundy jako liczba dziesiętna
- `%t` - znak tabulacji
- `%T` - aktualny czas, odpowiednik `%H:%M:%S`
- `%u` - numer dnia tygodnia jako liczba dziesiętna [1,7], gdzie 1 oznacza poniedziałek
- `%U` - numer tygodnia aktualnego roku jako liczba dziesiętna, poczawszy od pierwszej niedzieli jako pierwszego dnia pierwszego tygodnia
- `%V` - numer tygodnia aktualnego roku wg ISO 8601:1988 jako liczba dziesiętna, przedział od 01 do 53, gdzie tydzień 1 jest pierwszym tygodniem, którym ma co najmniej 4 dni w aktualnym roku, przy czym pierwszym dniem tygodnia jest poniedziałek. (Przy użyciu `%G` lub `%g` otrzymuje się rok, który odpowiada numerowi tygodnia dla podanego znacznika czasu).
- `%W` - numer tygodnia aktualnego roku jako liczba dziesiętna, poczawszy od pierwszego poniedziałku, jako pierwszego dnia pierwszego tygodnia
- `%w` - dzień tygodnia jako liczba dziesiętna, poczawszy od niedzieli - numer 0
- `%x` - preferowana reprezentacja daty, zgodnie z lokalizacją, bez czasu
- `%X` - preferowana reprezentacja czasu, zgodnie z lokalizacją, bez daty
- `%y` - rok jako liczba dziesiętna, bez uwzględnienia wieku (przedział od 00 do 99)
- `%Y` - rok jako liczba dziesiętna, z wiekiem włącznie
- `%Z` - strefa czasowa, nazwa lub skrót
- `%%` - znak " % "

Dane zwracane przez `strftime`, będą zgodne z bieżącymi ustawieniami lokalnymi (narodowymi), które mogą być zmieniane za pomocą funkcji `setlocale`. Wywołanie `setlocale` ma postać:

```
setlocale(kategoria, locale)
```

Parametr `kategoria` wskazuje, jakie informacje mają być pobierane z uwzględnieniem ustawień narodowych, natomiast `locale` określa, który zestaw narodowy ma być użyty. Możliwe wartości parametru `kategoria`:

- `LC_ALL` dla wszystkich poniżej
- `LC_COLLATE` ciąg dla porównania
- `LC_CTYPE` na charakter i klasyfikację konwersji
- `LC_MONETARY` na postać ciągów numerycznych i walutowych
- `LC_NUMERIC` dla separatora dziesiętnego
- `LC_TIME` do formatowania daty i czasu z `strftime()`

Argument `locale` jest wyrażany w różny sposób w różnych systemach operacyjnych. W systemach uniksowych dla ustawień polskich należy użyć ciągu `pl_PL`. W systemach rodziny Windows należy użyć `plk` lub `polish`.

Wykorzystanie funkcji `setlocale`:

```
<?php
    setlocale(LC_ALL,'pl_PL.UTF-8');
    echo strftime("Bieżąca strefa czasowa: %Z.<br/>");
    echo strftime("Data: %d-%m-%Y<br/>");
    echo strftime("Czas: %H:%M:%S<br/>");
    echo strftime("Mamy %U tydzień i %j dzień roku.<br/>");
    echo strftime("Dziś jest %A, %d, %B, %Y r.<br/>");
?>
```

Step 4

Do utworzenia znacznika czasu, który może być wykorzystywany jako drugi parametr funkcji opisywanych wcześniej, można wykorzystać funkcję `mktime`. W pełnej wersji przyjmuje ona aż siedem argumentów, chociaż w rzeczywistości wszystkie są opcjonalne. Schemat wywołania jest następujący:

```
mktime(godzina, minuta, sekunda, miesiąc, dzień, rok)
```

Parametry można pomijać wyłącznie od strony prawej do lewej. Brakujące parametry są uzupełniane przez czas lokalny. Jeśli więc interesuje nas znacznik czasu odpowiadający godzinie 13:24:52 w aktualnym dniu, zapisujemy wywołanie:

```
mktime(13, 24, 52)
```

Jak w praktyce można wykorzystać funkcję `mktime`, pokazuje poniższy skrypt. Pozwala on na obliczenie liczby dni występujących pomiędzy dwiema datami.

```
<?php
$r1 = 2018;
$m1 = 5;
$d1 = 1;
$r2 = 2018;
$m2 = 9;
$d2 = 1;

$time1 = mktime(0,0,0,$m1,$d1,$r1);
$time2 = mktime(0,0,0,$m2,$d2,$r2);
$time = abs(ceil(($time1 - $time2)/86400));
echo "Pomiędzy $d1-$m1-$r1, a $d2-$m2-$r2 mamy $time dni."
?>
```

Znacznik czasu można również uzyskać poprzez konwersję ciągu znaków opisujących datę i czas. Służy do tego funkcja `strtotime`, której wywołanie jest następujące:

```
strtotime(format[ , timestamp]);
```

Argument `format` to opis daty (ciąg znaków), dla której chcemy uzyskać znacznik czasu. Natomiast `timestamp` to opcjonalny znacznik czasu, który zostanie wykorzystany przy obliczaniu wyniku. Poniższy kod przedstawia kilka przykładów użycia funkcji `strtotime`.

```
<?php
echo strtotime("now") . "<br/>";
echo strtotime("18 march 1976") . "<br/>";
echo strtotime("-12 day") . "<br/>";
echo strtotime("+2 day 4 hours 15 minutes 12 seconds") . "<br/>";
echo strtotime("next friday") . "<br/>";
?>
```

Ostatnią funkcją, którą zajmiemy się w tej sekcji, to `microtime`. Ona również zwraca aktualny znacznik czasu, z tym że zawierający dodatkowe dane odnośnie liczby mikrosekund. Ogólne wywołanie ma postać:

```
microtime(tryb)
```

Tryb jest tu opcjonalnym parametrem typu `boolean`. W przypadku gdy jest obecny i równy `true`, zwracana jest wartość rzeczywista. W pozostałych przypadkach zwracany jest ciąg w postaci `msec sec`, gdzie `msec` oznacza liczbę mikrosekund, a `sec` liczbę sekund, które upłynęły od początku epoki Uniksa. Takie zachowanie pozwala np. obliczyć czas generowania strony.

```
<?php
$time1 = microtime(true);
for ($i=0; $i < 10000; $i++)
    for ($j=0; $j < 1000; $j++);
$time2 = microtime(true);
$time = round($time2 - $time1,4);
echo "Strona została wygenerowana w czasie $time sekund";
?>
```

Step 5

Oprócz wspomnianych wyżej istnieje jeszcze wiele innych funkcji służących do obsługi daty i czasu. Należą do nich między innymi:

- `checkdate` - Zadaniem tej funkcji jest sprawdzenie, czy dane przekazane jej w postaci argumentów tworzą poprawną datę. Funkcja zwraca wartość `true`, jeżeli podany dzień miesiąc rok tworzą poprawną datę.

```
<?php
echo checkdate(5,25,2014);
echo checkdate(2,29,2021);
?>
```

- `gmdate` - Funkcja zwraca ciąg znaków sformatowany zgodnie z szablonem podanym jako parametr `format`. Jej wywołanie ma postać: `gmdate(format[, timestamp])`. W odróżnieniu od omawianej wcześniej `date`, w tym przypadku zwracany jest czas GMT a nie lokalny.

```

<?php
    echo "Wywołanie funkcji date<br/>";
    echo "date(\"G:i\") = " . date("G:i") . "<br/>";
    echo "date(\"r\") = " . date("r") . "<br/>";
    echo "date(\"c\") = " . date("c") . "<br/>";
    echo "date(\"T\") = " . date("T") . "<br/>";
    echo "date(\"Z\") = " . date("Z") . "<br/>";

    echo "Wywołanie funkcji gmdate<br/>";
    echo "gmdate(\"G:i\") = " . gmdate("G:i") . "<br/>";
    echo "gmdate(\"r\") = " . gmdate("r") . "<br/>";
    echo "gmdate(\"c\") = " . gmdate("c") . "<br/>";
    echo "gmdate(\"T\") = " . gmdate("T") . "<br/>";
    echo "gmdate(\"Z\") = " . gmdate("Z") . "<br/>";

?>

```

- `localtime` - Funkcja zwraca tablicę, której poszczególne elementy zawierają dane odnośnie daty i czasu. Jej wywołanie jest następujące:

```
localtime ([ int $znacznik_czasu [, bool $asocjacyjna ] ] )
```

Pierwszym argumentem funkcji jest `znacznik_czasu`. Jeśli żaden `znacznik_czasu` nie zostanie podany, funkcja wykorzystuje aktualny czas, taki jak zwrócony przez `time()`. Drugi argument, `asocjacyjna`, jeśli jest ustawiony na `FALSE`, lub nie jest podany, powoduje zwrócenie zwykłej, indeksowanej liczbowo tablicy. Jeśli zaś jest ustawiony na `TRUE`, to funkcja zwróci tablicę asocjacyjną zawierającą wszystkie rozmaite elementy struktury zwracanej przez wywołanie funkcji. Nazwy kluczy tablicy są następujące:

Nazwa klucza	Opis klucza
<code>tm_sec</code>	sekundy
<code>tm_min</code>	minuty
<code>tm_hour</code>	godziny
<code>tm_mday</code>	dzień miesiąca
<code>tm_mon</code>	miesiąc roku, gdzie 0 to styczeń
<code>tm_year</code>	ilość lat od 1900
<code>tm_wday</code>	dzień tygodnia
<code>tm_yday</code>	dzień roku
<code>tm_isdst</code>	czy aktualnie czas zimowy

```

<?php
    $time = localtime(time(),True);
    $wday = $time["tm_wday"];
    $yday = $time["tm_yday"];
    $month = $time["tm_mon"] + 1;
    echo "Dziś jest $wday dzień tygodnia. Jest to $yday dzień w roku. Obecnie mamy $month miesiąc." ;
?>

```

Step 6

PHP pozwala na wykonywanie różnych operacji na strukturze systemu plików. Bez problemów można odczytywać rozmaite informacje, np. zawartość wskazanego katalogu lub wielkość pliku, jak również dokonywać modyfikacji samej struktury, czyli tworzyć, usuwać lub przenosić w inne miejsce na dysku pliki i katalogi. Każda taka operacja jest wykonywana przez dedykowaną funkcję. W celu odczytania zawartości wybranego katalogu należy wykonać w sumie trzy operacje:

1. Otwarcie katalogu
2. Odczyt danych
3. Zamknięcie katalogu

Otwarcie katalogu to rezerwacja zasobów oraz informacja dla systemu, że będziemy wykonywać dalsze operacje. Aby otworzyć katalog, używa się funkcji `opendir`, której wywołanie ma postać:

```
opendir('nazwa_katalogu')
```

Funkcja zwraca deskryptor katalogu, czyli specjalny identyfikator, którym należy się posługiwać przy wykonywaniu dalszych operacji. Odczyt zawartości katalogu, przeprowadza się za pomocą funkcji `readdir`, której należy przekazać w formie argumentu deskryptor uzyskany przy wywołaniu `opendir`. Funkcja `readdir` działa w taki sposób, że każde jej wywołanie powoduje zwrocenie nazwy kolejnego elementu znajdującego się w katalogu; po odczytaniu wszystkich elementów zwraca ona wartość `false`. Gdy odczytana zostanie zawartość katalogu, należy go zamknąć za pomocą funkcji `closedir`, przekazując jej w postaci argumentu deskryptor otrzymany w wywołaniu `opendir`. Wykorzystując powyższe informacje, można już w prosty sposób napisać skrypt, który wyświetli zawartość wybranego katalogu.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Test PHP</title>
</head>
<body>
<?php
    $dir = "./";
    if (!($fd = opendir($dir))){
        exit("Nie mogę otworzyć katalogu $dir");
    }
    while (($file = readdir($fd)) !== false)
        echo "$file<br/>";
    closedir($fd);
?>
</body>
</html>
```

W PHP5 istnieje również funkcja o nazwie `scandir`, która za jednym wywołaniem pobiera zawartość całego katalogu i zwraca ją w postaci tablicy. Wywołanie tej funkcji ma postać:

```
scandir('nazwa katalogu'[, sortowanie])
```

Parametr `nazwa_katalogu` określa katalog, z którego będą pobierane dane, natomiast `sortowanie` – sposób sortowania. Ustawienie drugiego parametru na `0` powoduje, że nazwy plików i katalogów będą sortowane alfabetycznie w porządku rosnącym, a ustawienie na `1` lub inną wartość niezerową – że nazwy będą sortowane w porządku malejącym.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Test PHP</title>
</head>
<body>
<?php
    $dir = "./";
    $arr = scandir("$dir");
    foreach ($arr as $file){
        if($file != '.' && $file != '..')
            echo "$file<br/>";
    }
?>
</body>
</html>
```

Step 7

Do tworzenia katalogów służy funkcja `mkdir`. Jej wywołanie wygląda następująco:

```
mkdir('nazwa'[, tryb[, zagnieżdżone]])
```

Parametr `nazwa` określa tu nazwę katalogu, natomiast `tryb` – prawa dostępu. Argument `nazwa` może określać zarówno względną, jak i bezwzględną ścieżkę dostępu. Jeśli nie ma określonej ścieżki, nowy katalog zostanie utworzony w katalogu bieżącym. Trzeci parametr ustawiony na `true` umożliwia utworzenie zagnieżdżonej struktury katalogów.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Test PHP</title>
</head>
<body>
<?php
    mkdir("nowy_katalog"); #Utworzy katalog w projekcie
    mkdir("/home/mateusz/nieistniejacy_katalog/nowy_katalog"); #Nie utworzy katalogu nowy_katalog, ponieważ
    nie ma katalogu o nazwie nieistniejący katalog
    mkdir("/home/mateusz/nieistniejacy_katalog2/nowy_katalog",0777,true); #Tutaj zostanie utworzona ścieżka
    katalogu oraz zostanie jemu przydzielone pełne prawa dostępu w systemie Linux
?>
</body>
</html>

```

The screenshot shows the PhpStorm IDE interface. The code editor displays the same PHP script as above. A warning message is visible in the bottom right corner of the editor area:

```

PHP Warning: mkdir(): File exists in /home/mateusz/PhpstormProjects/PHPExampleProject/index.php on line 9
PHP Warning: mkdir(): No such file or directory in /home/mateusz/PhpstormProjects/PHPExampleProject/index.php on line 10

```

Below the editor, the run tool window shows the output:

```

Process finished with exit code 0

```

Do usuwania katalogów służy funkcja `rmdir`. Jego wywołanie wygląda następująco:

```
rmdir('nazwa')
```

Ważne, że usuwany katalog **musi być pusty**.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Test PHP</title>
</head>
<body>
<?php
    rmdir("nowy_katalog");
    rmdir("/home/mateusz/nieistniejacy_katalog2/nowy_katalog");
    rmdir("/home/mateusz/nieistniejacy_katalog2");
?>
</body>
</html>

```

The screenshot shows the PhpStorm IDE interface. The project structure on the left includes files like index.php, train1.jpg, back.jpeg, style.css, kalkulator-skrypt.php, and jakis-skrypt.php. The current file is index.php. The code in the editor is:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Test PHP</title>
</head>
<body>
<?php
    rmdir( directory: "nowy_katalog");
    rmdir( directory: "/home/mateusz/nieistniejacy_katalog2/nowy_katalog");
    rmdir( directory: "/home/mateusz/nieistniejacy_katalog2");
?
</body>
</html>
```

The run tab at the bottom shows the command: /usr/bin/php /home/mateusz/PhpstormProjects/PHPExampleProject/index.php. The output shows a PHP warning: mkdir(): No such file or directory in /home/mateusz/PhpstormProjects/PHPExampleProject/index.php on line 10.

Jeśli chcemy się dowiedzieć, jaki jest aktualny katalog bieżący, powinniśmy użyć funkcji `getcwd`, która zwraca jego nazwę w postaci ciągu znaków.

The screenshot shows the PhpStorm IDE interface. The project structure on the left includes files like index.php, train1.jpg, back.jpeg, style.css, kalkulator-skrypt.php, and jakis-skrypt.php. The current file is index.php. The code in the editor is:

```
<?php
echo getcwd();
?>
```

The run tab at the bottom shows the command: /usr/bin/php /home/mateusz/PhpstormProjects/PHPExampleProject/index.php. The output shows the current working directory: /home/mateusz/PhpstormProjects/PHPExampleProject.

Process finished with exit code 0

Do zmiany katalogu bieżącego wykorzystywana jest funkcja `chdir`, której w postaci argumentu należy przekazać nazwę nowego katalogu bieżącego. Wywołanie ma postać:

The screenshot shows the PhpStorm IDE interface. The project structure on the left includes files like index.php, train1.jpg, back.jpeg, style.css, kalkulator-skrypt.php, and jakis-skrypt.php. The current file is index.php. The code in the editor is:

```
chdir('nazwa')
```



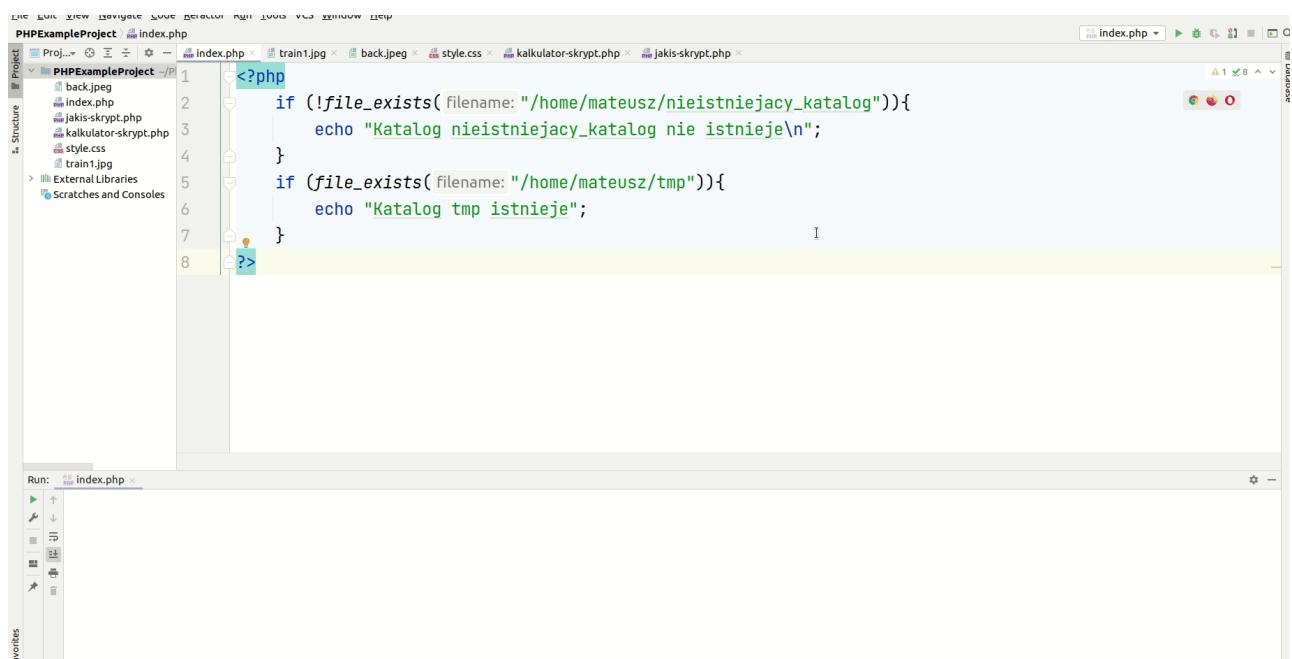
```
<?php
echo getcwd() . "\n";
chdir("../");
echo getcwd();
?>
```

Aby ustalić czy dany katalog istnieje należy użyć funkcji `file_exists`, której wywołanie ma postać:

```
file_exists('nazwakatalogu')
```

Zwracaną wartością jest `true`, jeśli katalog istnieje, lub `false` w przeciwnym razie. Ta sama funkcja jest wykorzystywana do sprawdzania plików.

```
<?php
if (!file_exists("/home/mateusz/nieistniejacy_katalog")){
    echo "Katalog nieistniejacy_katalog nie istnieje\n";
}
if (file_exists("/home/mateusz/tmp")){
    echo "Katalog tmp istnieje";
}
?>
```



Step 8

Czynności tworzenia plików na dysku oraz ich otwierania są ze sobą związane i wykonuje je jedna funkcja – `fopen`. Wywołanie funkcji ma postać:

```
fopen ( string$nazwa_pliku , string$tryb [, bool$include_path ] )
```

Argument `nazwa_pliku` to ciąg znaków wskazujący nazwę pliku, który należy otworzyć. Parametr `tryb_otwarcia` określa tryb otwarcia pliku i może przyjmować wartości:

Tryb	Opis
'r'	Otwiera plik tylko do odczytu; umieszcza wskaźnik pliku na jego początku.
'r+'	Otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku.
'w'	Otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje, to próbuje go utworzyć.
'w+'	Otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje, to próbuje go utworzyć.
'a'	Otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje, to próbuje go utworzyć.
'a+'	Otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje, to próbuje go utworzyć.
'x'	Tworzy i otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie funkcji <code>fopen()</code> nie powiedzie się.
'x+'	Tworzy i otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie <code>fopen()</code> nie powiedzie się.

Parametr `tryb_otwarcia` może również zawierać określenie typu pliku: `b` – dla plików binarnych bądź `t` – dla plików tekstowych. Parametr `include_path` jest to opcjonalny parametr, który może być ustawiony na `'1'` lub `true` jeśli chcesz szukać pliku także w `include_path`.

Gdy zakończą się wykonywane na pliku operacje, powinien on zostać zamknięty za pomocą funkcji `fclose`, której typowe wywołanie ma postać

```
fclose(deskryptor);
```

deskryptor to oczywiście wartość uprzednio zwrócona przez `fopen`.

Pojedyncze wiersze tekstu można odczytać z pliku za pomocą funkcji o nazwie `fgets`, której schematyczne wywołanie ma postać:

```
fgets(deskryptor[, ile])
```

deskryptor to oczywiście wartość uprzednio zwrócona przez `fopen`, a argument `ile` jest opcjonalny i określa maksymalną liczbę znaków do odczytu. Funkcję `fgets` najczęściej stosuje się w pętli `While` testującej osiągnięcie końca pliku (`feof`). Pokazuje to poniższy przykład:

```
<?php
if (!$fd = fopen('./test.txt', 'r')){
    echo "Nie można otworzyć pliku test.txt";
}
else{
    while(!feof($fd)){
        $str = fgets($fd);
        $str = nl2br($str); #Funkcja ta dodaje automatycznie znacznik <br/>
        echo $str;
    }
    fclose($fd);
}
?>
```

Step 9

Jeżeli spodziewamy się, że w pliku, z którego będziemy odczytywać dane, znajdują się niepotrzebne nam znaczniki HTML, możemy do odczytu danych wykorzystać funkcję `fgetss`. Jej wywołanie ma postać:

```
fgetss(deskryptor[, ile[, tagi]])
```

deskryptor to oczywiście wartość uprzednio zwrócona przez `fopen`, argument `ile` jest opcjonalny i określa maksymalną liczbę znaków do odczytu, a parametr `tagi` określa znaczniki, które mają nie być usuwane.

```

<?php
    if (!($fd = fopen('plik.html', 'r'))){
        echo "Nie można otworzyć pliku plik.html";
    }
    else{
        $lineno = 0;
        while(!feof($fd)){
            $str = fgets($fd);
            $str = nl2br($str); #Funkcja ta dodaje automatycznie znacznik <br/>
            echo $lineno . " " . $str;
            $lineno++;
        }
        fclose($fd);
    }
?>

```

Uwaga. Funkcja `fgetss` nie występuje już w PHP8 i jej używanie jest wysoko odradzane.

Odczyt pojedynczych znaków z pliku umożliwia funkcja `fgetc`. Przyjmuje ona jeden argument, jakim jest deskryptor pliku, który będzie odczytywany, oraz zwraca ciąg zawierający pojedynczy odczytany znak. Wywołanie ma więc schematyczną postać:

```
fgetc(deskryptor);
```

Po jej wykonaniu wskaźnik pliku przesuwany jest o jeden bajt do przodu. W przypadku gdy zostanie osiągnięty koniec pliku (`feof`), `fgetc` zwraca wartość `false`.

```

<?php
    if (!($fd = fopen('test.txt', 'r'))){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        while (($str = fgetc($fd)) !== false){
            if ($str == "\n")
                $str = "<br/>";
            echo $str;
        }
        fclose($fd);
    }
?>

```

Kiedy chce się odczytać określoną liczbę bajtów lub też dane z pliku binarnego, należy użyć funkcji `fread`. Jej wywołanie ma postać:

```
fread(deskryptor, ile)
```

`deskryptor` to wartość zwrócona przez `fopen`, argument `ile` określa liczbę bajtów do odczytania. Funkcja zwraca odczytany fragment pliku w postaci ciągu typu string.

```

<?php
    if (!($fd = fopen('test.txt', 'rb'))){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        while(!feof($fd)){
            echo fread($fd, 4096);
        }
        fclose($fd);
    }
?>

```

Powyższy kod można przerobić tak, aby nie korzystać z funkcji `feof` następująco:

```

<?php
    if (!($fd = fopen('test.txt', 'rb'))){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        while (($str = fread($fd, 4096)) != ""){
            echo $str;
        }
        fclose($fd);
    }
?>

```

Odczyt pełnej zawartości pliku można wykonać na kilka sposobów. Sposób pierwszy to wykorzystanie przedstawionej wyżej funkcji `fread`, której jako drugi argument zostanie przekazana wielkość pliku (funkcja `filesize`).

```

<?php
    if (!($fd = fopen('test.txt', 'rb'))){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        $r = filesize("test.txt");
        $str = fread($fd,$r);
        echo $str;
        fclose($fd);
    }
?>

```

Czynność taka może również zostać przeprowadzona dużo prościej, jeśli skorzysta się z funkcji dedykowanych `readfile` lub `file_get_contents`. Aby wysłać zawartość pliku `test.txt` do przeglądarki, wystarczy wykonać tylko jedną instrukcję:

```
readfile('test.txt');
```

W podobny sposób działa też `file_get_contents`. Jako argument przyjmuje ona nazwę pliku, zwraca natomiast jego odczytaną zawartość w postaci wartości typu string lub wartości `false`, jeśli odczyt się nie udał.

```

<?php
    if (!($fd = fopen('test.txt', 'rb'))){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        echo readfile("test.txt") . "<br/>";
        echo file_get_contents("test.txt");
    }
?>

```

Step 10

Zapis danych do pliku może być przeprowadzony za pomocą funkcji `fwrite` (można też korzystać z jej aliasu `fputs`). Przyjmuje ona trzy argumenty, a jej schematyczne wywołanie ma postać:

```
fwrite(deskryptor, str[, ile])
```

`deskryptor` to deskryptor pliku zwrócony przez wywołanie funkcji `fopen`. `str` to ciąg typu string zawierający dane, które mają zostać zapisane, natomiast `ile` to opcjonalny parametr wskazujący, ile bajtów z ciągu `str` ma zostać zapisanych.

```

<?php
    $str = "Przykładowy wiersz tekstu";
    if (!($fd = fopen("test2.txt", "wb"))){
        echo "Nie mogę utworzyć pliku test2.txt";
    }
    else{
        if (fwrite($fd, $str) === false){
            echo "Wystąpił błąd. Zapis nie został dokonany";
        }
        else{
            echo "Ciąg został zapisany\n";
        }
        $str = "<br/>Kolejna linia tekstu";
        if (fwrite($fd, $str, 10) === false){
            echo "Wystąpił błąd. Zapis nie został dokonany";
        }
        else{
            echo "Ciąg o długości 10 został zapisany";
        }
    }
?>

```

Drugą funkcją, która pozwala na zapis danych do pliku, jest `file_put_contents`. Ta funkcja może przyjmować do czterech argumentów. Schemat jej wywołania jest następujący:

```
file_put_contents(„nazw_pliku”, dane[, flagi]);
```

Oznacza ono zapisanie danych wskazywanych przez argument `dane` do pliku o nazwie wskazywanej przez argument `nazwa_pliku`. Argument `dane` może być ciągiem znaków bądź tablicą. Parametr `flagi` może przyjmować następujące wartości;

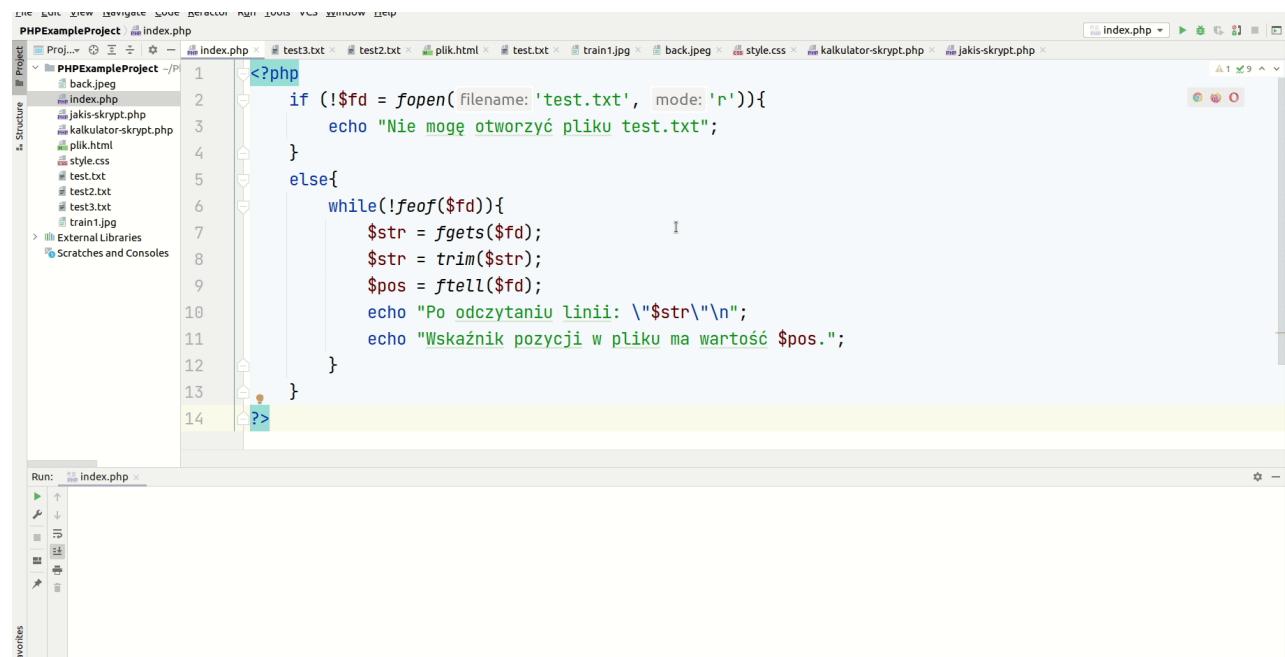
- `FILE_USE_INCLUDE_PATH` – obecność pliku będzie sprawdzana w lokalizacjach wskazanych przez zmienną konfiguracyjną `include_path`.
- `FILE_APPEND` – dane będą dopisywane na końcu pliku
- `LOCK_EX` – plik zostanie zablokowany na czas zapisu.

```
<?php
    $str = "Przykładowy wiersz tekstu\n";
    if (file_put_contents("test3.txt", $str, FILE_APPEND) === false){
        echo "Wystąpił błąd. Zapis nie został dokonany";
    }
    else{
        echo "Ciąg został zapisany";
    }
?>
```



Odczyt danych powoduje przesuwanie wskaźnika pozycji w pliku, dzieje się to jednak automatycznie, bez wiedzy programisty. Istnieje jednak możliwość odczytania aktualnej pozycji tego wskaźnika, a także zmiany jego wartości. Służą do tego trzy funkcje: `fseek`, `frewind` i `fgetpos`. Funkcja `fgetpos` zwraca aktualną pozycję w pliku (typ int). Przyjmuje ona jako argument deskryptor pliku otwartego za pomocą `fopen`.

```
<?php
    if (!($fd = fopen('test.txt', 'r'))){
        echo "Nie mogę otworzyć pliku test.txt";
    }
    else{
        while(!feof($fd)){
            $str = fgets($fd);
            $str = trim($str);
            $pos = ftell($fd);
            echo "Po odczytaniu linii: \"$str\"\n";
            echo "Wskaźnik pozycji w pliku ma wartość $pos.";
        }
    }
?>
```



Aktualna pozycja w pliku może zostać zmieniona za pomocą funkcji `fseek`. Jej schematyczne wywołanie ma postać:

```
fseek(deskryptor, ile[, skąd])
```

`deskryptor` to parametr zwrócony przez funkcję `fopen`, `ile` określa liczbę bajtów przesunięcia, natomiast `skąd` określa pozycję, od której nastąpi przesunięcie.

Parametr `skąd` jest opcjonalny i może przyjmować następujące wartości:

- `SEEK_SET` – oznacza przesunięcie względem początku pliku
- `SEEK_CUR` – oznacza przesunięcie względem pozycji bieżącej
- `SEEK_END` – oznacza przesunięcie względem końca pliku

Domyślną wartością jest `SEEK_SET`. Jeśli na przykład zmienna `$fd` zawiera deskryptor pliku zwrócony przez `fopen` to wywołanie:

- `fseek($fd, 0)` – ustawi wskaźnik na początek pliku
- `fseek($fd, 0,SEEK_SET)` – ustawi wskaźnik na początek pliku
- `fseek($fd, 0,SEEK_END)` – ustawi wskaźnik na koniec pliku
- `fseek($fd, 10,SEEK_SET)` – ustawi wskaźnik na 20 bajcie pliku
- `fseek($fd, -10,SEEK_CUR)` – ustawi wskaźnik na 10 bajcie przed aktualną pozycją pliku
- `fseek($fd, 10,SEEK_CUR)` – ustawi wskaźnik na 10 bajcie za aktualną pozycją pliku
- `fseek($fd, -10,SEEK_END)` – ustawi wskaźnik na 10 bajcie przed końcem pliku

```

<?php
    if (!($fd = fopen('test.txt', 'r'))){
        echo "Nie mogę otworzyć pliku test.txt";
    }
    else{
        $str = fgets($fd);
        echo $str;
        $str = fgets($fd);
        echo $str;
        fseek($fd, SEEK_SET);
        $str = fgets($fd);
        echo $str;
    }
?>

```

The screenshot shows a PHP code editor with the following code:

```

<?php
if (!($fd = fopen('test.txt', 'r'))){
    echo "Nie mogę otworzyć pliku test.txt";
}
else{
    $str = fgets($fd);
    echo $str;
    $str = fgets($fd);
    echo $str;
    fseek($fd, SEEK_SET);
    $str = fgets($fd);
    echo $str;
}
?>

```

The line `fseek($fd, SEEK_SET);` is highlighted in yellow. The IDE interface includes a toolbar, a project tree on the left, and a run configuration bar at the bottom.

Ostatnia z omawianych funkcji – `rewind` – przesuwa wskaźnik pozycji w pliku na pozycję 0, czyli na jego początek. Funkcja `rewind` przyjmuje tylko jeden argument, którym jest deskryptor pliku.

```

<?php
if (!($fd = fopen('test.txt', 'r'))){
    echo "Nie mogę otworzyć pliku test.txt";
}
else{
    $str = fgets($fd);
    echo $str;
    $str = fgets($fd);
    echo $str;
    rewind($fd);
    $str = fgets($fd);
    echo $str;
}
?>

```

The screenshot shows the same PHP code as the previous one, but with the `rewind` function added after the first `fgets` call:

```

<?php
if (!($fd = fopen('test.txt', 'r'))){
    echo "Nie mogę otworzyć pliku test.txt";
}
else{
    $str = fgets($fd);
    echo $str;
    $str = fgets($fd);
    echo $str;
    rewind($fd);
    $str = fgets($fd);
    echo $str;
}
?>

```

The line `rewind($fd);` is highlighted in yellow. The IDE interface is identical to the previous screenshot.

Step 12

Jeżeli nie zadbane o prawidłową synchronizację dostępu przechowywanie danych w plikach może przysporzyć nam wielu problemów. Co się bowiem stanie, jeśli naszą witrynę odwiedzi naraz kilka osób i wszystkie wywołują skrypt operujący na danych zapisanych w jednym z plików? Do synchronizacji dostępu do pliku można użyć funkcji `flock`. Wywołanie funkcji `flock` powoduje założenie takiej blokady na dany plik, że dostęp do niego będzie miał tylko skrypt, który tę funkcję wywołał. Wywołanie ma postać:

```
flock(deskryptor, operacja)
```

`deskryptor` to parametr zwrócony przez funkcje `fopen`, natomiast `operacja` określa rodzaj blokowania. Parametr ten może przyjmować następujące wartości:

- `LOCK_SH` – blokada zapisu, możliwy odczyt.
- `LOCK_EX` – pełna blokada.
- `LOCK_UN` – zwolnienie blokady.

```
<?php
$fp = fopen("test.txt", "r+");

if (flock($fp, LOCK_EX)) {
    fwrite($fp, "Write something here\n");
    flock($fp, LOCK_UN);
} else {
    echo "Couldn't get the lock!";
}

fclose($fp);
?>
```



Step 13

Do usunięcia plików służy funkcja `unlink`, która w postaci argumentu przyjmuje nazwę usuwanego pliku, a zatem jej wywołanie ma schematyczną postać:

```
unlink('nazwa_pliku');
```

```
<?php
    unlink("test.txt");
?>
```

```
<?php  
    unlink(filename: "test.txt");  
?>
```

Run: /usr/bin/php /home/mateusz/PhpstormProjects/PHPExampleProject/index.php

Process finished with exit code 0

Rozmiar pliku można sprawdzić, wywołując funkcje o nazwie `filesize`, schematycznie:

```
filesize('nazwapliku')
```

Funkcja zwraca wartość typu `integer` określającą wielkość pliku w bajtach.

Oprócz informacji o wielkości pliku przydatnych może być również kilka innych danych pobieranych za pomocą odpowiednich funkcji. Oto niektóre z nich:

- `fileatime` – zwraca znacznik czasu Uniksa określający czas ostatniego dostępu do pliku
- `filectime` - zwraca znacznik czasu Uniksa określający czas ostatniej zmiany metadanych pliku.
- `filetype` – zwraca ciąg znaków określający typ pliku. Zwracane wartości to: `fifo`, `char`, `dir`, `block`, `link`, `file` i `unknown`.
- `fileperms` – zwraca wartość określającą prawa dostępu do pliku
- `fileowner` – zwraca identyfikator właściciela pliku.

```
<?php  
    echo filesize('test2.txt') . "\n";  
    echo filetype('test2.txt') . "\n";  
    echo filetype('../') . "\n";  
    echo fileperms('test2.txt') . "\n";  
    echo fileowner('test2.txt') . "\n";  
?>
```

Jeśli chcemy uzyskać informacje o ilości wolnego miejsca na dysku, możemy skorzystać z funkcji `disk_free_space`. Przyjmuje ona w postaci argumentu `nazwę katalogu` i zwraca ilość wolnego miejsca (w bajtach) na dysku logicznym, na którym znajduje się ten katalog. Jeśli interesuje nas nie wolna, ale całkowita ilość miejsca, zamiast `disc_free_space` należy zastosować funkcję `disk_total_space`.

```
<?php  
    $wolne_miejsce = disk_free_space("/");  
    $calkowite_miejsce = disk_total_space("/");  
    $zajete_miejsce = $calkowite_miejsce - $wolne_miejsce;  
  
    echo "Całkowite miejsce na dysku: $calkowite_miejsce\n";  
    echo "Dostępne miejsce na dysku: $wolne_miejsce\n";  
    echo "Zajęte miejsce na dysku: $zajete_miejsce\n";  
?>
```

4.1 4.1 Teoria

Step 1

Ciasteczka (ang. cookies) to niewielkie informacje tekstowe, wysyłane przez serwer WWW i zapisywane po stronie użytkownika (zazwyczaj na twardym dysku). Domyślne parametry ciasteczek pozwalają na odczytanie informacji w nich zawartych jedynie serwerowi, który je utworzył. Ciasteczka są stosowane najczęściej w przypadku liczników, sond, sklepów internetowych czy stron wymagających logowania. Mechanizm ciasteczek został wymyślony przez byłego pracownika Netscape Communications – Lou Montulliego. Ciasteczka mogą zawierać różnego rodzaju informacji o użytkowniku danej strony WWW i "historii" jego łączności z daną stroną (a właściwie serwerem). Zazwyczaj wykorzystywane są do automatycznego rozpoznawania danego użytkownika przez serwer, dzięki czemu może on wygenerować stronę ściśle dedykowaną. Umożliwia to tworzenie spersonalizowanych serwisów WWW, obsługi logowania, "koszyków zakupowych" w internetowych sklepach itp. Zastosowanie ciasteczek do sond i liczników internetowych wygląda następująco – serwer może łatwo sprawdzić, czy z danego komputera oddano już głos lub też czy odwiedzono daną stronę, i na tej podstawie wykonać odpowiednie operacje.



Dane zapisane w ciasteczkach mają postać naprzemiennych ciągów nazwy i wartości odpowiadającej jej zmiennej. Serwer WWW chcąc wysłać żądanie utworzenia ciasteczka na dysku użytkownika dołącza do nagłówka HTTP polecenie " Set-Cookie ", po którym następuje ciąg przekazywanych danych. Zapamiętane ciasteczko może najczęściej odczytać jedynie serwer, który je wysłał. W danych po poleceniu Set-Cookie określone są:

- nazwa i przypisaną jej wartość;
- domena i ścieżka dostępu, które są związane z przekazywanym ciasteczkiem;
- czas ważności danego ciasteczka (po jego upłynięciu przeglądarka usunie je).

Do zapisania ciasteczka wymagana jest jedynie jego nazwa. Jeśli nie zostanie podana domena, do wartości zapisanych w ciasteczkach dostęp będzie miał jedynie serwer, z którego wysłano żądanie zapisu. Niepodanie czasu ważności spowoduje usunięcie ciasteczka po zamknięciu przeglądarki. Ciasteczka, które wygasają po zakończonej sesji, zwane są ciasteczkami sesyjnymi. Mają one ustalony okres ważności, którego mechanizm wymusza serwer. Działanie mechanizmu ciasteczek po stronie użytkownika zależy od konfiguracji jego przeglądarki. Niektóre z nich umożliwiają odmowę zapisu, inne pozwalają na ustawienie daty wygaśnięcia innej od tej deklarowanej w nagłówku HTTP. Do właściwości ciasteczka należą:

- Ciasteczka o tej samej nazwie ale o innych ścieżkach będą nadpisywane.
- W celu skasowania należy wysłać ciasteczko o takiej samej nazwie i czasie wygaśnięcia z minioną datą.
- Możliwe jest wysyłanie kilku ciasteczek w jednym nagłówku
- Istnieją limity przy zapisywaniu ciasteczek na dysku (po ich przekroczeniu przeglądarka usuwa starsze ciasteczka, w niektórych przeglądarkach poniższe limity mogą się różnić):
 - maksymalna liczba ciasteczek: 300.
 - maksymalna wielkość ciasteczka: 4 kilobajty
 - maksymalna liczba ciasteczek z jednego serwera lub z jednej ścieżki: 20.

Step 2

Pojedyncze cookie to ciąg znaków określający nazwę i przypisaną jej wartość oraz dodatkowe parametry. Taki ciąg ma postać:

```
nazwa=wartość; expires=DATA; path=ŚCIEŻKA; domain=DOMENA; secure
```

Ciąg `nazwa=wartość` jest jedynym wymaganym atrybutem przy wysyłaniu ciasteczka. Składa się z dowolnych znaków z wyjątkiem średników, przecinków, białych spacji i slashów (/). `expires=data` informuje przeglądarkę o dacie wygaśnięcia danego ciasteczka. Zostanie ono usunięte z dysku, gdy jego data ważności zostanie przekroczena. Jeśli nie podano daty wygaśnięcia, to ciasteczko zostanie usunięte po zakończeniu sesji. Data musi być podana w następującym formacie (przykład): "Tuesday, 05-Nov-2004 08:30:09 GMT". `path=ścieżka` jest podawany w celu ograniczenia widoczności ciasteczka do danej ścieżki dostępu do katalogu. Przykładowo ustawienie tego parametru na `/` powoduje że cookie dostępne jest w całej domenie. `domain=domena` określa widoczność ciasteczka. W trakcie sprawdzania pliku na komputerze klienta zawierającego ciasteczka, przeglądarka porównuje zapisaną domenę z domeną serwera, do którego wysyła nagłówki. Przeglądarka wysyła wszystkie nie przeterminowane ciasteczka, których domena jest zawarta w domenie serwera (dodatekowo może być sprawdzana ścieżka wywoływanego pliku i typ połączenia). `secure` nie posiada wartości. Jeśli zostanie podany, to ciasteczko będzie widoczne (wysiane) tylko wtedy gdy połączenie będzie szyfrowane `HTTPS`.

PHP posiada jedną funkcję do tworzenia ciasteczek - `setcookie`, którego wywołanie wygląda następująco:

```
setcookie("nazwa", "wartość", "czaswygaśnięcia", "ścieżka", "domena", "tryb")
```

```
<!--index.php-->
<?php
    setcookie("imie", "Mateusz");
    setcookie("data", "2021-04-15", time() + 60 * 60 * 24 * 14);
?>
```

Do odczytu ciasteczka używamy tablicy `$_COOKIE['nazwa ciasteczka']`.

```
<!--index.php-->
<?php
    setcookie("imie", "Mateusz");
    setcookie("data", "2021-04-15", time() + 60 * 60 * 24 * 14);
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Przykład i implementacja sesji</title>
</head>
<body>
    <div>
        <?php
            echo "Cookie imie = " . $_COOKIE["imie"] . "<br/>";
            echo "Cookie data = " . $_COOKIE["data"] . "<br/>";
        ?>
    </div>
</body>
</html>
```

Step 3

Napiszmy bardzo prosty mechanizm zapamiętywania danych użytkownika na naszej stronie. Będzie się on składał z następujących plików:

- `index.php` - skrypt zarządzający
- `form.html` - kod formularza
- `header.html` - kod nagłówka strony
- `footer.html` - kod stopki strony

```
<!--form.html-->
<form method="post" action="index.php" xmlns="http://www.w3.org/1999/xhtml">
<div>
    <label>Wprowadź imię i nazwisko: </label>
    <input type="text" name="nazwa"/><br/>
    <input type="submit"/>
</div>
</form>
```

```
<!--header.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Zapamiętywanie danych</title>
</head>
```

```
<!--footer.html-->
<body>

</body>
<footer>Stopka</footer>
</html>
```

```
<!--index.php-->
<?php
    if(!isset($_COOKIE['nazwa']) && !isset($_POST['nazwa'])){
        include("header.html");
        include("form.html");
        include("footer.html");
    }
    else if(isset($_POST['nazwa'])){
        setcookie("nazwa",$_POST['nazwa'],time() + 60 * 60 * 24 * 365 );
        include("header.html");
        echo "<p>Dziękujemy za podanie danych.</p>";
        include("footer.html");
    }
    else{
        include("header.html");
        echo "Witamy, zostałeś rozpoznany jako {$_COOKIE['nazwa']}." ;
        include("footer.html");
    }
?>
```

Step 4

W PHP został wprowadzony mechanizm identyfikacyjny, tzw. sesje. Pozwala on na śledzenie postępowań danego użytkownika na witrynie od pierwszego połączenia z nią aż do jej opuszczenia. Umożliwia to m.in. prostą obsługę logowania i uwierzytelniania użytkowników, śledzenie preferencji, wyświetlanie spersonalizowanych stron, realizację wirtualnych koszyków itp.

Każda sesja użytkownika ma przypisany własny, **unikalny identyfikator**. Ten identyfikator to losowa liczba generowana przez PHP. Jest ona przechowywana na serwerze i na komputerze użytkownika. Każdej sesji można przypisać zmienne, które będą dostępne przez cały czas jej trwania. Jeśli zatem rozpoczęmy sesję wtedy, gdy użytkownik odwiedzi stronę główną naszej witryny, zmienne sesji będą dostępne na wszystkich podstronach, i to tak długo, aż zostanie ona zakończona. Wszystkie takie zmienne są przechowywane na serwerze, a użytkownik nie ma do nich dostępu. Sesja może być rozpoczęta na kilka sposobów. Pierwszy z nich, to po prostu wywołanie funkcji `session_start`. Dobra jest umieścić wywołanie `session_start` na początku każdego skryptu korzystającego z sesji. Funkcja `session_start` nie przyjmuje żadnych parametrów, a zatem jej wywołanie ma postać:

```
session_start();
```

W jednym skrypcie `session_start` należy **wywołać tylko raz**. Ponowne wywołanie będzie zignorowane. W celu zakończenia sesji należy wywołać funkcję `session_destroy`. Usuwa ona zasoby powiązane z sesją, nie usuwa jednak zarejestrowanych zmiennych ani cookie zapisanego na komputerze użytkownika. Funkcja `session_destroy` nie przyjmuje argumentów, a jej wywołanie ma postać:

```
session_destroy();
```

Po rozpoczęciu sesji można stosować jej zmienne, które będą dostępne aż do jej zakończenia. Zmienne sesji są zapisywane w globalnej tablicy o nazwie `$_SESSION`, która może być wykorzystywana tak jak każda inna tablica. Aby zatem zapisać zmienną w sesji, należy skorzystać z konstrukcji:

```
$_SESSION['nazwa_zmiennej'] = wartość;
```

Aby natomiast odczytać wartość zmiennej, trzeba odwołać się do indeksu wskazującego jej nazwę, np.:

```
$zmienna=$_SESSION['nazwa_zmiennej'];
```

Gdy trzeba sprawdzić, czy dana zmienna sesji istnieje, stosuje się funkcję `isSet` w postaci:

```
isSet($_SESSION['nazwa_zmiennej'])
```

Jeśli zwrócona wartością będzie `true`, oznacza to, że zmienna jest zarejestrowana, a jeśli `false`, że nie jest.

Zmienne, które zostały zarejestrowane w sesji, należy przed jej **zakończeniem wyrejestrować**. Należy skorzystać z funkcji `unset` w schematycznej postaci:

```
unset($_SESSION['nazwa_zmiennej'])
```

Nastąpi wtedy wyrejestrowanie z sesji oraz usunięcie zmiennej.

```
<!--index.php-->
<?php
    session_start();
    $_SESSION['zmienna_sesji'] = 'abcd';
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Przykład sesji</title>
</head>
<body>
    <div>
        Witamy na stronie. Została tutaj rozpoczęta sesja.<br/>
        Identyfikatorem sesji jest: <?php echo session_id(); ?><br/>
        Została ustawiona zmienna o nazwie: zmienna_sesji<br/>
        Wartością zmiennej zmienna_sesji jest: <?php echo $_SESSION['zmienna_sesji']; ?><br/>
    </div>
</body>
</html>
```

```
<!--index2.php-->
<?php
    session_start();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Implementacja sesji</title>
</head>
<body>
    <div>
        Witamy na drugiej stronie sesji.<br/>
        Identyfikatorem sesji jest: <?php echo session_id(); ?><br/>
        Została ustawiona zmienna o nazwie: zmienna_sesji<br/>
        Wartością zmiennej zmienna_sesji jest: <?php echo $_SESSION['zmienna_sesji']; ?><br/>
        <a href="index3.php">Następna strona</a>
    </div>
</body>
</html>
```

```

<!--index3.php-->
<?php
    session_start();
    unset($_SESSION['zmienna_sesji']);
    if (isset($_COOKIE[session_name()])){
        setcookie(session_name(),'',time() - 360);
    }
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Przykład sesji</title>
</head>
<body>
<div>
    Witamy na trzeciej stronie sesji.<br/>
    Identyfikatorem sesji jest: <?php echo session_id(); ?><br/>
    Została ustawiona zmienna o nazwie: zmienna_sesji<br/>
    Wartością zmiennej zmienna_sesji jest:
    <?php
        echo $_SESSION['zmienna_sesji'] . "<br/>";
        if (session_destroy())
            echo "Sesja została zakończona";
        else
            echo "Próba zakończenia sesji nie powiodła się";
    ?>
</div>
</body>
</html>

```

Step 5

Sesje mogą być wykorzystane do śledzenia zachowań użytkownika. W prosty sposób możemy na przykład sprawdzić, ile razy w ciągu jednej wizyty odwiedził on poszczególne strony naszego serwisu. Przygotujmy zatem dwa skrypty symulujące dwie podstrony serwisu.

```

<!--index.php-->
<?php
    session_start();
    if (!isset($_SESSION['page1hits'])){
        $_SESSION['page1hits'] = 1;
    }
    else{
        $_SESSION['page1hits']++;
    }
    if (!isset($_SESSION['page2hits']))
        $_SESSION['page2hits'] = 0;
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Przykład i implementacja sesji</title>
</head>
<body>
    <div>
        <?php
            echo "Liczba wizyt na pierwszej stronie: " . $_SESSION['page1hits'] . "<br/>";
            echo "Liczba wizyt na drugiej stronie: " . $_SESSION['page2hits'] . "<br/>";
        ?>
        <a href="index2.php">Następna strona</a>
    </div>
</body>
</html>

```

```

<!--index2.php-->
<?php
session_start();
if (!isset($_SESSION['page2hits'])){
    $_SESSION['page2hits'] = 1;
}
else{
    $_SESSION['page2hits']++;
}
if (!isset($_SESSION['page1hits'])){
    $_SESSION['page1hits'] = 0;
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Przykład i implementacja sesji</title>
</head>
<body>
<div>
    <?php
        echo "Liczba wizyt na pierwszej stronie: " . $_SESSION['page1hits'] . "<br/>";
        echo "Liczba wizyt na drugiej stronie: " . $_SESSION['page2hits'] . "<br/>";
    ?>
    <a href="index.php">Pierwsza strona</a>
</div>
</body>
</html>

```

Step 6

Istnieją dwa podstawowe sposoby wysyłania e-maili za pomocą PHP: wbudowana funkcja poczty `mail()` i zewnętrzne pakiety pocztowe (o których będzie mowa później przy okazji omówienia obiektowego PHP).

Wbudowana funkcja poczty PHP `mail()` jest bardzo prosta, ale zapewnia ograniczoną funkcjonalność w zakresie wysyłania wiadomości e-mail. Nie będzie można dodawać załączników do wiadomości e-mail, a tworzenie pięknego szablonu HTML z osadzonymi obrazami również będzie trudnym zadaniem. Drugą stroną funkcji poczty PHP `mail()` jest to, że wiadomość e-mail jest wysyłana z serwera WWW, co może powodować problemy z dostarczalnością ze względu na kwestie bezpieczeństwa, takie jak **podejrzenie spamu i czarna lista**. Najlepszym sposobem rozwiązania tego problemu jest wysyłanie wiadomości przez serwer **SMTP**, jednak ta funkcjonalność również jest ograniczona. PHP `mail()` zazwyczaj nie pozwala na użycie zewnętrznego serwera **SMTP** i nie obsługuje uwierzytelniania **SMTP**. Dzięki wbudowanej funkcji poczty PHP `mail()` możemy tworzyć proste wiadomości HTML/tekstowe bez załączników i obrazów oraz dodawać kilku odbiorców dzięki parametrowi `$to`. Nadaje się do prostych, głównie tekstowych powiadomień w lokalnym środowisku. Jeśli chcesz komunikować się z użytkownikami swojej aplikacji, lepiej jest zainstalować zewnętrzny pakiet pocztowy.

Składnia poczty PHP `mail()` wygląda następująco:

```
mail($to_email_address,$subject,$message,[ $headers ],[ $parameters ]);
```

Parametr `$to` to odbiorcy wiadomości. Format adresu e-mail może mieć postać `uzytkownik@example.com` lub `Uzytkownik <uzytkownik@example.com>`. Musi być on zgodny ze standardem [RFC 2822 \(https://tools.ietf.org/html/rfc2822\)](https://tools.ietf.org/html/rfc2822). `$subject` oznacza tytuł wiadomości, `$message` zawiera treść wiadomości. Wiersze należy oddzielać znakiem `\r` lub `\n`, a każda linia nie powinna przekraczać 70 znaków. `$headers` służy do uwzględnienia dodatkowych odbiorców wiadomości w DW lub UDW.

Treść wiadomości można zapisać w formacie HTML. W funkcji poczty PHP `mail()` część HTML może wyglądać następująco:

```

<!--index.php-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Wysyłanie wiadomości email</title>
</head>
<body>
    <form method="post" action="">
        <label>Wpisz adres email: </label><input type="email" name="mail"><br/>
        <input type="submit" name="submit">
    </form>
    <?php
        if (isset($_POST['submit'])){
            $to = $_POST['name'];
            $subject = "Simple subject";
            $message = "<b>This is HTML message</b>\n";
            $message .= "<h1>This is headline</h1>";

            $header = "MIME-Version: 1.0\r\n";
            $header .= "Content-type: text/html\r\n";
            $header .= 'From: <info@example.com>' . "\r\n";

            $retval = mail($to,$subject,$message,$header);
            if ($retval == true)
                echo "Message sent successfully";
            else
                echo "Message could not be sent";
        }
    ?>
</body>
</html>

```

Uwaga: Działanie powyższego kodu wymaga poprawnej konfiguracji serwera, wysyłająca adres email. Z poziomu localhost nie da się wysyłać mail do zewnętrznych dostawców (typu gmail czy outlook) ze względu na blokowanie przez nich wiadomości z poziomu localhost. Aby poradzić sobie z tym problemem warto skorzystać z narzędzi zewnętrznych oraz użyć protokołu `smtp`, edytować plik `php.ini`. Na serwerze szuflandia ze względu na ograniczenia dyskowe funkcjonalność ta nie jest dostępna.

5.1 5.1 Teoria

Step 1

Programowanie zorientowane obiektowo uznaje podstawowe połączenie między danymi a kodem, który na nich działa, i umożliwia projektowanie i wdrażanie programów związanych z tym połączeniem. Na przykład system tablic ogłoszeń zwykle śledzi wielu użytkowników. W **proceduralnym języku programowania** każdy użytkownik jest reprezentowany przez strukturę danych i prawdopodobnie istniałby zestaw funkcji, które współpracują z tymi strukturami danych (w celu tworzenia nowych użytkowników, uzyskiwania ich informacji itp.). W języku **OOP** każdy użytkownik jest reprezentowany przez **obiekt** - strukturę danych z dołączonym kodem. Dane i kod wciąż tam są, ale są traktowani jako nierozerłączna całość. Obiekt, będący połączeniem kodu i danych, jest modułową jednostką służącą do tworzenia aplikacji i ponownego wykorzystania kodu. W tym projekcie tablicy ogłoszeń obiekty mogą reprezentować nie tylko użytkowników, ale także wiadomości i wątki. Obiekt użytkownika ma nazwę użytkownika i hasło dla tego użytkownika oraz kod identyfikujący wszystkie wiadomości tego autora. Obiekt wiadomości wie, do którego wątku należy, i ma kod do opublikowania nowej wiadomości, odpowiedzi na istniejącą wiadomość i wyświetlenia wiadomości. Obiekt wątku jest zbiorem obiektów wiadomości i zawiera kod wyświetlący indeks wątku. Jest to jednak tylko jeden sposób na podzielenie niezbędnej funkcjonalności na obiekty. Na przykład w alternatywnym projekcie kod wysyłający nową wiadomość znajduje się w obiekcie użytkownika, a nie w obiekcie wiadomości. Projektowanie systemów obiektowych to złożony temat i napisano na ten temat wiele książek. Dobra wiadomość jest taka, że bez względu na to, jak zaprojektujemy swój system, można go zaimplementować w PHP.

Obiekt jest instancją (lub wystąpieniem) **klasy**. W tym przypadku jest to rzeczywista struktura danych użytkownika z dołączonym kodem. Obiekty i klasy są trochę podobne do wartości i typów danych. Dane powiązane z obiektem nazywane są jego **właściwościami**. Funkcje skojarzone z obiektem nazywane są jego **metodami**. Definiując klasę, definiowane są nazwy, właściwości oraz metody. Debugowanie i konserwacja programów jest znacznie łatwiejsze, podczas używania OOP, ponieważ stosowany jest mechanizm **hermetyzacji**. Chodzi o to, że klasa udostępnia określone metody (interfejs) kodowi, który używa jej obiektów, więc zewnętrzny kod nie ma bezpośredniego dostępu do struktur danych tych obiektów. Debugowanie jest więc łatwiejsze, ponieważ wiemy, gdzie szukać błędów - jedyny kod, który zmienia strukturę danych obiektu, znajduje się w klasie - a konserwacja jest łatwiejsza, ponieważ możemy wymieniać implementacje klasy bez zmiany kodu który używa tej klasy, o ile utrzymujemy ten sam interfejs. Każdy nietrywialny projekt zorientowany obiektowo prawdopodobnie obejmuje mechanizm **dziedziczenia**. Jest to sposób definiowania nowej klasy przez stwierdzenie, że przypomina ona istniejącą klasę, ale ma pewne nowe właściwości i metody. Oryginalna klasa jest nazywana **nadklassą** (lub klasą nadzczną lub podstawową), a nowa klasa jest nazywana **podklassą** (lub klasą pochodną). Dziedziczenie jest formą ponownego wykorzystania kodu - kod nadklasy jest używany ponownie zamiast kopiowania i wklejania do podklasy. Wszelkie ulepszenia lub modyfikacje nadklasy są automatycznie przekazywane do podklasy.

Tworzenie (lub tworzenie instancji) obiektów i używanie ich jest o wiele łatwiejsze niż definiowanie klas obiektów. Aby stworzyć obiekt danej klasy, użyj nowego słowa kluczowego `new`:

```
<?php  
    $object = new Nazwa_klasy;  
?>
```

Zakładając, że klasa `Person` została zdefiniowana, oto jak utworzyć obiekt `Person`:

```
<?php  
    $moana = new Person;  
?>
```

Nie należy nazwy_klasy traktować jako ciąg znaków; będzie to traktowane jako błąd kompilacji.

Niektóre klasy pozwalają na przekazywanie argumentów do tworzenia obiektu tej klasy. Dokumentacja klasy powinna określać, czy przyjmuje argumenty. Jeśli tak, to obiekty tworzymy w następujący sposób (na przykładzie klasy `Person`):

```
<?php  
    $object = new Person("Mateusz", 35);  
?>
```

Nazwa klasy nie musi być zakodowana na stałe w programie. Nazwę klasy możesz podać za pomocą zmiennej:

```
<?php
    $class = "Person";
    $object = new $class;
    // poniższa linia jest równoważna
    $object = new Person;
?>
```

Zmienne zawierające odniesienia do obiektów są zwykłymi zmiennymi - można ich używać w taki sam sposób, jak innych zmiennych. Należy zwrócić uwagę, że zmienne działają z obiektami, co pokazuje kolejny przykład:

```
<?php
    $account = new Account;
    $object = "account";
    ${$object}->init(50000, 1.10); // to samo co $account->init
?>
```

Gdy mamy już obiekt, możemy użyć notacji `->` (jak w przykładzie powyżej), aby uzyskać dostęp do metod i właściwości obiektu:

```
$object->propertname $object->methodname([arg, ... ])
```

```
<?php
    echo "Mateusz is {$person->age} years old.\n"; // wywołanie atrybutu danych
    $person->birthday(); // wywołanie metody
    $person->setAge(21); // wywołanie metody z parametrami
?>
```

Metody działają tak samo jak funkcje (tylko w odniesieniu do danego obiektu), więc mogą przyjmować argumenty i zwracać wartość:

```
<?php
    $clan = $moana->family("extended");
?>
```

W definicji klasy można określić, które metody i właściwości są publicznie dostępne, a które są dostępne tylko z poziomu samej klasy, przy użyciu modyfikatorów dostępu publicznego (`public`) i prywatnego (`private`). Ich użycia zapewnia nam właściwie mechanizm hermetyzacji. **Metoda statyczna** to metoda wywoływana w klasie, a nie w obiekcie. Takie metody nie mają dostępu do właściwości. Nazwa metody statycznej to nazwa klasy, po której następują dwa dwukropki (`::`) i nazwa funkcji.

```
<?php
    HTML::p("Hello, world");
?>
```

Deklarując klasę, definiuje się, które właściwości i metody są statyczne przy użyciu właściwości dostępu statycznego. Po utworzeniu obiekty są przekazywane przez odniesienie - to znaczy zamiast kopowania wokół samego obiektu (przedsięwzięcie pochłaniające czas i pamięć), zamiast tego przekazywane jest odniesienie do obiektu. Na przykład:

```
<?php
    $f = new Person("Pua", 75);
    $b = $f; // $b and $f point at same object
    $b->setName("Hei Hei");
    printf("%s and %s are best friends.\n", $b->getName(), $f->getName()); //Hei Hei and Hei Hei are best
friends.
?>
```

Jeśli chcemy stworzyć prawdziwą kopię obiektu, należy użyć operatora `clone`:

```
<?php
    $f = new Person("Pua", 35);
    $b = clone $f;
    $b->setName("Hei Hei");
    printf("%s and %s are best friends.\n", $b->getName(), $f->getName()); //Pua and Hei Hei are best
friends.
?>
```

Kiedy używamy operatora `clone` do tworzenia kopii obiektu i ta klasa deklaruje metodę `__clone()`, ta metoda jest wywoływana w nowym obiekcie natychmiast po jego sklonowaniu. Możemy tego użyć w przypadkach, gdy obiekt przechowuje zasoby zewnętrzne (takie jak uchwyty plików) do tworzenia nowych zasobów, zamiast kopiować istniejące.

Step 2

Aby zaprojektować program lub bibliotekę kodu w sposób obiektowy, musimy zdefiniować własne klasy, używając słowa kluczowego `class`. Definicja klasy zawiera nazwę klasy oraz właściwości i metody klasy. Nazwy klas nie uwzględniają wielkości liter i muszą być zgodne z regułami dotyczącymi identyfikatorów PHP. Między innymi zastrzeżona jest nazwa klasy `stdClass`. Oto składnia definicji klasy:

```
<?php
    class classname [ extends baseclass ] [ implements interfacename , [interfacename, ...] ] {
        [ use traitname, [ traitname, ... ]; ]
        [ visibility $property [ = value ]; ... ]
        [ function functionname (args) [ : type ] {
            // code
        }
        ...
    }
?>
```

Metoda to funkcja zdefiniowana wewnętrz klasy. Chociaż PHP nie nakłada żadnych specjalnych ograniczeń, większość metod działa tylko na danych w obiekcie, w którym znajduje się metoda. Nazwy metod zaczynające się od dwóch podkreśników (`__`) mogą być używane w przyszłości przez PHP, więc zaleca się nie zaczynać nazw metod od tej sekwencji. W ramach metody zmienna `$this` zawiera odniesienie do obiektu, na którym ta metoda została wywołana. Na przykład, jeśli zostanie wywołana linia `$moana->birthday()`, wewnętrz metody `birthday()`, `$this` ma taką samą wartość jak `$moana`. Metody używające zmienną `$this`, uzyskają dostęp do właściwości bieżącego obiektu i wywołują inne metody na tym obiekcie.

Przejdzmy do wcześniej wspomnianej klasie Person. Jego definicja może wyglądać następująco:

```
<?php
    class Person {
        public $name = '';
        function getName() {
            return $this->name;
        }
        function setName($newName) {
            $this->name = $newName;
        }
    }
    $object = new Person();
    $object->setName("Mateusz");
    echo $object->getName();
?>
```

Jak widać, metody `getName()` i `setName()` używają `$this` do uzyskiwania dostępu i ustawiania właściwości `$name` bieżącego obiektu.

Aby zadeklarować metodę jako metodę statyczną, użyj słowa kluczowego `static`. W metodach statycznych zmienna `$this` nie jest zdefiniowana. Na przykład:

```
<?php
    class HTMLStuff {
        static function startTable() {
            echo "<table border=\"1\">\n";
        }
        static function endTable() {
            echo "</table>\n";
        }
    }
    HTMLStuff::startTable();
    // print HTML table rows and columns
    HTMLStuff::endTable();
?>
```

Używając modyfikatorów dostępu, możemy zmienić widoczność metod. Metody, które są dostępne poza metodami w obiekcie, powinny być zadeklarowane jako publiczne (`public`); metody instancji, które mogą być wywoływane tylko przez metody należące do tej samej klasy, powinny być zadeklarowane jako prywatne (`private`). Wreszcie metody zadeklarowane jako chronione (`protected`) mogą być wywoływane tylko z poziomu metod klasy obiektu i metod klas dziedziczących po klasie. Definiowanie widoczności metod klas jest opcjonalne; jeśli widoczność nie jest określona, metoda jest publiczna.

```
<?php
    class Person {
        public $age;
        public function __construct() {
            $this->age = 0;
        }
        public function incrementAge() {
            $this->age += 1;
            $this->ageChanged();
        }
        protected function decrementAge() {
            $this->age -= 1;
            $this->ageChanged();
        }
        private function ageChanged() {
            echo "Age changed to {$this->age}";
        }
    }
    class SupernaturalPerson extends Person {
        public function incrementAge() {
// ages in reverse
            $this->decrementAge();
        }
    }
    $person = new Person;
    $person->incrementAge();
//    $person->decrementAge(); // not allowed
//    $person->ageChanged(); // also not allowed
    $person = new SupernaturalPerson;
    $person->incrementAge(); // calls decrementAge under the hood
?>
```

Możemy również w klasie deklarować metody następująco:

```
<?php
    class Person {
        function takeJob(Job $job) {
            echo "Now employed as a {$job->title}\n";
        }
    }
?>
```

Gdy metoda zwraca wartość, możemy zadeklarować typ wartości zwracanej metody:

```
<?php
    class Person {
        function bestJob(): Job {
            $job = Job("PHP developer");
            return $job;
        }
    }
?>
```

Step 3

W poprzedniej definicji klasy `Person` została jawnie zadeklarowana właściwość `$name`. Deklaracje własności są opcjonalne i po prostu grzecznościowe dla każdego, kto zajmuje się programem. Deklarowanie właściwości jest dobrym stylem PHP, ale w dowolnym momencie można dodać nowe właściwości. Oto wersja klasy `Person`, która ma niezadeklarowaną właściwość `$name`:

```
<?php
    class Person {
        function getName() {
            return $this->name;
        }
        function setName($newName) {
            $this->name = $newName;
        }
    }
    $obj1 = new Person;
    $obj2 = new Person;
    $obj1->setName("Mateusz");
    $obj2->setName(1234);
    echo $obj1->getName() . "\n";
    echo $obj2->getName() . "\n";
?>
```

Możemy przypisać wartości domyślne do właściwości, ale te wartości domyślne muszą być stałymi prostymi:

```
<?php
    class Person {
        public $name = "Mateusz";
        public $age = 0;
//        public $day = 60 * 60 * hoursInDay(); // nie zadziała
        function getName() {
            return $this->name;
        }
        function setName($newName) {
            $this->name = $newName;
        }
    }
    $obj = new Person;
    echo $obj->getName() . "\n" . $obj->age . "\n";
?>
```

Z pomocą modyfikatorów dostępu można zmienić widoczność właściwości. Właściwości, które są dostępne poza zakresem obiektu, powinny być zadeklarowane jako publiczne (`public`); właściwości instancji, do których można uzyskać dostęp tylko za pomocą metod z tej samej klasy, należy zadeklarować jako prywatne (`private`). Wreszcie, do właściwości zadeklarowanych jako chronione (`protected`) można uzyskać dostęp tylko za pomocą metod klasy obiektu i metod klas dziedziczących po klasie.

```
<?php
    class Person {
        protected $rowId = 0;
        public $username = 'Anyone can see me';
        private $hidden = true;
    }
    $obj = new Person;
    echo $obj->username . "\n";
//    echo $obj->$rowId . "\n";
//    echo $obj->$hidden . "\n";
?>
```

Oprócz właściwości instancji obiektów, PHP umożliwia definiowanie właściwości statycznych, które są zmiennymi w klasie obiektów, do których można uzyskać dostęp, odwołując się do właściwości za pomocą nazwy klasy.

```
<?php
    class Person {
        static $global = 23;
    }
    $localCopy = Person::$global;
    echo $localCopy;
?>
```

Jeśli dostęp do właściwości jest uzyskiwany w obiekcie, który nie istnieje, i jeśli metoda `__get()` lub `__set()` jest zdefiniowana dla klasy obiektu, metoda ta ma możliwość pobrania wartości lub ustawienia wartości dla tej właściwości. Na przykład możemy zadeklarować klasę, która reprezentuje dane pobierane z bazy danych, ale możemy nie chcieć pobierać dużych wartości danych - takich jak duże obiekty binarne (BLOB). Jednym ze sposobów realizacji tego byłoby oczywiście utworzenie metod dostępu do właściwości, które odczytują i zapisują dane na żądanie. Inną metodą może być użycie tych metod korzystając z tzw. mechanizmu przeciążania:

```
<?php
    class Person {
        public function __get($property) {
            if ($property === 'biography') {
                $biography = "long text here..."; // would retrieve from database
                return $biography;
            }
        }
        public function __set($property, $value) {
            if ($property === 'biography') {
                // set the value in the database
            }
        }
    }
?>
```

```
<?php
    class Person {
        private $name = "";
        private $age = 0;
        public function __get($property) {
            if ($property === 'name') {
                return $this->name;
            }
            if ($property === 'age'){
                return $this->age;
            }
        }
        public function __set($property, $value) {
            if ($property === 'name') {
                $this->name = $value;
            }
            if ($property === 'age'){
                $this->age === $value;
            }
        }
    }
    $obj = new Person;
    $obj->name = "Mateusz";
    echo $obj->name;
?>
```

Podobnie jak w przypadku stałych globalnych, przypisywanych za pomocą funkcji `define()`, PHP umożliwia przypisywanie stałych w klasie. Podobnie jak właściwości statyczne, dostęp do stałych można uzyskać bezpośrednio przez klasę lub w metodach obiektów przy użyciu notacji własnejszej. Po zdefiniowaniu stałej nie można zmienić jej wartości:

```
<?php
    class PaymentMethod {
        public const TYPE_CREDITCARD = 0;
        public const TYPE_CASH = 1;
    }
    echo PaymentMethod::TYPE_CREDITCARD;
?>
```

Podobnie jak w przypadku stałych globalnych, powszechną praktyką jest definiowanie stałych klas z identyfikatorami wielkimi literami.

Z pomocą modyfikatorów dostępu można zmienić widoczność stałych klas. Stałe klas, które są dostępne poza metodami obiektu, powinny być zadeklarowane jako publiczne; stałe klasy instancji, do której można uzyskać dostęp tylko za pomocą metod z tej samej klasy, należy zadeklarować jako prywatne. Wreszcie, do stałych zadeklarowanych jako chronione można uzyskać dostęp tylko z poziomu metod klasy obiektu i pliku metody klas dziedziczących po klasie. Definiowanie widoczności stałych klas jest opcjonalne; jeśli widoczność nie jest określona, metoda jest publiczna.

```
<?php
    class Person {
        protected const PROTECTED_CONST = false;
        public const DEFAULT_USERNAME = "<unknown>";
        private const INTERNAL_KEY = "ABC1234";
    }
    echo Person::DEFAULT_USERNAME;
?>
```

Step 4

Aby dziedziczyć właściwości i metody z innej klasy, należy użyć słowa kluczowego `extends` w definicji klasy, po którym następuje nazwa klasy bazowej:

```
<?php
    class Person {
        public $name, $address, $age;
    }
    class Employee extends Person {
        public $position, $salary;
    }
?>
```

Klasa `Employee` zawiera właściwości `$position` i `$salary`, a także właściwości `$name`, `$address` i `$age` odziedziczone z klasy `Person`. Jeśli klasa pochodna ma właściwość lub metodę o takiej samej nazwie jak klasa nadzędna, właściwość lub metoda w klasie pochodnej ma pierwszeństwo przed właściwością lub metodą w klasie nadzędnej. Odwołanie do właściwości zwraca wartość właściwości w przypadku elementu podzielnego, podczas gdy odwołanie do metody wywołuje metodę w przypadku elementu podzielnego. Notacja `parent::method()`, służy do uzyskania dostępu do nadpisanej metody w klasie nadzędnej obiektu:

```
<?php
    class Person {
        public $name, $address, $age;
        public function birthday(){
            echo "This is a birthday\n";
        }
    }
    class Employee extends Person {
        public $position, $salary;
        public function birthday()
        {
            parent::birthday();
            echo "This is a birthday from Employee\n";
        }
    }
    $obj = new Person;
    $obj->birthday();
    $obj1 = new Employee;
    $obj1->birthday();
?>
```

Częstym błędem jest zakodowanie na stałe nazwy klasy nadzędnej w wywołaniach nadpisanych metod:

```
<?php
    class Person {
        public $name, $address, $age;
        public function birthday(){
            echo "This is a birthday\n";
        }
    }
    class Employee extends Person {
        public $position, $salary;
        public function birthday()
        {
            Person::birthday();
            echo "This is a birthday from Employee\n";
        }
    }
    $obj = new Person;
    $obj->birthday();
    $obj1 = new Employee;
    $obj1->birthday();
?>
```

Jest to błąd, ponieważ rozpowszechnia wiedzę o nazwie klasy nadzędnej w całej klasie pochodnej. Użycie `parent::` centralizuje wiedzę o klasie nadzędnej w klauzuli `extends`. Jeśli metoda może być podklassą i chcemy mieć pewność, że wywołujemy ją w bieżącej klasie, należy użyć notacji `self::method()`:

```

<?php
    class Person {
        public $name, $address, $age;
        public function birthday(){
            echo "This is a birthday\n";
        }
    }
    class Employee extends Person {
        public $position, $salary;
        public function birthday(){
            echo "This is a birthday from Employee\n";
        }
        public function get_birthday(){
            self::birthday();
        }
    }
    $obj = new Person;
    $obj->birthday();
    $obj1 = new Employee;
    $obj1->get_birthday();
?>

```

Aby sprawdzić, czy obiekt jest instancją określonej klasy lub czy implementuje określony interfejs, należy użyć operatora `instanceof` :

```

<?php
    class Person {
        public $name, $address, $age;
        public function birthday(){
            echo "This is a birthday\n";
        }
    }
    class Employee extends Person {
        public $position, $salary;
        public function birthday(){
            echo "This is a birthday from Employee\n";
        }
        public function get_birthday(){
            self::birthday();
        }
    }
    $obj = new Person;
    if ($obj instanceof Person){
        echo "This is a Person object\n";
    }
?>

```

Step 5

Możemy podać listę argumentów występujących po nazwie klasy podczas tworzenia wystąpienia obiektu:

```
$person = new Person("Fred", 35);
```

Te argumenty są przekazywane do konstruktora klasy, specjalnej funkcji, która inicjuje właściwości klasy. **Konstruktor** to funkcja w klasie o nazwie `__construct()`. Oto konstruktor dla klasy `Person` :

```

<?php
    class Person {
        function __construct($name, $age) {
            $this->name = $name;
            $this->age = $age;
        }
    }
    $obj = new Person("Mateusz", 30);
    echo $obj->name . " " . $obj->age;
?>

```

PHP nie zapewnia automatycznego łańcucha konstruktorów; to znaczy, że jeśli utworzymy instancję obiektu klasy pochodnej, tylko konstruktor z tej klasy jest wywoływany automatycznie. Aby można było wywołać konstruktor klasy nadrzędnej, konstruktor w klasie pochodnej musi jawnie wywołać konstruktor. W tym przykładzie konstruktor klasy `Employee` wywołuje konstruktor `Person`:

```
<?php
class Person {
    public $name, $address, $age;
    function __construct($name, $address, $age) {
        $this->name = $name;
        $this->address = $address;
        $this->age = $age;
    }
}
class Employee extends Person {
    public $position, $salary;
    function __construct($name, $address, $age, $position, $salary) {
        parent::__construct($name, $address, $age);
        $this->position = $position;
        $this->salary = $salary;
    }
}
$obj = new Person("Mateusz", "Gdańsk", 30);
$emp = new Employee($obj->name, $obj->address, $obj->age, "Teacher", 2000);
var_dump($emp);
?>
```

Gdy obiekt zostanie zniszczony, na przykład gdy ostatnie odwołanie do obiektu zostanie usunięte lub osiągnięty zostanie koniec skryptu, wywoływany jest jego **destruktör**. Ponieważ PHP automatycznie czyści wszystkie zasoby, gdy wykraczają poza zakres i pod koniec wykonywania skryptu, ich zastosowanie jest ograniczone. Destruktor to metoda o nazwie `__destruct()`:

```
<?php
class Building {
    function __destruct() {
        echo "A Building is being destroyed!";
    }
}
$obj = new Building;
?>
```

Step 6

Interfejsy umożliwiają definiowanie kontraktów, do których stosuje się klasa; interfejs zawiera prototypy i stałe metod, a każda klasa implementująca interfejs musi zapewniać implementacje dla wszystkich metod w interfejsie. W PHP składnia interfejsu wygląda następująco:

```
interface interfacename {
    [ function functionname();
    ...
]
```

Aby zadeklarować, że klasa implementuje interfejs, należy dołączyć słowo kluczowe `implements` i **dowolną liczbę interfejsów**, oddzielając je przecinkami:

```
<?php
interface Printable {
    function printOutput();
}
class ImageComponent implements Printable {
    function printOutput() {
        echo "Printing an image...";
    }
}
$obj = new ImageComponent();
$obj->printOutput();
?>
```

Interfejs może **dziedziczyć z innych interfejsów** (w tym wielu interfejsów), o ile żaden z interfejsów nie dziedziczy z deklarowanych metod o takiej samej nazwie, jak te zadeklarowane w interfejsie podzielonym.

Trait (cecha) zapewniają mechanizm ponownego wykorzystania kodu poza hierarchią klas. Traity umożliwiają udostępnianie funkcji różnym klasom, które nie mają (i nie powinny) mieć wspólnego przodka w hierarchii klas. W PHP składnia `trait` wygląda następująco:

```
trait traitname [ extends baseclass ] {  
    [ use traitname, [ traitname, ... ]; ]  
    [ visibility $property [ = value ]; ... ]  
    [ function functionname (args) {  
        // code  
    }  
    ...  
}
```

Aby zadeklarować, że klasa powinna zawierać metody `traita`, należy dodać słowo kluczowe `use` i **dowolną liczbę cech** oddzielonych przecinkami:

```
<?php  
trait Logger {  
    public function log($logString) {  
        $className = __CLASS__;  
        echo date("Y-m-d h:i:s", time()) . ": [" . $className . "] {$logString}";  
    }  
}  
class User {  
    use Logger;  
    public $name;  
    function __construct($name = '') {  
        $this->name = $name;  
        $this->log("Created user '{$this->name}'");  
    }  
    function __toString() {  
        return $this->name;  
    }  
}  
class UserGroup {  
    use Logger;  
    public $users = array();  
    public function addUser(User $user) {  
        if (!in_array($this->users, (array)$user)) {  
            $this->users[] = $user;  
            $this->log("Added user '{$user}' to group");  
        }  
    }  
}  
$group = new UserGroup;  
$group->addUser(new User("Franklin"));  
?>
```

Metody zdefiniowane przez `trait` `Logger` są dostępne dla wystąpień klasy `UserGroup` tak, jakby zostały zdefiniowane w tej klasie.

Aby zadeklarować, że `trait` powinien składać się z innych `trait`, należy dodać instrukcję `use` w deklaracji `trait`, a następnie jedną lub więcej nazw `trait` oddzielonych przecinkami, jak pokazano poniżej:

```
<?php
    trait First {
        public function doFirst() {
            echo "first\n";
        }
    }
    trait Second {
        public function doSecond() {
            echo "second\n";
        }
    }
    trait Third {
        use First, Second;
        public function doAll() {
            $this->doFirst();
            $this->doSecond();
        }
    }
    class Combined {
        use Third;
    }
    $object = new Combined;
    $object->doAll();
?>
```

Trait mogą deklarować metody abstrakcyjne. Jeśli klasa używa wielu trait definiujących tę samą metodę, PHP podaje błąd krytyczny. Możesz jednak przesłonić to zachowanie, informując kompilator konkretnie, której implementacji danej metody chcemy użyć. Definiując traity, które zawierają klasę, należy użyć słowa kluczowego insteadof dla każdego konfliktu:

```
<?php
    trait Command {
        function run() {
            echo "Executing a command\n";
        }
    }
    trait Marathon {
        function run() {
            echo "Running a marathon\n";
        }
    }
    class Person {
        use Command, Marathon {
            Marathon::run insteadof Command;
        }
    }
    $person = new Person;
    $person->run();
?>
```

Zamiast wybierać tylko jedną metodę do uwzględnienia, możemy użyć słowa kluczowego as, aby aliasować metodę trait w klasie, włączając ją do innej nazwy. Nadal jednak musimy wyraźnie rozwiązać wszelkie konflikty dotyczące dołączonych trait. Na przykład:

```
<?php
    trait Command {
        function run() {
            echo "Executing a command";
        }
    }
    trait Marathon {
        function run() {
            echo "Running a marathon";
        }
    }
    class Person {
        use Command, Marathon {
            Command::run as runCommand;
            Marathon::run insteadof Command;
        }
    }
    $person = new Person;
    $person->run();
    $person->runCommand();
?>
```

Step 7

PHP zapewnia również mechanizm deklarowania, że pewne metody w klasie muszą być implementowane przez podklasy - implementacja tych metod nie jest zdefiniowana w klasie nadrzędnej. W takich przypadkach mówimy, że są to **metody abstrakcyjne**; dodatkowo, jeśli klasa zawiera metody zdefiniowane jako abstrakcyjne, to musimy również zadeklarować klasę jako klasę abstrakcyjną:

```
<?php
abstract class Component {
    abstract function printOutput();
}
class ImageComponent extends Component {
    function printOutput() {
        echo "Pretty picture";
    }
}
$obj = new ImageComponent();
$obj->printOutput();
?>
```

Nie można utworzyć instancji klas abstrakcyjnych. W przeciwieństwie do niektórych języków, PHP nie pozwala na zapewnienie domyślnej implementacji metod abstrakcyjnych. Trait mogą również deklarować metody abstrakcyjne. Klassy zawierające trait definiującą metodę abstrakcyjną muszą implementować tę metodę:

```
<?php
trait Sortable {
    abstract function uniqueId();
    function compareById($object) {
        return ($object->uniqueId() < $this->uniqueId()) ? -1 : 1;
    }
}
class Bird {
    use Sortable;
    function uniqueId() {
        return 2;
    }
}
// to wywoła błąd kompilacji
// class Car {
//     use Sortable;
// }
class Car {
    use Sortable;
    public function uniqueId() {
        return 10;
    }
}
$bird = new Bird;
$car = new Car;
$comparison = $bird->compareById($car);
echo $comparison;
?>
```

Podczas implementowania metody abstrakcyjnej w klasie potomnej sygnatury metod muszą być zgodne - to znaczy muszą przyjmować taką samą liczbę wymaganych parametrów, a jeśli którykolwiek z parametrów ma wskazówki dotyczące typu, te wskazówki muszą być zgodne. Ponadto metoda musi mieć taką samą lub mniej ograniczoną widoczność.

Podczas tworzenia pozorowanych obiektów (ang. mocks) do testów przydatne jest tworzenie **anonimowych klas**. Klasa anonimowa zachowuje się tak samo, jak każda inna klasa, z wyjątkiem tego, że nie posiada ona nazwy (co oznacza, że nie można jej bezpośrednio utworzyć):

```

<?php
    class Person {
        public $name = '';
        function getName() {
            return $this->name;
        }
    }
    $anonymous = new class() extends Person {
        public function getName() {
            return "Moana";
        }
    };
    echo $anonymous->getName();
?>

```

Step 8

Introspekcja (ang. introspection) to zdolność programu do zbadania cech obiektu, takich jak jego nazwa, klasa nadzędna (jeśli istnieje), właściwości i metody. Dzięki introspekcji możemy pisać kod działający na dowolnej klasie lub obiekcie. Nie musimy wiedzieć, które metody lub właściwości są zdefiniowane podczas pisania kodu; zamiast tego możemy odkryć te informacje w czasie wykonywania, co umożliwia pisanie ogólnych debugerów, serializatorów, profilerów i tym podobnych.

Aby określić, czy klasa istnieje, należy użyć funkcji `class_exists()`, która pobiera ciąg i zwraca wartość logiczną. Alternatywnie możemy również użyć funkcji `get_declared_classes()`, która zwraca tablicę zdefiniowanych klas i sprawdza, czy nazwa klasy znajduje się w zwróconej tablicy:

```

<?php
    class Person{

    }
    class Employee extends Person{

    }
    echo class_exists("Person") . "\n";
    $classes = get_declared_classes();
    var_dump($classes);
    $doesClassExist = in_array("Employee", (array)$classes) . "\n";
?>

```

Możemy pobrać metody i właściwości, które istnieją w klasie (w tym te, które są dziedziczone z nadklas) za pomocą funkcji `get_class_methods()` i `get_class_vars()`. Te funkcje pobierają nazwę klasy i zwracają tablicę:

```

<?php
    class Person{
        private $name = '';

        public function getName(): string
        {
            return $this->name;
        }

        public function setName(string $name): void
        {
            $this->name = $name;
        }
    }

    class Employee extends Person{
        private $salary = 0;
        public $yob = 0;
        public function getSalary(): int
        {
            return $this->salary;
        }

        public function setSalary(int $salary): void
        {
            $this->salary = $salary;
        }
    }

    var_dump(get_class_methods("Employee"));
    var_dump(get_class_vars("Employee"));

?>

```

Nazwa klasy może być zmienną zawierającą nazwę klasy, nieosłonięte słowo lub ciąg znaków w cudzysłowie:

```

<?php
    class Person{
        private $name = '';

        public function getName(): string
        {
            return $this->name;
        }

        public function setName(string $name): void
        {
            $this->name = $name;
        }
    }

    class Employee extends Person{
        private $salary = 0;
        public $yob = 0;
        public function getSalary(): int
        {
            return $this->salary;
        }

        public function setSalary(int $salary): void
        {
            $this->salary = $salary;
        }
    }

    $str = "Employee";
    var_dump(get_class_vars("Employee"));
    var_dump(get_class_vars(Employee::class));
    var_dump(get_class_vars($str));

?>

```

Tablica zwrocona przez funkcje `get_class_methods()` to prosta lista nazw metod. Tablica asocjacyjna zwracana przez `get_class_vars()` odwzorowuje nazwy właściwości na wartości, a także zawiera właściwości dziedziczone.

```

<?php
    class Person{
        private $name = '';
        public $secondname;
        public function getName(): string
        {
            return $this->name;
        }

        public function setName(string $name): void
        {
            $this->name = $name;
        }
    }
    class Employee extends Person{
        private $salary = 0;
        public $yob = 0;
        public $a;
        public function getSalary(): int
        {
            return $this->salary;
        }

        public function setSalary(int $salary): void
        {
            $this->salary = $salary;
        }
    }
    $str = "Employee";
    var_dump(get_class_vars("Employee"));
?>

```

Aby odnaleźć nazwę klasy nadzędnej, należy użyć metody `get_parent_class()`.

```

<?php
    class A {}
    class B extends A {}
    $obj = new B;
    echo get_parent_class($obj);
    echo get_parent_class(B::class);
?>

```

Aby pobrać nazwę klasy, do której należy obiekt, należy najpierw upewnić się, że jest to obiekt, korzystając z funkcji `is_object()`:

```

<?php
    class Person{
        private $name = '';
        public $secondName = '';

        public function setName(string $name): void
        {
            $this->name = $name;
        }

        public function getName(): string
        {
            return $this->name;
        }
    }
    $var = new Person();
    //    $var = "Person";
    if (is_object($var))
        echo get_class($var);
    else
        echo "\$var is not an object!!!"

?>

```

Przed wywołaniem metody na obiekcie można upewnić się, że ona istnieje, używając funkcji `method_exists()`:

```

<?php
    class Person{
        private $name = '';
        public $secondName = '';

        public function setName(string $name): void
        {
            $this->name = $name;
        }

        public function getName(): string
        {
            return $this->name;
        }
    }
    $var = new Person();
    if (method_exists($var, "getName"))
        echo "Method getName exists";
    else
        echo "Method getName not exists";

?>

```

Tak jak `get_class_vars()` zwraca tablicę właściwości dla klasy, `get_object_vars()` zwraca tablicę właściwości ustawionych w obiekcie:

```

<?php
    class Person{
        private $name = '';
        public $secondName = '';

        public function setName(string $name): void
        {
            $this->name = $name;
        }

        public function getName(): string
        {
            return $this->name;
        }
    }
    $var = new Person();
    $var->setName("Mateusz");
    $var->secondName = "Miotk";
    var_dump(get_class_vars("Person"));
    var_dump(get_object_vars($var));

?>

```

Poniżej znajduje się kod, który wypisuje wszystkie zadeklarowane klasy, ich metody statyczne i wartości:

```

<?php
function displayClasses() {
    $classes = get_declared_classes();
    foreach ($classes as $class) {
        echo "Showing information about {$class}<br />";
        $reflection = new ReflectionClass($class);
        $isAnonymous = $reflection->isAnonymous() ? "yes" : "no";
        echo "Is Anonymous: {$isAnonymous}<br />";
        echo "Class methods:<br />";
        $methods = $reflection->getMethods(ReflectionMethod::IS_STATIC);
        if (!count($methods)) {
            echo "<i>None</i><br />";
        }
        else {
            foreach ($methods as $method) {
                echo "<b>{$method}</b>()<br />";
            }
        }
        echo "Class properties:<br />";
        $properties = $reflection->getProperties();
        if (!count($properties)) {
            echo "<i>None</i><br />";
        }
        else {
            foreach(array_keys($properties) as $property) {
                echo "<b>\${$property}</b><br />";
            }
        }
        echo "<hr />";
    }
}
displayClasses();
?>

```

Step 9

Serializacja (ang. *serializing*) obiektu oznacza konwersję go do reprezentacji strumienia bajtowego, która może być przechowywana w pliku. Jest to przydatne w przypadku trwałych danych; na przykład sesje PHP automatycznie zapisują i przywracają obiekty. Serializacja w PHP jest w większości automatyczna - wymaga niewielkiej dodatkowej pracy, poza wywołaniem funkcji `serialize()` oraz `unserialize()`:

```

$encoded = serialize(something);
$something = unserialize(encoded);

```

Serializacja jest najczęściej używana w sesjach PHP, którą ją obsługują. Wystarczy, że wskażemy PHP, które zmienne mają być śledzone, a zostaną one automatycznie zachowane między wizytami na stronach w witrynie. Jednak sesje nie są jedynym zastosowaniem serializacji - jeśli chcemy zaimplementować własną formę trwałych obiektów, `serialize()` i `unserialize()` są naturalnym wyborem. Klasa obiektu musi zostać zdefiniowana, zanim będzie można przeprowadzić odserializację. Próba odserializacji obiektu, którego klasa nie została jeszcze zdefiniowana, umieszcza obiekt w `stdClass`, co czyni go prawie bezużytecznym. Jedną z praktycznych konsekwencji tego jest to, że jeśli używamy sesji PHP do automatycznej serializacji i odserializacji obiektów, to musimy dołączyć plik zawierający definicję klasy obiektu na każdej stronie w serwisie.

```

include "object_definitions.php"; // load object definitions
session_start(); // load persistent variables
?>
<html>...

```

PHP ma posiada dwie metody podczas procesu serializacji i odserializacji: `__sleep()` i `__wakeup()`. Te metody służą do powiadamiania obiektów, że są serializowane lub nieserializowane. Obiekty można serializować, jeśli nie mają tych metod; jednak nie zostaną powiadomieni o tym procesie. Metoda `__sleep()` jest wywoływana na obiekcie tuż przed serializacją; może wykonać dowolne czyszczenie niezbędne do zachowania stanu obiektu, takie jak zamknięcie połączeń z bazą danych, wypisywanie niezapisanych trwałych danych i tak dalej. Powinien zwrócić tablicę zawierającą nazwy członków danych, które mają zostać zapisane w strumieniu bajtowym. Jeśli zwróciśmy pustą tablicę, żadne dane nie zostaną zapisane. Metoda `__wakeup()` jest wywoływana na obiekcie natychmiast po utworzeniu obiektu ze strumienia bajtowego. Metoda może wykonywać dowolne działania, których wymaga,

na przykład ponownie otwieranie połączenia z bazą danych i wykonywać inne zadania inicjujące. Poniżej znajduje się definicja klasy `Log`, która udostępnia dwie przydatne metody: `write()` do dołączania komunikatu do pliku dziennika i `read()` do pobrania aktualnej zawartości pliku dziennika. Używa `__wakeup()`, aby ponownie otworzyć plik dziennika i `__sleep()`, aby zamknąć plik dziennika.

```
//log.php
<?php
    class Log {
        private $filename;
        private $fh;
        function __construct($filename) {
            $this->filename = $filename;
            $this->open();
        }
        function open() {
            $this->fh = fopen($this->filename, 'a') or die("Can't open {$this->filename}");
        }
        function write($note) {
            fwrite($this->fh, "{$note}\n");
        }
        function read() {
            return join('', file($this->filename));
        }
        function __wakeup(): void {
            $this->filename = 'persistent_log';
            $this->open();
        }
        function __sleep() {
            fclose($this->fh);
            return ["filename" => $this->filename];
        }
    }
?>
```

Strona główna HTML poniżej używa klasy `Log` i sesji PHP do utworzenia trwałej zmiennej dziennika, `$logger`.

```
//front.php
<?php
    include_once "log.php";
    session_start();
?>
<html><head><title>Front Page</title></head>
<body>
<?php
    $now = strftime("%c");
    if (!isset($_SESSION['logger'])) {
        $logger = new Log("persistent_log");
        $_SESSION['logger'] = $logger;
        $logger->write("Created $now");
        echo "<p>Created session and persistent log object.</p>";
    }
    else {
        $logger = $_SESSION['logger'];
    }
    $logger->write("Viewed first page {$now}");
    echo "<p>The log contains:</p>";
    echo nl2br($logger->read());
?>
<a href="next.php">Move to the next page</a>
</body></html>
```

Plik `next.php`, jest stroną HTML. Podążając za linkiem ze strony głównej do tej strony wyzwala wczytywanie trwałego obiektu `$logger`. Wywołanie `__wakeup()` ponownie otwiera plik dziennika, dzięki czemu obiekt jest gotowy do użycia.

6.1 6.1 Teoria

Step 1

Baza danych to uporządkowany zbiór rekordów lub danych przechowywanych w systemie komputerowym i zorganizowany w taki sposób, aby można było je szybko przeszukiwać, a informacje można było szybko odzyskać. **SQL** oznacza **Structured Query Language**. Ten język jest luźno oparty na języku angielskim i jest również używany w innych bazach danych, takich jak **Oracle** i **Microsoft SQL Server**. Został zaprojektowany, aby umożliwić proste żądania z bazy danych za pomocą poleceń, takich jak:

```
SELECT title FROM publications WHERE author = 'Charles Dickens';
```

Do najpopularniejszego systemu bazodanowego dla serwerów WWW należy system **MySQL** (<https://www.mysql.com/> (<https://www.mysql.com/>)). Opracowana w połowie lat 90. XX wieku jest obecnie dojrzałą technologią, która obsługuje wiele z najczęściej odwiedzanych obecnie miejsc w Internecie. Jednym z powodów jego sukcesu (tak jak w przypadku PHP) jest fakt, że jest on darmowy. Ale jest też niezwykle potężny i wyjątkowo szybki - może działać nawet na najbardziej podstawowym sprzęcie i prawie nie wpływa na zasoby systemowe. MySQL jest również wysoce skalowalny, co oznacza, że może rosnąć wraz z witryną.



Baza danych MySQL zawiera jedną lub więcej tabel, z których każda zawiera rekordy lub wiersze. W tych wierszach znajdują się różne kolumny lub pola, które zawierają same dane. Poniżej znajduje się przykład bazy danych pięciu publikacji z wyszczególnieniem autora, tytułu, rodzaju i roku publikacji.

Author	Title	Type	Year
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
William Shakespeare	Romeo and Juliet	Play	1594

Każdy wiersz w tabeli jest taki sam jak wiersz w tabeli MySQL, kolumna w tabeli odpowiada kolumnie w MySQL, a każdy element w wierszu jest taki sam jak pole MySQL. Do głównych terminów związanych z obsługą baz danych (w kontekście MySQL) należą:

- **Baza danych** - Ogólny kontener dla kolekcji danych MySQL
- **Tabela** - Kontener podrzędny w bazie danych, który przechowuje rzeczywiste dane
- **Krotka (wiersz)** - Pojedynczy rekord w tabeli, który może zawierać kilka pól
- **Kolumna** - Nazwa pola w wierszu

Instalacja serwera MySQL w systemie Ubuntu wygląda następująco:

1. Najpierw należy zaktualizować pakiet, korzystając z polecenia `sudo apt update`;
2. Następnie dokonujemy instalacji serwera za pomocą polecenia `sudo apt install mysql-server-8.0`.

Spowoduje to zainstalowanie MySQL w wersji 8 i wyżej, ale nie wyświetli monitu o ustawienie hasła ani żadnych innych zmian w konfiguracji.

```
mateusz@mateusz-HP-ProBook-650-G4:~$
```

█

W przypadku nowych instalacji warto uruchomić skrypt bezpieczeństwa (ang. security script). Spowoduje to zmianę niektórych mniej bezpiecznych opcji domyślnych, takich jak zdalne logowanie do roota i przykładowi użytkownicy. Aby uruchomić skrypt bezpieczeństwa należy skorzystać z polecenia:

```
sudo mysql_secure_installation
```

Powyższe polecenie poprowadzi przez serię monitów, w których możemy wprowadzić pewne zmiany w opcjach bezpieczeństwa instalacji MySQL. Pierwszy monit zapyta, czy chcemy skonfigurować wtyczkę `Validate Password`, której można użyć do przetestowania siły hasła MySQL. Niezależnie od wyboru następnym pytaniem będzie ustawienie hasła dla użytkownika `root` MySQL. Następnie można nacisnąć klawisz ENTER, aby zaakceptować wartości domyślne dla wszystkich kolejnych pytań. Spowoduje to usunięcie anonimowych użytkowników i testową bazę danych, wyłączenie zdalnego logowania roota i załadowanie tych nowych reguł, aby MySQL natychmiast uwzględnił wprowadzone zmiany.

```
mateusz@mateusz-HP-ProBook-650-G4:~$
```

█

Aby zainicjować katalog danych MySQL, należy użyć `mysql_install_db` dla wersji wcześniejszych niż 5.7.6 i `mysqld --initialize` dla wersji 5.7.6 i nowszych. Jeśli jednak zainstalowany został MySQL tak jak powyżej, katalog danych został zainicjowany automatycznie; Jeśli mimo wszystko spróbujemy uruchomić polecenie, zobaczymy następujący błąd:

```
mysqld: Can't create directory '/var/lib/mysql/' (OS errno 17 - File exists)
2021-05-13T09:38:14.757885Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.23-0ubuntu0.20.10.1) initializing of server in progress as process 273800
2021-05-13T09:38:14.758794Z 0 [ERROR] [MY-010187] [Server] Could not open file '/var/log/mysql/error.log' for error logging: Permission denied
2021-05-13T09:38:14.758815Z 0 [ERROR] [MY-013236] [Server] The designated data directory /var/lib/mysql/ is unusable. You can remove all files that the server added to it.
2021-05-13T09:38:14.758821Z 0 [ERROR] [MY-010119] [Server] Aborting
2021-05-13T09:38:14.758928Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.23-0ubuntu0.20.10.1) (Ubuntu).
```

Mimo to, że zostało ustawione hasło dla użytkownika root MySQL, użytkownik ten nie jest skonfigurowany do uwierzytelniania za pomocą hasła podczas łączenia się z powłoką MySQL. W systemach Ubuntu z MySQL 5.7 (i nowszymi wersjami) `root` użytkownika MySQL jest domyślnie ustawiony na uwierzytelnianie za pomocą wtyczki `auth_socket`, a nie za pomocą hasła. Pozwala to w wielu przypadkach na większe bezpieczeństwo i użyteczność, ale może również skomplikować sytuację, gdy trzeba zezwolić zewnętrznemu programowi (np. PhpMyAdmin) na dostęp do użytkownika. Aby użyć hasła do połączenia się z MySQL jako root, musimy zmienić jego metodę uwierzytelniania z `auth_socket` na `mysql_native_password`. Aby to zrobić, należy otworzyć monit MySQL w terminalu:

```
sudo mysql
```

Następnie należy sprawdzić, która metoda uwierzytelniania używa każde z kont użytkowników MySQL, używając następującego polecenia:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

Wynik prawdopodobnie będzie wyglądał następująco:

user	authentication_string	plugin	host
root	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	auth_socket	localhost
mysql.session	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	mysql_native_password	localhost
mysql.sys	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	mysql_native_password	localhost
debian-sys-maint	*CC744277A401A7D25BE1CA89AFF17BF607F876FF	mysql_native_password	localhost

4 rows in set (0.00 sec)

W tym przykładzie widać, że użytkownik root faktycznie uwierzytelnia się za pomocą wtyczki `auth_socket`. Aby skonfigurować konto `root` do uwierzytelniania za pomocą hasła, należy uruchomić następujące polecenie `ALTER USER`. Należy pamiętać, że hasło powinno być silne oraz, że to polecenie zmieni hasło roota ustawione podczas użycia polecenia `mysql_secure_installation`.

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'silneHaslo123@';
```

Następnie należy uruchomić polecenie `FLUSH PRIVILEGES`, które nakazują serwerowi ponowne załadowanie tabel grantów i wprowadzenie nowych zmian w życie:

```
FLUSH PRIVILEGES;
```

Możemy sprawdzić ponownie metody uwierzytelniania stosowane przez każdego z użytkowników, aby upewnić się, że `root` nie jest już uwierzytelniany za pomocą wtyczki `auth_socket`:

user	authentication_string	plugin	host
root	*3636DACC8616D997782ADD0839F92C1571D6D78F	mysql_native_password	localhost
mysql.session	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	mysql_native_password	localhost
mysql.sys	*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE	mysql_native_password	localhost
debian-sys-maint	*CC744277A401A7D25BE1CA89AFF17BF607F876FF	mysql_native_password	localhost

4 rows in set (0.00 sec)

Po wyjściu z konsoli (po dwukrotnym poleceniu `exit`), nie będziemy mogli już wejść do `mysql` za pomocą polecenia `sudo mysql`, ale za pomocą polecenia (dla konta `root`):

```
mysql -u root -p
```

Step 2

Niezależnie od tego, jak system MySQL został zainstalowany, powinien zacząć działać automatycznie. Aby to przetestować, możemy sprawdzić jego stan za pomocą polecenia:

```
systemctl status mysql.service
```

Wynik polecenia będzie mniej więcej wyglądał następująco:

```
mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:>)
   Active: active (running) since Thu 2021-05-13 11:28:40 CEST; 31min ago
     Main PID: 268387 (mysqld)
       Status: "Server is operational"
         Tasks: 39 (limit: 18955)
        Memory: 337.2M
      CGroup: /system.slice/mysql.service
              └─268387 /usr/sbin/mysqld

maj 13 11:28:40 mateusz-HP-ProBook-650-G4 systemd[1]: Starting MySQL Community >
maj 13 11:28:40 mateusz-HP-ProBook-650-G4 systemd[1]: Started MySQL Community S
```

Jeśli MySQL nie działa, możemy go uruchomić za pomocą polecenia:

```
sudo systemctl start mysql
```

W celu dodatkowego sprawdzenia możemy spróbować połączyć się z bazą danych za pomocą narzędzia `mysqladmin`, które jest klientem umożliwiającym uruchamianie poleceń administracyjnych. Na przykład to polecenie mówi, aby połączyć się z MySQL jako root (-u root), poprosić o hasło (-p) i zwrócić wersję:

```
sudo mysqladmin -p -u root version
```

Aby wyświetlić bazy danych w MySQL, należy z poziomu konsoli wpisać polecenie:

```
SHOW DATABASES;
```

Po świeżej instalacji wynik wygląda następująco:

```
+-----+
| Database           |
+-----+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+-----+
4 rows in set (0.01 sec)
```

Znak średnika jest używany przez MySQL do oddzielania lub kończenia poleceń. Jeśli zapomnimy go wprowadzić, MySQL wyświetli monit i zaczeka, aż użytkownik zakończy polecenie. Wymagany średnik został wprowadzony do składni, aby umożliwić wprowadzanie poleceń wielowierszowych, co może być wygodne, ponieważ niektóre polecenia są dość długie. Pozwala również na wydanie więcej niż jednego polecenia naraz, umieszczając średnik po każdym z nich. Interpreter pobiera je wszystkie w pakiecie po naciśnięciu klawisza Enter (lub Return) i wykonuje je po kolej. Bardzo często pojawia się w tej sytuacji znak zachęty MySQL w celu oczekiwania wpisania znaku średnika. Istnieje sześć różnych podpowiedzi, które może wyświetlić MySQL, które zobrazuje poniższa tabela:

Znak zachęty MySQL	Znaczenie
<code>mysql></code>	Gotowy i czekający na polecenie
<code>-></code>	Oczekивание на следующий ввод команды
<code>'></code>	Oczekивание na следующую строку ввода, начинаящуюся с одинарного кавычка
<code>"></code>	Oczekивание на следующую строку ввода, начинаящуюся с двойного кавычка

Znak zachęty MySQL	Znaczenie
`>	Oczekiwanie na następną linię łańcucha rozpoczęte od `
/*>	Oczekiwanie na następny wiersz komentarza zaczynający się od /*

Jeśli jesteśmy w trakcie wprowadzania polecenia i zdecydujemy się na nie wykonywanie go, nie należy nacisnąć kombinacji Ctrl-C ! To zamknie program. Zamiast tego możemy wpisać \c i nacisnąć klawisz ENTER. Kiedy wpiszemy znak \c , to MySQL zignoruje wszystko, co zostało wpisane i wyświetli nowy znak zachęty. Bez \c wyświetliłby się komunikat o błędzie. Jeśli jednak został otwarty napis lub komentarz, należy zamknąć go przed użyciem \c ; inaczej MySQL pomyśli, że \c jest tylko częścią ciągu.



Step 3

Do najpopularniejszych poleceń używanych w konsoli mysql należą:

- ALTER - Dokonuje zmian w tabeli lub bazie danych
- BACKUP - Tworzy zapasową kopię bazy danych
- \c - Anuluj wprowadzenie polecenia
- CREATE - Tworzy bazę danych
- DELETE - Usuwa wiersz z tabeli
- DESCRIBE - Opisuje kolumnę tabeli
- DROP - Usuń bazę danych lub tabelę
- EXIT (CTRL-C) - Wyjście
- GRANT - Zmienia uprawnienia użytkownika
- HELP (\h , \?) - Wyświetla pomoc
- INSERT - Wstawia dane do tabeli
- LOCK - Blokada tabeli
- QUIT (\q) - Ma takie samo działanie jak EXIT
- RENAME - Zmień nazwę tabeli
- SHOW - Wyświetla szczegóły dotyczące obiektu
- SOURCE - Wykonuje plik z poleceniami
- STATUS (\s) - Wyświetla aktualny stan
- TRUNCATE - Opróżnij tabelę
- UNLOCK - Odblokuj tabelę
- UPDATE - Zaktualizuj istniejące rekordy
- USE - Użyj bazy danych

W poleceniach i słowach kluczowych SQL nie jest rozróżniana wielkość liter. `CREATE`, `create`, oraz `CrEaTe` oznacza to samo. Jednak ze względu na przejrzystość możemy preferować używanie wielkich liter. W nazwach tabel jest rozróżniana wielkość liter w systemie Linux i macOS, ale w systemie Windows nie jest rozróżniana wielkość liter. Dlatego ze względu na przenośność należy zawsze wybierać etui i się go trzymać. Zalecanym stylem jest używanie małych liter w nazwach tabel.

Aby utworzyć nową bazę danych o nazwie `publications` należy w konsoli `mysql` skorzystać z następującego polecenia:

```
CREATE DATABASE publications;
```

Powodzenie powyższego polecenie zwróci komunikat, który jeszcze nie znaczy wiele - `Query OK, 1 row affected (0.02 sec)`. Po utworzeniu bazy danych jeżeli chcemy z nią pracować, należy wydać następujące polecenie:

```
USE publications;
```



Teraz należy przyjrzeć się, jak tworzyć użytkowników, ponieważ prawdopodobnie nie będziemy chcieli przyznawać skryptom PHP dostępu `root` do MySQL. Aby utworzyć użytkownika, należy skorzystać z polecenia `CREATE USER`, która przyjmuje następującą postać:

```
CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';
GRANT PRIVILEGES ON database.object TO 'username'@'hostname'
```

Wszystko to powinno wyglądać całkiem prosto, z możliwym wyjątkiem części `database.object`, która odnosi się do samej bazy danych i zawartych w niej obiektów. W przypadku użycia polecenia GRANT można skorzystać z następujących parametrów:

Argument	Znaczenie
<code>*.*</code>	Wszystkie bazy danych oraz wszystkie obiekty
<code>database.*</code>	Tylko baza danych o nazwie <code>database</code> i wszystkie zawarte w niej obiekty
<code>database.object</code>	Tylko baza danych o nazwie <code>database</code> i obiekt o nazwie <code>object</code>

Stwórzmy więc użytkownika, który będzie miał dostęp tylko do nowej bazy danych publikacji i wszystkich jej obiektów, wprowadzając następujące polecenia:

```
CREATE USER 'jim'@'localhost' IDENTIFIED BY 'password';
GRANT ALL ON publications.* TO 'jim'@'localhost';
```

```
mysql> █
```

█

Umożliwia to użytkownikowi `jim@localhost` pełny dostęp do bazy danych publikacji przy użyciu hasła `silneHaslo123@`. Możemy teraz zalogować się na użytkownika jim korzystając z polecenia:

```
mysql -u jim -p
```

```
mateusz@mateusz-HP-ProBook-650-G4:~$ █
```

█

Aby utworzyć tabelę w bazie publications skorzystamy z prostego zapytania SQL:

```
CREATE TABLE classics (
  author VARCHAR(128),
  title VARCHAR(128),
  type VARCHAR(16),
  year CHAR(4)) ENGINE InnoDB;
```

MySQL powinien następnie wydać odpowiedź `Query OK,` wraz z informacją o tym, ile czasu zajęło wykonanie polecenia. Jeśli zamiast tego pojawi się komunikat o błędzie, należy dokładnie sprawdzić składnię. Liczy się każdy nawias i przecinek, a błędy wpisywania są łatwe do popełnienia.

```

Server version: 8.0.23-0ubuntu0.20.10.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| publications   |
+-----+
2 rows in set (0.00 sec)

mysql> CREATE TABLE classics (
    -> author VARCHAR(128),
    -> title VARCHAR(128),
    -> type VARCHAR(16),
    -> year CHAR(4)) ENGINE InnoDB; ■

```

Ostatnie dwa słowa powyższego polecenia wymagają małego wyjaśnienia. MySQL może wewnętrznie przetwarzać zapytania na wiele różnych sposobów, a te różne sposoby są obsługiwane przez różne silniki. Począwszy od wersji 5.6 InnoDB jest domyślnym mechanizmem przechowywania danych dla MySQL i jest on tutaj używany, ponieważ obsługuje wyszukiwania FULLTEXT. Dopóki mamy stosunkowo aktualną wersję MySQL, możemy pominąć sekcję ENGINE InnoDB polecenia podczas tworzenia tabeli. Jeśli używamy wersji MySQL wcześniejszej niż 5.6, silnik InnoDB nie będzie obsługiwał indeksów FULLTEXT, więc będzie trzeba zmienić InnoDB w poleceniu na MyISAM, aby wskazać, że chcemy używać tego silnika. InnoDB jest ogólnie bardziej wydajnym rozwiązaniem i jest zalecaną opcją.

Aby sprawdzić, czy nowa tabela została utworzona, należy wpisać:

```
DESCRIBE classics;
```

Powinna pojawić się następująca odpowiedź:

```

+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| author | varchar(128) | YES |   | NULL    |       |
| title  | varchar(128)  | YES |   | NULL    |       |
| type   | varchar(16)   | YES |   | NULL    |       |
| year   | char(4)      | YES |   | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

Polecenie DESCRIBE jest nieocenioną pomocą przy debugowaniu, gdy musimy upewnić się, że poprawnie zostały utworzone tabelę MySQL. Możemy go również użyć do przypomnienia sobie o nazwach pól lub kolumn tabeli i typach danych w każdej z nich. Polecenie to wyświetla tabelę, korzystając z następujących nagłówków:

- **Field** - Nazwa każdego pola lub kolumny w tabeli;
- **Type** - Typ danych przechowywanych w polu;
- **Null** - Czy pole może zawierać wartość **NULL**;
- **Key** - Typ klucza, jeśli został zastosowany;
- **Default** - Wartość domyślna, która zostanie przypisana do pola, jeśli żadna wartość nie zostanie określona podczas tworzenia;
- **Extra** - Dodatkowe informacje, na przykład czy pole jest ustawione na automatyczne zwiększanie wartości.

Step 4

W poprzednim przykładzie można było zauważyć, że trzem polom tabeli nadano typ danych **VARCHAR**, a jednemu nadano typ **CHAR**. Termin **VARCHAR** oznacza ciąg znaków o zmiennej długości, a polecenie przyjmuje wartość liczbową, która mówi MySQL o maksymalnej dozwolonej długości ciągu przechowywanego w tym polu. Zarówno **CHAR**, jak i **VARCHAR** akceptują ciągi tekstowe i nakładają ograniczenie na rozmiar pola. Różnica polega na tym, że każdy łańcuch w polu **CHAR** ma określony rozmiar. Jeśli umieścisz mniejszy tekst, zostanie on wypełniony spacjami. Pole **VARCHAR** nie wypełnia tekstu; pozwala zmieniać rozmiar pola, aby dopasować

go do wstawianego tekstu. Ale `VARCHAR` wymaga niewielkiej ilości narzutów, aby śledzić rozmiar każdej wartości. Tak więc `CHAR` jest nieco bardziej wydajny, jeśli rozmiary są podobne we wszystkich rekordach, podczas gdy `VARCHAR` jest bardziej wydajny, jeśli rozmiary mogą się znacznie różnić i stać się duże. Ponadto narzut powoduje, że dostęp do danych `VARCHAR` jest nieco wolniejszy niż do danych `CHAR`. Kolejną cechą kolumn znakowych i tekstowych, ważną dla dzisiejszego globalnego zasięgu w sieci, są zestawy znaków. Przypisuję one określone wartości binarne do określonych znaków. Zestaw znaków, którego używasz w języku angielskim, jest oczywiście inny niż w przypadku języka polskiego. Podczas tworzenia zestawu znaków można przypisać znak lub kolumnę tekstową. `VARCHAR` jest przydatny w naszym przykładzie, ponieważ może pomieścić nazwiska autorów i tytuły o różnej długości, jednocześnie pomagając MySQL w planowaniu rozmiaru bazy danych i łatwiejszym wykonywaniu wyszukiwań i wyszukiwań. Pamiętaj tylko, że jeśli kiedykolwiek spróbujesz przypisać wartość ciągu dłuższą niż dozwolona długość, zostanie ona obcięta do maksymalnej długości zadeklarowanej w definicji tabeli. Pole roku ma jednak przewidywalne wartości, dlatego zamiast `VARCHAR` używamy bardziej wydajnego typu danych `CHAR(4)`. Parametr 4 pozwala na 4 bajty danych, obsługując wszystkie lata od -999 do 9999; bajt składa się z 8 bitów i może mieć wartości od 00000000 do 11111111, które są dziesiętne od 0 do 255. Możemy oczywiście po prostu zapisać dwucyfrowe wartości dla roku, ale jeśli dane będą nadal potrzebne w następnym stuleciu lub w inny sposób mogą się zawijać, najpierw będą musiały zostać oczyszczone - pomyśl o „Milenijnym błędzie”, który spowodował, że daty rozpoczynające się 1 stycznia 2000 r. były traktowane jako 1900 w wielu największych instalacjach komputerowych na świecie. Również nie został użyty typ `YEAR`, ponieważ obsługuje tylko lata od 0000 i 1901 do 2155. Dzieje się tak, ponieważ MySQL przechowuje rok w jednym bajcie ze względu na wydajność, ale oznacza to, że dostępnych jest tylko 256 lat, a lata publikacji tytułów w tabeli klasyków są na długo przed 1901 rokiem.

Poniżej znajduje się tabela opisująca listę typów danych `CHAR`. Oba typy oferują parametr, który ustawia maksymalną (lub dokładną) długość ciągu dozwoloną w polu. Jak pokazuje tabela, każdy typ ma wbudowaną maksymalną liczbę bajtów, które może zajmować.

Typ	Ilość bajtów	Przykład
<code>CHAR(n)</code>	Dokładnie n (<= 255)	<code>CHAR(5) "Hello"</code> używa 5 bajtów; <code>CHAR(57) "Goodbye"</code> używa 57 bajtów
<code>VARCHAR(n)</code>	Co najwyżej n (<= 65535)	<code>VARCHAR(7) "Hello"</code> używa 5 bajtów; <code>VARCHAR(100) "Goodbye"</code> używa 7 bajtów

Typ `BINARY` przechowuje ciągi bajtów, które nie mają skojarzonego zestawu znaków. Na przykład możemy użyć typu danych `BINARY` do przechowywania obrazu GIF.

Typ	Ilość bajtów	Przykład
<code>BINARY(n)</code>	Dokładnie n (<=255)	Jako <code>CHAR</code> ale w postaci binarnej
<code>VARBINARY(n)</code>	Co najwyżej n (<=65535)	Jako <code>VARCHAR</code> ale w postaci binarnej

Dane tekstowe mogą być również przechowywane w jednym z zestawów pól `TEXT`. Różnice między tymi polami a polami `VARCHAR` są niewielkie: Przed wersją 5.0.3 MySQL usuwał początkowe i końcowe spacje z pól `VARCHAR`. Pola `TEXT` nie mogą mieć wartości domyślnych. MySQL indeksuje tylko pierwsze `n` znaków kolumny `TEXT` (`n` jest określone podczas tworzenia indeksu). Oznacza to, że `VARCHAR` jest lepszym i szybszym typem danych do wykorzystania, jeśli chcesz przeszukać całą zawartość pola. Jeśli nigdy nie będziemy przeszukiwać więcej niż określona liczbę początkowych znaków w polu, prawdopodobnie powinniśmy użyć typu danych `TEXT`.

Typ	Ilość bajtów	Przykład
<code>TINYTEXT(n)</code>	Co najwyżej n (<=255)	Traktowane jako ciąg z zestawem znaków
<code>TEXT(n)</code>	Co najwyżej n (<=65535)	Traktowane jako ciąg z zestawem znaków
<code>MEDIUMTEXT(n)</code>	Co najwyżej n (<=1.67e+7)	Traktowane jako ciąg z zestawem znaków
<code>LONGTEXT(n)</code>	Co najwyżej n (<=4.29e+9)	Traktowane jako ciąg z zestawem znaków

Termin `BL0B` oznacza **Binär Large OBject** i dlatego, jak można by pomyśleć, typ danych `BL0B` jest najbardziej przydatny w przypadku danych binarnych o rozmiarze przekraczającym 65 536 bajtów. Główną różnicą między typami danych `BL0B` i `BINARY` jest to, że obiekty `BL0B` nie mogą mieć wartości domyślnych.

Typ	Ilość bajtów	Przykład
<code>TINYBL0B(n)</code>	Co najwyżej n (<=255)	Traktowane jako dane binarne - bez zestawu znaków

Typ	Ilość bajtów	Przykład
BL0B(n)	Co najwyżej n (<=65535)	Traktowane jako dane binarne - bez zestawu znaków
MEDIUMBLOB(n)	Co najwyżej n (<=1.67e+7)	Traktowane jako dane binarne - bez zestawu znaków
LONGBLOB(n)	Co najwyżej n (<=4.29e+9)	Traktowane jako dane binarne - bez zestawu znaków

MySQL obsługuje różne numeryczne typy danych, od pojedynczych bajtów po liczby zmiennoprzecinkowe o podwójnej precyzyji. Chociaż najwięcej pamięci, jaką może wykorzystać pole numeryczne, wynosi 8 bajtów, zaleca się wybranie najmniejszego typu danych, który będzie odpowiednio obsługiwał największą oczekiwana wartość. Poniżej znajduje się lista numerycznych typów danych obsługiwanych przez MySQL oraz zakresy wartości, które mogą zawierać. Jeśli nie jesteś zaznajomiony z terminami, liczba ze znakiem to liczba z możliwym zakresem od wartości ujemnej, przez 0, do dodatniej; a liczba bez znaku ma wartość od 0 do dodatniej. Oba mogą zawierać tę samą liczbę wartości.

Typ	Ilość bajtów	Min wartość signed	Min wartość unsigned	Max wartość signed	Max wartość unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8.38e+6	0	8.38e+6	1.67e+7
INT/INTEGER	4	-2.15e+9	0	2.15e+9	4.29e+9
BIGINT	8	-9.22e+18	0	9.22e+18	1.84e+19
FLOAT	4	-3.40e+38	n/a	3.40e+38	n/a
DOUBLE/REAL	8	-1.80e+308	n/a	1.80e+308	n/a

Aby określić, czy typ danych jest bez znaku, należy użyć kwalifikatora `UNSIGNED`.

```
CREATE TABLE tablename (fieldname INT UNSIGNED);
```

Tworząc pole liczbowe, możemy również przekazać opcjonalną liczbę jako parametr, na przykład:

```
CREATE TABLE tablename (fieldname INT(4));
```

Należy jednak pamiętać, że w przeciwieństwie do typów danych `BINARY` i `CHAR`, parametr ten nie wskazuje liczby bajtów pamięci do wykorzystania. Może się to wydawać sprzeczne z intuicją, ale to, co faktycznie reprezentuje liczba, to szerokość wyświetlanego danych w polu podczas ich pobierania. Jest powszechnie używany z kwalifikatorem `ZEROFILL`, na przykład:

```
CREATE TABLE tablename (fieldname INT(4) ZEROFILL);
```

Powoduje to, że wszystkie liczby o szerokości mniejszej niż cztery znaki są uzupełniane jednym lub większą liczbą zer, co wystarcza, aby szerokość wyświetlanego pola była dłuża na cztery znaki. Gdy pole ma już określoną szerokość lub większą, dopełnienie nie jest stosowane.

Główne pozostałe typy danych obsługiwane przez MySQL odnoszą się do daty i czasu i można je zobaczyć w poniższej tabeli:

Typ danych	Format
DATETIME	0000-00-00 00:00:00
DATE	0000-00-00
TIMESTAMP	0000-00-00 00:00:00
TIME	00:00:00
YEAR	0000 (Tylko 0000 oraz 1901-2155)

Typy danych `DATETIME` i `TIMESTAMP` są wyświetlane w ten sam sposób. Główna różnica polega na tym, że `TIMESTAMP` ma bardzo wąski zakres (od lat 1970 do 2037), podczas gdy `DATETIME` będzie zawierać prawie każdą datę, którą prawdopodobnie określmy, chyba że interesujemy się historią starożytną lub science fiction. `TIMESTAMP` jest jednak przydatny, ponieważ możesz pozwolić MySQL ustawić wartość za nas. Jeśli nie określmy wartości podczas dodawania wiersza, bieżący czas zostanie wstawiony automatycznie.

Step 5

Czasami trzeba się upewnić, że każdy wiersz w bazie danych jest niepowtarzalny. Możemy to zrobić w swoim programie, dokładnie sprawdzając wprowadzone dane i upewniając się, że istnieje co najmniej jedna wartość, która różni się w dowolnych dwóch wierszach, ale takie podejście jest podatne na błędy i działa tylko w określonych okolicznościach. Na przykład w tabeli `classics` autor może pojawiać się wielokrotnie. Podobnie rok publikacji będzie często powielany i tak dalej. Trudno byłoby zagwarantować, że nie będzie zduplikowanych wierszy. Ogólnym rozwiązaniem jest użycie dodatkowej kolumny tylko do tego celu z ustawionym atrybutem `AUTO_INCREMENT`. Jak sama nazwa wskazuje, kolumna o danym typie danych ustawi wartość swojej zawartości na wartość wpisu kolumny we wcześniej wstawionym wierszu, plus 1.

```
ALTER TABLE classics ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY;
```

Polecenie `ALTER` działa na istniejącej tabeli i może dodawać, zmieniać lub usuwać kolumny. Powyższy przykład dodaje kolumnę o nazwie `id` z następującymi cechami:

- `INT UNSIGNED` - Sprawia, że kolumna przyjmuje liczbę całkowitą wystarczająco dużą, abyśmy mogli przechowywać w tabeli ponad 4 miliardy rekordów.
- `NOT NULL` - Zapewnia, że kolumna ma wartość. Wielu programistów używa wartości `NULL` w polu, aby wskazać, że nie ma ono żadnej wartości. Dopuszciliby to jednak duplikaty, co naruszyłoby cały powód istnienia tej kolumny, dlatego nie zezwalamy na wartości `NULL`.
- `AUTO_INCREMENT` - Powoduje, że MySQL ustawia unikalną wartość dla tej kolumny w każdym wierszu, jak opisano wcześniej. Tak naprawdę nie mamy kontroli nad wartością, jaką ta kolumna przyjmie w każdym wierszu, ale nie obchodzi nas to: zależy nam tylko na tym, aby zagwarantować nam niepowtarzaną wartość.
- `KEY` - Kolumna z automatycznym zwiększaniem jest przydatna jako klucz, ponieważ będziemy wyszukiwać wiersze na podstawie tej kolumny.

Każdy wpis w identyfikatorze kolumny będzie miał teraz unikalny numer, przy czym pierwszy zaczyna się od 1, a pozostałe liczą się w góry. Za każdym razem, gdy zostanie wstawiony nowy wiersz, kolumna `id` automatycznie otrzyma kolejną liczbę w sekwencji. Zamiast stosować kolumnę z mocą wsteczną, można było ją uwzględnić, wydając polecenie `CREATE` w nieco innym formacie.

```
CREATE TABLE classics (
    author VARCHAR(128),
    title VARCHAR(128),
    type VARCHAR(16),
    year CHAR(4),
    id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY) ENGINE InnoDB;
```

Po wywołaniu `DESCRIBE classics` powinniśmy otrzymać następujący rezultat:

Field	Type	Null	Key	Default	Extra
author	varchar(128)	YES		NULL	
title	varchar(128)	YES		NULL	
type	varchar(16)	YES		NULL	
year	char(4)	YES		NULL	
id	int unsigned	NO	PRI	NULL	auto_increment

Gdyby się okazało, że kolumna `id` nie jest już potrzebna, to możemy ją usunąć kolumnę za pomocą polecenia:

```
ALTER TABLE classics DROP id;
```

Aby dodać dane do tabeli, należy użyć polecenia `INSERT INTO`. Na przykład dla tabeli `classics` wstawianie danych może wyglądać następująco:

```

INSERT INTO classics(author, title, type, year)
VALUES('Mark Twain','The Adventures of Tom Sawyer','Fiction','1876');
INSERT INTO classics(author, title, type, year)
VALUES('Jane Austen','Pride and Prejudice','Fiction','1811');
INSERT INTO classics(author, title, type, year)
VALUES('Charles Darwin','The Origin of Species','Non-Fiction','1856');
INSERT INTO classics(author, title, type, year)
VALUES('Charles Dickens','The Old Curiosity Shop','Fiction','1841');
INSERT INTO classics(author, title, type, year)
VALUES('William Shakespeare','Romeo and Juliet','Play','1594');

```

Po każdej drugiej linii powinien pojawić się komunikat `Query OK`. Po wprowadzeniu wszystkich wierszy wyświetlenie ich można wywołać polecением:

```
SELECT * FROM classics;
```

Rezultat powinien wyglądać następująco:

author	title	type	year
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
Charles Darwin	The Origin of Species	Non-Fiction	1856
Charles Dickens	The Old Curiosity Shop	Fiction	1841
William Shakespeare	Romeo and Juliet	Play	1594

5 rows in set (0.00 sec)



Może się zdarzyć, że zwrócone wyniki w poleceniu `SELECT` są w innej kolejności, ponieważ kolejność nie jest w tym momencie określona. Pierwsza część, `INSERT INTO classics`, mówi MySQL, gdzie wstawić następujące dane. Następnie w nawiasach wymienione są cztery nazwy kolumn - `author`, `title`, `type` i `year` - wszystkie oddzielone przecinkami. To mówi MySQL, że są to pola, do których mają zostać wstawione dane. Drugi wiersz każdego polecenia `INSERT` zawiera słowo kluczowe `VALUES`, po którym następują cztery ciągi w nawiasach, oddzielone przecinkami. Zapewnia to MySQL cztery wartości, które mają zostać wstawione do czterech wcześniej określonych kolumn. Każda pozycja danych zostanie wstawiona do odpowiedniej kolumny w korespondencji jeden do jednego. Jeśli przypadkowo umieścimy kolumny w innej kolejności niż dane, dane zostaną umieszczone w niewłaściwych kolumnach. Ponadto liczba kolumn musi odpowiadać liczbie elementów danych.

Zmiana nazwy tabeli, podobnie jak każda inna zmiana struktury lub metainformacji tabeli, odbywa się za pomocą polecenia `ALTER`. Na przykład, aby zmienić nazwę `classics` na `pre1900`, należy użyć następującego polecenia:

```
ALTER TABLE classics RENAME pre1900;
```

Zmiana typu danych w kolumnie również wykorzystuje polecenie `ALTER`, tym razem w połączeniu ze słowem kluczowym `MODIFY`. Aby zmienić typ danych roku kolumny z `CHAR(4)` na `SMALLINT` (który wymaga tylko 2 bajtów pamięci, a więc oszczędza miejsce na dysku), należy użyć polecenia:

```
ALTER TABLE classics MODIFY year SMALLINT;
```

Gdy to zrobisz, jeśli konwersja typu danych ma sens do MySQL, automatycznie zmieni dane, zachowując znaczenie. W takim przypadku zmieni każdy ciąg na porównywalną liczbę całkowitą, o ile ciąg jest rozpoznawalny jako odnoszący się do liczby całkowitej. Założmy, że utworzyliśmy tabelę i zapełniliśmy ją dużą ilością danych, tylko po to, aby odkryć, że potrzebujemy dodatkowej kolumny. Aby dodać nową kolumnę, określającą ilość stron należy użyć polecenia:

```
ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
```

Spowoduje to dodanie nowej kolumny ze stronami z nazwami korzystającymi z typu danych `UNSIGNED` `SMALLINT`, wystarczającym do przechowywania wartości do 65 535. Wywołując polecenie `DESCRIBE classics` otrzymamy następujący rezultat:

```
+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| author | varchar(128) | YES  |     | NULL    |       |
| title  | varchar(128)  | YES  |     | NULL    |       |
| type   | varchar(16)   | YES  |     | NULL    |       |
| year   | char(4)        | YES  |     | NULL    |       |
| pages  | smallint unsigned | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Patrząc ponownie powyższą tabelę, łatwo stwierdzić, że kolumna o nazwie `type` jest myląca, ponieważ jest to nazwa używana przez MySQL do identyfikowania typów danych. Zmienimy jego nazwę na `category`:

```
ALTER TABLE classics CHANGE type category VARCHAR(16);
```

```
+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| author | varchar(128) | YES  |     | NULL    |       |
| title  | varchar(128)  | YES  |     | NULL    |       |
| category | varchar(16) | YES  |     | NULL    |       |
| year   | char(4)        | YES  |     | NULL    |       |
| pages  | smallint unsigned | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Zwróć uwagę na dodanie `VARCHAR(16)` na końcu tego polecenia. Dzieje się tak, ponieważ słowo kluczowe `CHANGE` wymaga określenia typu danych, nawet jeśli nie zamierzamy go zmieniać, a `VARCHAR(16)` był typem danych określonym podczas początkowego tworzenia tej kolumny jako typ. Właściwie po zastanowieniu możesz zdecydować, że strony z kolumnami z liczbą stron nie są w rzeczywistości przydatne w tej konkretnej bazie danych, więc oto jak usunąć tę kolumnę za pomocą słowa kluczowego `DROP`:

```
ALTER TABLE classics DROP pages;
```

```
+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| author | varchar(128) | YES  |     | NULL    |       |
| title  | varchar(128)  | YES  |     | NULL    |       |
| category | varchar(16) | YES  |     | NULL    |       |
| year   | char(4)        | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Należy pamiętać, że polecenie `DROP` jest nieodwracalne. Powinno zawsze używać się go ostrożnie, ponieważ możemy nieumyślnie usunąć całe tabele (a nawet bazy danych)!

Usunięcie tabeli jest rzeczywiście bardzo łatwe. Dla przykładu utwórzmy bardzo prostą tabelę o nazwie trash:

```
CREATE TABLE trash(id INT);
```

Wynik polecenia `SHOW TABLES` wygląda następująco:

```
+-----+  
| Tables_in_publications |  
+-----+  
| classics                |  
| trash                   |  
+-----+  
2 rows in set (0.01 sec)
```

Aby usunąć tabelę, korzystamy tak samo jak w przypadku kolumny z polecenia `DROP`:

```
DROP TABLE trash;
```

```
+-----+  
| Tables_in_publications |  
+-----+  
| classics                |  
+-----+  
1 row in set (0.01 sec)
```

Step 6

Do tej pory stworzyliśmy tabelę `classics`. Wszystkie publikacje w tabeli można przeszukiwać, ale nie ma jednego unikalnego klucza dla każdej publikacji, aby umożliwić natychmiastowy dostęp do wiersza. Znaczenie posiadania klucza z unikalną wartością dla każdego wiersza pojawi się, gdy zaczniemy łączyć dane z różnych tabel. Atrybut `AUTO_INCREMENT` przedstawiał ideę klucza podstawowego podczas tworzenia identyfikatora kolumny z automatycznym zwiększaniem wartości, który mógłby zostać użyty jako klucz podstawowy dla tej tabeli. Wykorzystamy to dodając do każdej publikacji unikalny numer ISBN. Numery ISBN mają 13 znaków, więc zaczniemy od dodania odpowiedniej kolumny do naszej tabeli:

```
ALTER TABLE classics ADD isbn CHAR(13) PRIMARY KEY;
```

Powyższe polecenie zakończy się niepowodzeniem. Otrzymamy błąd `Duplicate entry '' for key 'classics.PRIMARY'`. Przyczyną jest to, że tabela jest już wypełniona niektórymi danymi i to polecenie próbuje dodać kolumnę o wartości NULL do każdego wiersza, co jest niedozwolone, ponieważ wszystkie wartości muszą być unikalne w każdej kolumnie mającej indeks klucza podstawowego. Jeśli jednak w tabeli nie ma żadnych danych, to polecenie działałoby dobrze. W tej obecnej sytuacji musimy stworzyć nową kolumnę bez indeksu, wypełnić ją danymi, a następnie dodać indeks klucza głównego. Na szczęście każdy rok jest unikalny w bieżącym zestawie danych, więc możemy użyć kolumny roku do zidentyfikowania każdego wiersza do aktualizacji.

```
ALTER TABLE classics ADD isbn CHAR(13);  
UPDATE classics SET isbn='9781598184891' WHERE year='1876';  
UPDATE classics SET isbn='9780582506206' WHERE year='1811';  
UPDATE classics SET isbn='9780517123201' WHERE year='1856';  
UPDATE classics SET isbn='9780099533474' WHERE year='1841';  
UPDATE classics SET isbn='9780192814968' WHERE year='1594';  
ALTER TABLE classics ADD PRIMARY KEY(isbn);  
DESCRIBE classics;
```

```
+-----+-----+-----+-----+-----+  
| Field    | Type      | Null | Key  | Default | Extra |  
+-----+-----+-----+-----+-----+  
| author   | varchar(128)| YES  |      | NULL    |       |  
| title    | varchar(128)| YES  |      | NULL    |       |  
| category | varchar(16) | YES  |      | NULL    |       |  
| year     | char(4)    | YES  |      | NULL    |       |  
| isbn     | char(13)   | NO   | PRI  | NULL    |       |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

Aby utworzyć klucz podstawowy podczas tworzenia klasycznych tabel, można użyć następującego polecenia:

```

CREATE TABLE classics (
    author VARCHAR(128),
    title VARCHAR(128),
    category VARCHAR(16),
    year SMALLINT,
    isbn CHAR(13),
    INDEX(author(20)),
    INDEX(title(20)),
    INDEX(category(4)),
    INDEX(year),
    PRIMARY KEY (isbn)) ENGINE InnoDB;

```

Step 7

Sposobem na szybkie wyszukiwanie w bazie danych jest **dodanie indeksu** podczas tworzenia tabeli lub później w dowolnym momencie istnieją różne typy indeksów, takie jak zwykły **INDEX**, wspomniany wcześniej **PRIMARY KEY** lub indeks **FULLTEXT**. Musimy także zadecydować, które kolumny wymagają indeksu, co wymaga przewidywania, czy będziemy przeszukiwać dowolne dane w każdej kolumnie. Indeksy również mogą być bardziej skomplikowane, ponieważ można łączyć wiele kolumn w jednym indeksie. Nawet jeśli już zadecydujemy, nadal możemy zmniejszyć rozmiar indeksu, ograniczając liczbę indeksowanych kolumn. Jeśli wyobrażymy sobie wyszukiwania, które można przeprowadzić w tabeli **classics**, stanie się jasne, że wszystkie kolumny mogą wymagać przeszukania. Aby dodać indeks do kolumny należy skorzystać z polecenia **ALTER TABLE** oraz **ADD INDEX**:

```

ALTER TABLE classics ADD INDEX(author(20));
ALTER TABLE classics ADD INDEX(title(20));
ALTER TABLE classics ADD INDEX(category(4));
ALTER TABLE classics ADD INDEX(year);
DESCRIBE classics;

```

Wynik powyższego polecenia będzie wyglądał następująco (należy zwrócić uwagę na kolumnę **Key**):

Field	Type	Null	Key	Default	Extra
author	varchar(128)	YES	MUL	NULL	
title	varchar(128)	YES	MUL	NULL	
category	varchar(16)	YES	MUL	NULL	
year	smallint	YES	MUL	NULL	
isbn	char(13)	NO	PRI	NULL	

5 rows in set (0,00 sec)

Pierwsze dwie komendy tworzą indeksy w kolumnach **author** i **title**, ograniczając każdy indeks do pierwszych 20 znaków. Na przykład, gdy MySQL indeksuje tytuł **The Adventures of Tom Sawyer**, to w rzeczywistości będzie przechowywać w indeksie tylko pierwsze 20 znaków, czyli **The Adventures of To**. Ma to na celu zminimalizowanie rozmiaru indeksu i zoptymalizowanie szybkości dostępu do bazy danych. Jeśli MySQL znajdzie dwa indeksy o tej samej zawartości, będzie musiał tracić czas na przechodzenie do samej tabeli i sprawdzanie zindeksowanej kolumny, aby dowiedzieć się, które wiersze są naprawdę dopasowane. W kolumnie **category** obecnie tylko pierwszy znak jest wymagany, aby zidentyfikować ciąg jako unikalny (**F** jak fikcja, **N** jak literatura faktu i **P** jak gra), ale wybrano indeks czterech znaków, aby umożliwić przyszłym kategoriom indeksowanie, które mogą być identyfikowane przez pierwsze trzy znaki. W przypadku limitu indeksu kolumny **year** nie ma limitu, ponieważ ma on jasno określona długość czterech znaków. Klucz **MUL** dla każdej kolumny oznacza, że w tej kolumnie może wystąpić wiele wystąpień wartości, co jest dokładnie tym, czego chcemy, ponieważ autorzy mogą pojawiać się wiele razy, ten sam tytuł książki może być używany przez wielu autorów i tak dalej.

Alternatywą dla użycia **ALTER TABLE** w celu dodania indeksu jest użycie polecenia **CREATE INDEX**. Są równoważne, z tą różnicą, że **CREATE INDEX** nie może być użyte do utworzenia **PRIMARY KEY**. Format tego polecenia może wyglądać następująco:

```

ALTER TABLE classics ADD INDEX(author(20));
CREATE INDEX author ON classics (author(20));

```

Nie musimy czekać do momentu utworzenia tabeli, aby dodać indeksy. W rzeczywistości może to być czasochłonne, ponieważ dodanie indeksu do dużej tabeli może zająć bardzo dużo czasu. Polecenie poniżej, które tworzy tabelę **classics** z indeksami już na miejscu:

```
CREATE TABLE classics (
    author VARCHAR(128),
    title VARCHAR(128),
    category VARCHAR(16),
    year SMALLINT,
    INDEX(author(20)),
    INDEX(title(20)),
    INDEX(category(4)),
    INDEX(year)) ENGINE InnoDB;
```

W przeciwieństwie do zwykłego indeksu, `FULLTEXT` MySQL umożliwia superszybkie przeszukiwanie całych kolumn tekstu. Przechowuje każde słowo w każdym ciągu danych w specjalnym indeksie, który można przeszukiwać przy użyciu „języka naturalnego”, podobnie jak przy użyciu wyszukiwarki. Oto kilka rzeczy, które charakteryzują indeks `FULLTEXT`:

- Od MySQL 5.6 tabele `InnoDB` mogą używać indeksów `FULLTEXT`, ale wcześniej indeksy `FULLTEXT` mogły być używane tylko z tabelami `MyISAM`. Jeśli chcemy przekonwertować tabelę na `MyISAM`, należy użyć polecenia MySQL `ALTER TABLE nazwa_tabeli ENGINE = MyISAM ;`.
- Indeksy `FULLTEXT` można tworzyć tylko dla kolumn `CHAR`, `VARCHAR` i `TEXT`.
- Definicję indeksu `FULLTEXT` można podać w instrukcji `CREATE TABLE` podczas tworzenia tabeli lub dodać później za pomocą `ALTER TABLE` (lub `CREATE INDEX`).
- W przypadku dużych zestawów danych o wiele szybciej można załadować dane do tabeli, która nie ma indeksu `FULLTEXT`, a następnie utworzyć indeks, niż załadować dane do tabeli, która ma istniejący indeks `FULLTEXT`.

Aby utworzyć indeks `FULLTEXT`, w tabeli `classics` dla pary `author` oraz `title` należy użyć polecenia:

```
ALTER TABLE classics ADD FULLTEXT(author,title);
```

Możemy teraz przeprowadzić wyszukiwanie `FULLTEXT` w tej parze kolumn. Ta funkcja mogłaby naprawdę zadziałać, gdybyśmy mogli teraz dodać cały tekst tych publikacji do bazy danych (szczególnie, że nie są one objęte ochroną praw autorskich) i byłyby w pełni przeszukiwalne.

Nie jest do końca prawdą, że MySQL przechowuje wszystkie słowa w indeksie `FULLTEXT`, ponieważ ma wbudowaną listę ponad 500 słów, które decyduje się ignorować, ponieważ są one tak powszechnne, że i tak nie są zbyt pomocne w wyszukiwaniu tzw. `stopwords`. Ta lista obejmuje między innymi słowa jak: `the`, `as`, `is`, `and`, `so on`. Lista pomaga MySQL działać znacznie szybciej podczas wyszukiwania `FULLTEXT` i utrzymuje małe rozmiary bazy danych. Jeśli dojdziemy do wniosku, że MySQL działa wolniej, podczas uzyskiwania dostępu do bazy danych, problem jest zwykle związany z indeksami. Albo nie ma indeksu, tam gdzie go potrzebujemy, albo indeksy nie są optymalnie zaprojektowane.

Step 8

Do tej pory utworzyliśmy bazę danych MySQL i tabelle, zapełniliśmy je danymi i dodaliśmy indeksy, aby przyspieszyć ich wyszukiwanie. Nadszedł czas, aby przyjrzeć się, w jaki sposób te wyszukiwania są wykonywane, oraz z różnymi dostępnymi poleceniami i kwalifikatorami.

Polecenie `SELECT` służy do wyodrębniania danych z tabeli. Podstawowa składnia to:

```
SELECT something FROM tablename;
```

```
SELECT author,title FROM classics;
```

`something` może być `*` (gwiazdką), co oznacza każdą kolumnę, lub możemy wybrać tylko niektóre kolumny, na przykład:

```
SELECT author,title FROM classics;
```

```
+-----+-----+
| author           | title
+-----+-----+
| Charles Dickens | The Old Curiosity Shop
| William Shakespeare | Romeo and Juliet
| Charles Darwin   | The Origin of Species
| Jane Austen     | Pride and Prejudice
| Mark Twain      | The Adventures of Tom Sawyer
+-----+
5 rows in set (0,00 sec)
```

Innym zamiennikiem parametru `something` jest `COUNT`, którego można używać na wiele sposobów. Poniżej wyświetlana jest liczba wierszy w tabeli, przekazując `*` jako parametr, co oznacza wszystkie wiersze. Jak można się spodziewać, zwrócony wynik to 5, ponieważ w tabeli jest pięć publikacji.

```
SELECT COUNT(*) FROM classics;
```

```
+-----+
| COUNT(*) |
+-----+
|      5 |
+-----+
1 row in set (0,00 sec)
```

Kwaliifikator `DISTINCT` (i jego synonim `DISTINCTROW`) umożliwia wyeliminowanie wielu wpisów, które zawierają te same dane. Na przykład założymy, że chcemy wyświetlić listę wszystkich autorów w tabeli. Jeśli wybierzemy tylko kolumnę autora z tabeli zawierającej wiele książek tego samego autora, zwykle zobaczymy długą listę z tymi samymi nazwiskami autorów w kółko. Ale dodając słowo kluczowe `DISTINCT`, możemy wyświetlić każdego autora tylko raz. Przetestujmy więc to, dodając kolejny wiersz, który powtarza jednego z naszych istniejących autorów:

```
INSERT INTO classics(author, title, category, year, isbn)
VALUES('Charles Dickens','Little Dorrit','Fiction','1857', '9780141439969');
```

Teraz, gdy `Charles Dickens` pojawia się w tabeli dwukrotnie, możemy porównać wyniki użycia funkcji `SELECT` z kwalifikatorem `DISTINCT` i bez niego:

```
SELECT author FROM classics;
```

```
+-----+
| author           |
+-----+
| Charles Dickens |
| Charles Dickens |
| William Shakespeare |
| Charles Darwin   |
| Jane Austen     |
| Mark Twain      |
+-----+
6 rows in set (0,00 sec)
```

```
SELECT DISTINCT author FROM classics;
```

```
+-----+
| author           |
+-----+
| Charles Dickens |
| William Shakespeare |
| Charles Darwin   |
| Jane Austen     |
| Mark Twain      |
+-----+
5 rows in set (0,00 sec)
```

Słowo kluczowe `WHERE` umożliwia zawężenie zapytań, zwracając tylko te, w których określone wyrażenie jest prawdziwe. Poniższy przykład zwraca tylko wiersze, w których kolumna dokładnie odpowiada ciągowi `Little Dorrit`, używając operatora równości `=`.

```
SELECT author,title FROM classics WHERE author="Mark Twain";
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Mark Twain | The Adventures of Tom Sawyer |
+-----+
1 row in set (0,00 sec)
```

Możemy również dopasować wzorce dla swoich wyszukiwań, używając kwalifikatora `LIKE`, który umożliwia wyszukiwanie na częściach ciągów. Tego kwalifikatora należy używać ze znakiem `%` przed lub po jakimś tekście. Umieszczony przed słowem kluczowym `%` oznacza cokolwiek wcześniej. Po słowie kluczowym oznacza wszystko po.

```
SELECT author,title FROM classics WHERE author LIKE "Charles%";
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Charles Darwin | The Origin of Species |
| Charles Dickens | The Old Curiosity Shop |
| Charles Dickens | Little Dorrit |
+-----+
3 rows in set (0,00 sec)
```

```
SELECT author,title FROM classics WHERE title LIKE "%and%";
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| William Shakespeare | Romeo and Juliet |
| Jane Austen       | Pride and Prejudice |
+-----+
```

Pierwsze polecenie wyświetla publikacje zarówno `Charles Darwin`, jak i `Charles Dickens`, ponieważ kwalifikator `LIKE` został ustawiony tak, aby zwracał wszystko, co pasuje do ciągu `Charles`, po którym następuje dowolny inny tekst. Następnie zwracane są zarówno `Pride and Prejudice` oraz `Romeo and Juliet`, ponieważ oba pasowały do ciągu i dowolnego miejsca w kolumnie. `%` Będzie również pasował, jeśli na pozycji, którą zajmuje, nie ma niczego; innymi słowy, może dopasować pusty串.

Kwalifikator `LIMIT` umożliwia wybranie liczby wierszy zwracanych w zapytaniu i miejsca w tabeli, w którym mają one zacząć zwracać. Przekazany pojedynczy parametr nakazuje MySQL rozpoczęcie od początku wyników i zwrócenie liczby wierszy podanej w tym parametrze. Jeśli przekazane mu będą dwa parametry, pierwszy wskazuje przesunięcie od początku wyników, w którym MySQL powinien rozpocząć wyświetlanie, a drugi wskazuje, ile ma zwracać. Możemy wyobrazić sobie, że pierwszy parametr mówi: „Pomiń tę liczbę wyników na początku”.

```
SELECT author,title FROM classics LIMIT 3;
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop |
| Charles Dickens | Little Dorrit |
| William Shakespeare | Romeo and Juliet |
+-----+
3 rows in set (0,00 sec)
```

```
SELECT author,title FROM classics LIMIT 1,2;
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Charles Dickens | Little Dorrit |
| William Shakespeare | Romeo and Juliet |
+-----+
2 rows in set (0,00 sec)
```

```
SELECT author,title FROM classics LIMIT 3,1;
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Charles Darwin | The Origin of Species |
+-----+-----+
1 row in set (0,00 sec)
```

Pierwsze polecenie zwraca pierwsze trzy wiersze z tabeli. Drugie zwraca dwa wiersze, zaczynając od pozycji 1 (ponijając pierwszy wiersz). Ostatnie polecenie zwraca pojedynczy wiersz, zaczynając od pozycji 3 (ponijając pierwsze trzy wiersze). Należy uważać na słowo kluczowe `LIMIT`, ponieważ przesunięcia zaczynają się od `0`, ale liczba wierszy do zwrócenia zaczyna się od `1`. Tak więc `LIMIT 1,3` oznacza zwracanie trzech wierszy, zaczynając od drugiego wiersza. Możemy spojrzeć na pierwszy argument jako określający liczbę wierszy do pominięcia, tak aby w języku angielskim instrukcja brzmiała „`Zwróć 3 wiersze, pomijając pierwszy 1.`”.

Step 9

Konstrukcja `MATCH ... AGAINST` może być używana w kolumnach, które otrzymały indeks `FULLTEXT`. W przeciwnieństwie do użycia `WHERE ... = ...` lub `WHERE ... LIKE ...`, `MATCH ... AGAINST` umożliwia wprowadzenie wielu słów w zapytaniu i porównuje je ze wszystkimi słowami w kolumnach `FULLTEXT`. W indeksach `FULLTEXT` wielkość liter nie jest rozróżniana, więc nie ma znaczenia, jaką wielkość liter jest używana w zapytaniach. Zakładając, że dodaliśmy wcześniej indeks `FULLTEXT` do kolumny `author` oraz `title`, wprowadźmy trzy zapytania poniżej:

```
SELECT author,title FROM classics
WHERE MATCH(author,title) AGAINST('and');
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| William Shakespeare | Romeo and Juliet |
| Jane Austen       | Pride and Prejudice |
+-----+-----+
2 rows in set (0,00 sec)
```

```
SELECT author,title FROM classics
WHERE MATCH(author,title) AGAINST('curiosity shop');
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop |
+-----+-----+
1 row in set (0,00 sec)
```

```
SELECT author,title FROM classics
WHERE MATCH(author,title) AGAINST('tom sawyer');
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Mark Twain | The Adventures of Tom Sawyer |
+-----+-----+
1 row in set (0,00 sec)
```

Pierwsza prosi o wszystkie wiersze, które zawierają słowo `and`. Jeśli używamy silnika pamięci `MyISAM`, to ponieważ MySQL zignoruje go, ponieważ jest to słowo stopowe w tym silniku, a zapytanie zawsze zwróci pusty zestaw - bez względu na to, co jest zapisane w kolumnie. W przeciwnym razie, jeśli używasz `InnoDB`, jest to dozwolone słowo. Drugie zapytanie prosi o zwrócenie wszystkich wierszy zawierających oba słowa `curiosity shop` w dowolnym miejscu, w dowolnej kolejności. Ostatnie zapytanie dotyczy tego samego rodzaju wyszukiwania słów `tom sawyer`.

Jeśli chcemy dać zapytania `MATCH ... AGAINST` jeszcze więcej mocy, należy użyć trybu boolowskiego. Spowoduje to zmianę efektu standardowego zapytania `FULLTEXT`, tak aby szukało dowolnej kombinacji wyszukiwanych słów, zamiast wymagać, aby wszystkie wyszukiwane słowa znajdowały się w tekście. Obecność pojedynczego słowa w kolumnie powoduje, że wyszukiwanie zwraca wiersz. Tryb boolowski umożliwia również poprzedzanie wyszukiwanych słów znakiem `+` lub `-`, aby wskazać, czy należy je uwzględnić, czy

wykluczyć. Jeśli normalny tryb boolowski mówi: „Każde z tych słów wystarczy”, znak plus oznacza: „To słowo musi być obecne; w przeciwnym razie nie zwracaj wiersza”. Znak minus oznacza: „To słowo nie może być obecne; jego obecność dyskwalifikuje wiersz przed zwróceniem”.

```
SELECT author,title FROM classics
WHERE MATCH(author,title)
AGAINST('+charles -species' IN BOOLEAN MODE);
```

```
+-----+-----+
| author      | title        |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop |
| Charles Dickens | Little Dorrit   |
+-----+
2 rows in set (0,00 sec)
```

```
SELECT author,title FROM classics
WHERE MATCH(author,title)
AGAINST('"origin of"' IN BOOLEAN MODE);
```

```
+-----+-----+
| author      | title        |
+-----+-----+
| Charles Darwin | The Origin of Species |
+-----+
1 row in set (0,00 sec)
```

Pierwsze polecenie prosi o zwrócenie wszystkich wierszy zawierających słowo `charles`, a nie słowo `species`. Drugi używa podwójnych cudzysłów, aby zażądać zwrócenia wszystkich wierszy zawierających dokładne pochodzenie frazy.

Jeśli chcemy usunąć wiersz z tabeli, należy użyć polecenia `DELETE`. Jego składnia jest podobna do polecenia `SELECT` i umożliwia zwięźlenie dokładnego wiersza lub wierszy do usunięcia za pomocą kwalifikatorów, takich jak `WHERE` i `LIMIT`.

```
DELETE FROM classics WHERE title='Little Dorrit';
```

Ten przykład wywołuje polecenie `DELETE` dla wszystkich wierszy, których kolumna tytułu zawiera ciąg `Little Dorrit`.

Konstrukcja `UPDATE ... SET` umożliwia aktualizację zawartości pola. Jeśli chcemy zmienić zawartość jednego lub więcej pól, musimy najpierw zwiększyć tylko pole lub pola do zmiany, w podobny sposób, w jaki używamy polecenia `SELECT`.

```
UPDATE classics SET author='Mark Twain (Samuel Langhorne Clemens)'
WHERE author='Mark Twain';
UPDATE classics SET category='Classic Fiction'
WHERE category='Fiction';
```

```
+-----+-----+
| author                  | category    |
+-----+-----+
| Charles Dickens          | Classic Fiction |
| William Shakespeare       | Play         |
| Charles Darwin            | Non-Fiction  |
| Jane Austen               | Classic Fiction |
| Mark Twain (Samuel Langhorne Clemens) | Classic Fiction |
+-----+
5 rows in set (0,00 sec)
```

W pierwszym zapytaniu, prawdziwe imię `Samuela Langhorne Clemensa` `Marka Twaina` zostało dodane do jego pseudonimu w nawiasach, co dotyczyło tylko jednego wiersza. Drugie zapytanie dotyczyło jednak trzech wierszy, ponieważ zmieniło wszystkie wystąpienia słowa `Fiction` w kolumnie `category` na termin `Classic Fiction`. Wykonując aktualizację, możemy również skorzystać z wcześniejszych widzianych kwalifikatorów, takich jak `LIMIT` oraz następujących słów kluczowych `ORDER BY` i `GROUP BY`.

`ORDER BY` sortuje zwrócone wyniki według co najmniej jednej kolumny w porządku rosnącym lub malejącym.

```
SELECT author,title FROM classics ORDER BY author;
```

```
+-----+-----+
| author | title |
+-----+-----+
| Charles Darwin | The Origin of Species |
| Charles Dickens | The Old Curiosity Shop |
| Jane Austen | Pride and Prejudice |
| Mark Twain (Samuel Langhorne Clemens) | The Adventures of Tom Sawyer |
| William Shakespeare | Romeo and Juliet |
+-----+
5 rows in set (0,00 sec)
```

```
SELECT author,title FROM classics ORDER BY title DESC;
```

```
+-----+-----+
| author | title |
+-----+-----+
| Charles Darwin | The Origin of Species |
| Charles Dickens | The Old Curiosity Shop |
| Mark Twain (Samuel Langhorne Clemens) | The Adventures of Tom Sawyer |
| William Shakespeare | Romeo and Juliet |
| Jane Austen | Pride and Prejudice |
+-----+
5 rows in set (0,00 sec)
```

Jak widać, pierwsze zapytanie zwraca publikacje według autora w rosnącej kolejności alfabetycznej (ustawienie domyślne), a drugie zwraca je według tytułów w kolejności malejącej. Jeśli chcemy posortować wszystkie wiersze według autora, a następnie malejącego roku publikacji (aby wyświetlić najpierw najnowszy), możesz użyć następującego zapytania:

```
SELECT author,title,year FROM classics ORDER BY author,year DESC;
```

```
+-----+-----+-----+
| author | title | year |
+-----+-----+-----+
| Charles Darwin | The Origin of Species | 1856 |
| Charles Dickens | The Old Curiosity Shop | 1841 |
| Jane Austen | Pride and Prejudice | 1811 |
| Mark Twain (Samuel Langhorne Clemens) | The Adventures of Tom Sawyer | 1876 |
| William Shakespeare | Romeo and Juliet | 1594 |
+-----+-----+-----+
5 rows in set (0,00 sec)
```

Każdy rosnący i malejący kwalifikator ma zastosowanie do jednej kolumny. Słowo kluczowe `DESC` dotyczy tylko poprzedniej kolumny `year`. Ponieważ zezwalamy autorowi na używanie domyślnego porządku sortowania, jest on sortowany w kolejności rosnącej. Możemy również wyraźnie określić kolejność rosnącą dla tej kolumny, z tymi samymi wynikami:

```
SELECT author,title,year FROM classics ORDER BY author ASC,year DESC;
```

```
+-----+-----+-----+
| author | title | year |
+-----+-----+-----+
| Charles Darwin | The Origin of Species | 1856 |
| Charles Dickens | The Old Curiosity Shop | 1841 |
| Jane Austen | Pride and Prejudice | 1811 |
| Mark Twain (Samuel Langhorne Clemens) | The Adventures of Tom Sawyer | 1876 |
| William Shakespeare | Romeo and Juliet | 1594 |
+-----+-----+-----+
5 rows in set (0,00 sec)
```

Podobnie jak w przypadku `ORDER BY`, możemy grupować wyniki zwracane z zapytań za pomocą `GROUP BY`, co jest dobre do pobierania informacji o grupie danych. Na przykład, jeśli chcemy wiedzieć, ile publikacji z każdej kategorii znajduje się w tabeli `classics`, możemy wysłać następujące zapytanie:

```
SELECT category,COUNT(author) FROM classics GROUP BY category;
```

```
+-----+-----+
| category | COUNT(author) |
+-----+-----+
| Classic Fiction |      3 |
| Play             |      1 |
| Non-Fiction     |      1 |
+-----+-----+
3 rows in set (0,01 sec)
```

Step 10

Utrzymywanie wielu tabel w bazie danych, z których każda zawiera inny rodzaj informacji, jest całkiem normalne. Na przykład rozważmy przypadek tabeli `Customers`, która musi mieć możliwość odniesienia do publikacji zakupionych w tabeli `classics`. Utworzenie tabeli `Customers` może wyglądać następująco:

```
CREATE TABLE customers (
    name VARCHAR(128),
    isbn VARCHAR(13),
    PRIMARY KEY (isbn)) ENGINE InnoDB;
```

Następnie dodajmy parę rekordów do powyżej stworzonej tabeli oraz wyświetlimy jej zawartość:

```
INSERT INTO customers(name,isbn)
VALUES('Joe Bloggs','9780099533474');
INSERT INTO customers(name,isbn)
VALUES('Mary Smith','9780582506206');
INSERT INTO customers(name,isbn)
VALUES('Jack Wilson','9780517123201');
SELECT * FROM customers;
```

```
+-----+-----+
| name      | isbn       |
+-----+-----+
| Joe Bloggs | 9780099533474 |
| Jack Wilson | 9780517123201 |
| Mary Smith | 9780582506206 |
+-----+-----+
3 rows in set (0,00 sec)
```

Oczywiście w odpowiedniej tabeli zawierającej dane klientów byłyby również adresy, numery telefonów, adresy e-mail itd.. Ale nie są one potrzebne do tego wyjaśnienia. Tworząc nową tabelę, powinniśmy zauważyc, że ma ona coś wspólnego z tabelą `classics`: kolumnę o nazwie `isbn`. Ponieważ ma to samo znaczenie w obu tabelach (numer ISBN odnosi się do książki i zawsze tej samej książki), możemy użyć tej kolumny, aby powiązać dwie tabele w jedno zapytanie:

```
SELECT name,author,title FROM customers,classics
WHERE customers.isbn=classics.isbn;
```

```
+-----+-----+-----+
| name      | author      | title        |
+-----+-----+-----+
| Joe Bloggs | Charles Dickens | The Old Curiosity Shop |
| Jack Wilson | Charles Darwin   | The Origin of Species |
| Mary Smith  | Jane Austen    | Pride and Prejudice  |
+-----+-----+-----+
3 rows in set (0,00 sec)
```

Zapytanie to starannie połączyło ze sobą tabele, aby wyświetlić publikacje zakupione z tabeli `classics` przez osoby z tabeli `Customers`.

Używając `NATURAL JOIN`, możemy zaoszczędzić sobie trochę pisania i sprawić, że zapytania będą trochę jaśniejsze. Ten rodzaj łączenia zajmuje dwie tabele i automatycznie łączy kolumny o tej samej nazwie. Tak więc, aby osiągnąć te same wyniki, co w przykładzie powyżej, należy wpisać:

```
SELECT name,author,title FROM customers NATURAL JOIN classics;
```

```
+-----+-----+-----+
| name      | author        | title          |
+-----+-----+-----+
| Joe Bloggs | Charles Dickens | The Old Curiosity Shop |
| Jack Wilson | Charles Darwin   | The Origin of Species |
| Mary Smith  | Jane Austen    | Pride and Prejudice |
+-----+-----+-----+
3 rows in set (0,00 sec)
```

Jeśli chcemy określić kolumnę, w której chcemy połączyć dwie tabele, należy użyć konstrukcji `JOIN ... ON` :

```
SELECT name,author,title FROM customers
JOIN classics ON customers.isbn=classics.isbn;
```

```
+-----+-----+-----+
| name      | author        | title          |
+-----+-----+-----+
| Joe Bloggs | Charles Dickens | The Old Curiosity Shop |
| Jack Wilson | Charles Darwin   | The Origin of Species |
| Mary Smith  | Jane Austen    | Pride and Prejudice |
+-----+-----+-----+
3 rows in set (0,00 sec)
```

Możemy także zaoszczędzić sobie pisania i poprawić czytelność zapytań, tworząc aliasy za pomocą słowa kluczowego `AS`. Po prostu podążając za nazwą tabeli z `AS` i aliasem, którego chcemy użyć.

```
SELECT name,author,title from
customers AS cust, classics AS class WHERE cust.isbn=class.isbn;
```

```
+-----+-----+-----+
| name      | author        | title          |
+-----+-----+-----+
| Joe Bloggs | Charles Dickens | The Old Curiosity Shop |
| Jack Wilson | Charles Darwin   | The Origin of Species |
| Mary Smith  | Jane Austen    | Pride and Prejudice |
+-----+-----+-----+
3 rows in set (0,00 sec)
```

Możemy także użyć operatorów logicznych `AND`, `OR` i `NOT` w zapytaniach MySQL `WHERE`, aby dodatkowo zawęzić wybór naszych poszukiwań.

```
SELECT author,title FROM classics WHERE
author LIKE "Charles%" AND author LIKE "%Darwin";
```

```
+-----+-----+
| author      | title          |
+-----+-----+
| Charles Darwin | The Origin of Species |
+-----+-----+
1 row in set (0,00 sec)
```

```
SELECT author,title FROM classics WHERE
author LIKE "%Mark Twain%" OR author LIKE "%Samuel Langhorne Clemens%";
```

```
+-----+-----+-----+
| author                  | title          |
+-----+-----+-----+
| Mark Twain (Samuel Langhorne Clemens) | The Adventures of Tom Sawyer |
+-----+-----+-----+
1 row in set (0,01 sec)
```

```
SELECT author,title FROM classics WHERE
author LIKE "Charles%" AND author NOT LIKE "%Darwin";
```

```
+-----+-----+
| author | title      |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop |
+-----+-----+
1 row in set (0,00 sec)
```

Step 11

MySQL jest nazywany systemem zarządzania **relacyjnymi bazami danych**, ponieważ jego tabele przechowują nie tylko dane, ale także relacje między nimi. Istnieją trzy kategorie relacji: Jeden do jednego, jeden do wielu oraz wiele do wielu.

Relacja **jeden do jednego** przypomina (tradycyjne) małżeństwo: każdy element jest powiązany tylko z jednym elementem drugiego typu. Jest to zaskakująco rzadkie. Na przykład autor może napisać wiele książek, książka może mieć wielu autorów, a nawet adres może być powiązany z wieloma klientami. Założymy, że pod dowolnym adresem zawsze może być tylko jeden klient. W takim przypadku relacja Klienci-Adresy na rysunku poniżej to relacja jeden do jednego: tylko jeden klient mieszka pod każdym adresem, a każdy adres może mieć tylko jednego klienta.

Table 9-8a (Customers)

Table 9-8b (Addresses)

CustNo	Name	Address	Zip
1	Emma Brown	1565 Rainbow Road	90014
2	Darren Ryder	4758 Emily Drive	23219
3	Earl B. Thurston	862 Gregory Lane	40601
4	David Miller	3647 Cedar Lane	02154

Learning PHP, MySQL & JavaScript, chapter 8

Zwykle, gdy dwa elementy mają relację jeden do jednego, po prostu uwzględniamy je jako kolumny w tej samej tabeli. Istnieją dwa powody, dla których warto podzielić je na osobne tabele:

- Chcemy być przygotowany na wypadek, gdyby relacja zmieniła się później i nie będzie już indywidualna.
- Tabela zawiera wiele kolumn i podzielenie jej, zwiększyłoby wydajność lub konserwację.

Relacje **jeden do wielu** (lub **wiele do jednego**) występują, gdy jeden wiersz w jednej tabeli jest połączony z wieloma wierszami w innej tabeli. Założymy, że klient jest powiązany z wieloma zakupami. Możemy połączyć te dwie tabele obok siebie, gdzie przerywane linie łączące wiersze w każdej tabeli zaczynają się od pojedynczego wiersza w tabeli po lewej stronie, ale mogą łączyć się z więcej niż jednym wierszem w tabeli po prawej stronie. Ta relacja jeden do wielu jest również preferowanym schematem do opisywania relacji wiele do jednego, w którym to przypadku normalnie zamienia się lewą i prawą tabelę, aby wyświetlić je jako relację jeden do wielu.

Table 9-8a (Customers)

CustNo	Name
1	Emma Brown
2	Darren Ryder
	(etc...)
3	Earl B. Thurston
4	David Miller

Table 9-7. (Purchases)

CustNo	ISBN	Date
1	0596101015	Mar 03 2009
2	0596527403	Dec 19 2008
	0596101015	Dec 19 2008
3	0596005436	Jun 22 2009
4	0596006815	Jan 16 2009

Aby przedstawić relację jeden do wielu w relacyjnej bazie danych, należy utworzyć tabelę dla „wielu” i tabelę dla „jeden”. Tabela dla „wielu” musi zawierać kolumnę zawierającą klucz podstawowy z tabeli „jeden”. W związku z tym tabela `Purchases` będzie zawierać kolumnę zawierającą klucz podstawowy klienta.

W relacji **wiele do wielu** wiele wierszy w jednej tabeli jest połączonych z wieloma wierszami w innej tabeli. Aby utworzyć tę relację, należy dodać trzecią tabelę zawierającą tę samą kolumnę kluczową z każdej z pozostałych tabel. Ta trzecia tabela nie zawiera nic więcej, ponieważ jej jedynym celem jest połączenie innych tabel.

*Columns from
Table 9-8
(Customers)*

*Intermediary
Table 9-12
(Customer/ISBN)*

*Columns from
Table 9-4
(Titles)*

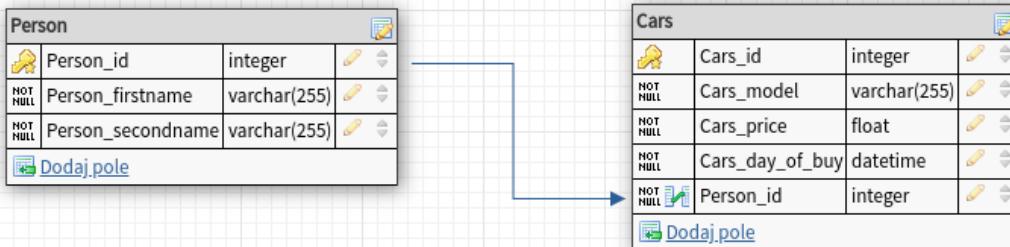
Zip	CustNo	CustNo	ISBN	ISBN	Title
90014	1	1	0596101015	0596101015	PHP Cookbook
23219	2	2	0596101015	(etc...)	
	(etc...)		0596527403	0596527403	Dynamic HTML
40601	3	3	0596005436	0596005436	PHP and MySQL
02154	4	4	0596006815	0596006815	Programming PHP

Tabela 9-12 jest właśnie taką tabelą. Została wyodrębniona z tabeli 9-7, tabeli zakupów, ale pomija informacje o dacie zakupu. Zawiera kopię ISBN każdego sprzedanego tytułu wraz z numerem klienta każdego nabywcy. Mając tę tabelę pośredniczącą, możemy przechodzić przez wszystkie informacje w bazie danych za pomocą szeregu relacji. Możemy wziąć adres jako punkt wyjścia i znaleźć autorów każdej książki zakupionej przez klienta mieszkającego pod tym adresem. Na przykład założymy, że chcemy dowiedzieć się o zakupach w kodzie pocztowym `23219`. W tym momencie możesz skorzystać z Tabeli 9-8, aby znaleźć imię i nazwisko klienta lub skorzystać z tabeli 9-12 nowego pośrednika, aby zobaczyć zakupione książki. Stąd dowiemy się, że dwa tytuły zostały zakupione.

W języku SQL aby utworzyć relację między tabelami należy dodać tzw. klucz obcy korzystając z polecenia `ALTER TABLE`, którego składnia wygląda następująco:

```
ALTER TABLE tabela1 ADD CONSTRAINT nazwa_ograniczenia FOREIGN KEY (tabela2) REFERENCES
tabela2(identyfikator_tabeli);
```

W powyższym przykładzie łączymy tabelę1 poprzez pole z tabeli2. Ważne jest, że pole z którymi chcemy dodać klucz obcy musi istnieć w tabeli1. Poniższy skrypt tworzy relację jeden do wielu dla następującej bazy danych:



```
CREATE TABLE `Person` (
    `Person_id` INT NOT NULL AUTO_INCREMENT,
    `Person_firstname` varchar(255) NOT NULL,
    `Person_secondname` varchar(255) NOT NULL,
    PRIMARY KEY (`Person_id`)
);

CREATE TABLE `Cars` (
    `Cars_id` INT NOT NULL AUTO_INCREMENT,
    `Cars_model` varchar(255) NOT NULL,
    `Cars_price` FLOAT NOT NULL,
    `Cars_day_of_buy` DATETIME NOT NULL,
    `Person_id` INT NOT NULL,
    PRIMARY KEY (`Cars_id`)
);

ALTER TABLE `Cars` ADD CONSTRAINT `Cars_fk0` FOREIGN KEY (`Person_id`) REFERENCES `Person`(`Person_id`);
```

Step 12

Najpopularniejszą platformą bazodanową używaną w PHP jest baza danych MySQL. Jeśli spojrzymy na witrynę MySQL, zobaczymy, że istnieje kilka różnych wersji MySQL, z których możemy korzystać. PHP ma również wiele różnych interfejsów do tego narzędzia bazy danych, więc przyjrzymy się interfejsowi obiektowemu znanemu jako **MySQLi**, czyli rozszerzenie **MySQL Improved**.

Dokumentacja do interfejsu MySQLi : <https://www.php.net/manual/en/book.mysqli.php>
(<https://www.php.net/manual/en/book.mysqli.php>).

Uwaga. Aby działał interfejs MySQLi należy w pliku `php.ini` w zakładce `extensions` odkomentować linię `extension=mysqli` (wcześniej należy zainstalować wsparcie MySQL dla PHP - w systemie Linux poprzez instalację pakietu `php*-mysql`, gdzie * oznacza wersję PHP).

MySQL udostępnia interfejs obiektowy. Zapewnia zarówno zorientowane obiektywnie, jak i proceduralne podejście do obsługi operacji bazy danych.

```
<?php
$mysqli = mysqli_connect("localhost", "root", "silneHaslo123@", "example");

$result = mysqli_query($mysqli, "SELECT 'Wprowadzenie do MySQLi' AS _message FROM DUAL");
$row = mysqli_fetch_assoc($result);
echo $row['_message'];
?>
```

```

<?php
    $mysqli = new mysqli("localhost", "root", "silneHaslo123@", "example");

    $result = $mysqli->query("SELECT 'Wprowadzenie do MySQLi' AS _message FROM DUAL");
    $row = $result->fetch_assoc();
    echo $row['_message'];
?>

```

PHP zapewnia różne funkcje dostępu do bazy danych MySQL i manipulowania rekordami danych wewnętrz bazy danych MySQL. Wymagane jest wywołanie funkcji PHP w taki sam sposób, jak każda inna funkcja PHP. Funkcje PHP do użytku z MySQL mają następujący ogólny format.

```
mysqli function(value,value,...);
```

PHP zapewnia **utworzenie obiektu poprzez konstruktor** `mysqli` lub użycie funkcji `mysqli_connect()` w celu otwarcia połączenia z bazą danych. Ta funkcja przyjmuje sześć parametrów i zwraca identyfikator łącza MySQL na sukces lub FALSE w przypadku niepowodzenia.

```
$mysqli = new mysqli($host, $username, $passwd, $dbName, $port, $socket);
```

Wszystkie parametry są opcjonalne. Ich znaczenie przedstawia poniższa tabela:

Parametr	Znaczenie
<code>\$host</code>	Nazwa hosta z uruchomionym serwerem bazy danych. Jeśli nie zostanie określony, wartość domyślna będzie <code>localhost:3306</code> .
<code>\$username</code>	Nazwa użytkownika uzyskująca dostęp do bazy danych. Jeśli nie zostanie określony, wartość domyślna będzie nazwa użytkownika, który jest właścicielem procesu serwera.
<code>\$passwd</code>	Hasło użytkownika uzyskującego dostęp do bazy danych. Jeśli nie zostanie określony, domyślnym będzie puste hasło.
<code>\$dbname</code>	Nazwa bazy danych, na której ma być wykonywana kwerenda.
<code>\$port</code>	numer portu, aby spróbować połączyć się z serwerem MySQL
<code>\$socket</code>	gniazdo lub potok, który powinien być używany do połączenia z serwerem

W każdej chwili można odłączyć się od bazy danych MySQL za pomocą metody PHP `close()`.

```
$mysqli->close();
```

```

<html>
<head>
    <title>Connecting MySQL Server</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass);

    if($mysqli->connect_errno) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');
    $mysqli->close();
?>
</body>
</html>

```

Step 13

PHP używa funkcji `mysqli->query()` lub `mysql_query()` do tworzenia lub usuwania bazy danych MySQL. Ta funkcja przyjmuje dwa parametry i zwraca wartość PRAWDA w przypadku powodzenia lub FAŁSZ w przypadku awarii.

```
$mysqli->query($sql,$resultmode)
```

Parametr `$sql` jest wymagany i jest to zapytanie SQL do utworzenia bazy danych. Parametr `$resultmode` jest opcjonalny i jest to stała `MYSQLI_USE_RESULT` lub `MYSQLI_STORE_RESULT`. Domyślnie używany jest `MYSQLI_STORE_RESULT`.

```
<html>
<head><title>Creating MySQL Database</title></head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass);

    if($mysqli->connect_errno) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    if ($mysqli->query("CREATE DATABASE TUTORIALS")) {
        printf("Database TUTORIALS created successfully.<br />");
    }
    if ($mysqli->errno) {
        printf("Could not create database: %s<br />", $mysqli->error);
    }

    $mysqli->close();
?>
</body>
</html>
```

Możemy zmodyfikować powyższy skrypt tak, że w przypadku istnienia bazy `TUTORIALS` zostanie ona skasowana i utworzona na nowo:

```
<html>
<head><title>Creating MySQL Database</title></head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass);

    if($mysqli->connect_errno) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    if ($mysqli->query("CREATE DATABASE TUTORIALS")) {
        printf("Database TUTORIALS created successfully.<br />");
    }
    if ($mysqli->errno) {
        if ($mysqli->query("DROP DATABASE TUTORIALS")) {
            printf("Database TUTORIALS are dropped successfully.<br />");
        }
        else{
            printf("Could not create database: %s<br />", $mysqli->error);
        }
    }

    $mysqli->close();
?>
</body>
</html>
```

PHP używa funkcji `mysqli_select_db`, aby wybrać bazę danych, na której mają być wykonywane kwerendy. Ta funkcja przyjmuje dwa parametry i zwraca wartość PRAWDA w przypadku powodzenia lub FAŁSZ w przypadku awarii.

```
mysqli_select_db ( mysqli $link , string $dbname ) : bool
```

Parametr `$link` jest wymagany i jest to identyfikator łącza zwrócony przez `mysqli_connect()`. Parametr `$dbname` jest wymagany i jest to nazwa bazy danych do połączenia.

```
<html>
<head>
    <title>Selecting MySQL Database</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $conn = mysqli_connect($dbhost, $dbuser, $dbpass);

    if(! $conn ) {
        die('Could not connect: ' . mysqli_error($conn));
    }
    echo 'Connected successfully<br />';

    $retval = mysqli_select_db( $conn, 'TUTORIALS' );

    if(! $retval ) {
        die('Could not select database: ' . mysqli_error($conn));
    }
    echo "Database TUTORIALS selected successfully\n";
    mysqli_close($conn);
?>
</body>
</html>
```

Metoda `query` może stanowić również podstawę tworzenia tabeli za pomocą języka PHP:

```

<html>
<head>
    <title>Creating MySQL Table</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $dbname = 'TUTORIALS';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

    if($mysqli->connect_errno ) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    $sql = "CREATE TABLE tutorials_tbl( ".
        "tutorial_id INT NOT NULL AUTO_INCREMENT, ".
        "tutorial_title VARCHAR(100) NOT NULL, ".
        "tutorial_author VARCHAR(40) NOT NULL, ".
        "submission_date DATE, ".
        "PRIMARY KEY ( tutorial_id )); ";
    if ($mysqli->query($sql)) {
        printf("Table tutorials_tbl created successfully.<br />");
    }
    if ($mysqli->errno) {
        if ($mysqli->query("DROP TABLE tutorials_tbl")){
            printf("Table tutorials_tbl dropped successfully");
        }
        else {
            printf("Could not create table: %s<br />", $mysqli->error);
        }
    }
    $mysqli->close();
?>
</body>
</html>

```

Step 14

Kolejny przykład korzysta z metody query w celu dodania rekordów do bazy danych:

```

<html>
<head>
    <title>Add New Record in MySQL Database</title>
</head>
<body>
<?php
    if(isset($_POST['add'])) {
        $dbhost = 'localhost';
        $dbuser = 'root';
        $dbpass = 'silneHaslo123@';
        $dbname = 'TUTORIALS';
        $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

        if($mysqli->connect_errno ) {
            printf("Connect failed: %s<br />", $mysqli->connect_error);
            exit();
        }
        printf('Connected successfully.<br />');

        $tutorial_title = $_POST['tutorial_title'];
        $tutorial_author = $_POST['tutorial_author'];

        $submission_date = $_POST['submission_date'];
        $sql = "INSERT INTO tutorials_tbl ".
            "(tutorial_title,tutorial_author, submission_date) "."VALUES ".
            "('$tutorial_title','$tutorial_author','$submission_date')";

        if ($mysqli->query($sql)) {
            printf("Record inserted successfully.<br />");
        }
        if ($mysqli->errno) {
            printf("Could not insert record into table: %s<br />", $mysqli->error);
        }
        $mysqli->close();
    } else {
        //$_PHP_SELF oznacza wykonanie skryptu będącego w pliku .php (w tym przypadku kodu powyżej)
    ?>
        <form method = "post" action = "<?php $_PHP_SELF ?>">
            <table width = "600" border = "0" cellspacing = "1" cellpadding = "2">
                <tr>
                    <td width = "250">Tutorial Title</td>
                    <td><input name = "tutorial_title" type = "text" id = "tutorial_title"></td>
                </tr>
                <tr>
                    <td width = "250">Tutorial Author</td>
                    <td><input name = "tutorial_author" type = "text" id = "tutorial_author"></td>
                </tr>
                <tr>
                    <td width = "250">Submission Date [ yyyy-mm-dd ]</td>
                    <td><input name = "submission_date" type = "text" id = "submission_date"></td>
                </tr>
                <tr>
                    <td width = "250"> </td>
                    <td></td>
                </tr>
                <tr>
                    <td width = "250"> </td>
                    <td><input name = "add" type = "submit" id = "add" value = "Add Tutorial"></td>
                </tr>
            </table>
        </form>
    <?php
    }
?>
</body>
</html>

```

Mając możliwość wstawiania danych, możemy również sprawdzić czy zostały one dodane poprzez użycie zapytania SELECT:

```

<html>
<head>
    <title>Creating MySQL Table</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $dbname = 'TUTORIALS';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

    if($mysqli->connect_errno) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    $sql = "SELECT tutorial_id, tutorial_title, tutorial_author, submission_date FROM tutorials_tbl";
    $result = $mysqli->query($sql);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            printf("Id: %s, Title: %s, Author: %s, Date: %d <br />",
                $row["tutorial_id"],
                $row["tutorial_title"],
                $row["tutorial_author"],
                $row["submission_date"]);
        }
    } else {
        printf('No record found.<br />');
    }
    mysqli_free_result($result);
    $mysqli->close();
?>
</body>
</html>

```

Używana jest tutaj metoda obiektu o nazwie `fetch_assoc()`, aby dostarczyć jeden wiersz danych naraz i przechowywać ten pojedynczy wiersz w zmiennej o nazwie `$row`. Pętla `while` trwa tak długo, jak długo istnieją wiersze do przetworzenia. W ramach pętli `while` zrzucana jest zawartość zmiennej `$row` do okna przeglądarki. Na koniec zamknięty jest zarówno wynik, jak i obiekty bazy danych.

Kolejny przykład aktualizuje rekordy w bazie danych korzystając z metody `query`:

```

<html>
<head>
    <title>Updating MySQL Table</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $dbname = 'TUTORIALS';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

    if($mysqli->connect_errno ) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    if ($mysqli->query('UPDATE tutorials_tbl set tutorial_title = "Learning Java" where tutorial_id = 2'))
    {
        printf("Table tutorials_tbl updated successfully.<br />");
    }
    if ($mysqli->errno) {
        printf("Could not update table: %s<br />", $mysqli->error);
    }
    $sql = "SELECT tutorial_id, tutorial_title, tutorial_author, submission_date FROM tutorials_tbl";

    $result = $mysqli->query($sql);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            printf("Id: %s, Title: %s, Author: %s, Date: %d <br />",
                $row["tutorial_id"],
                $row["tutorial_title"],
                $row["tutorial_author"],
                $row["submission_date"]);
        }
    } else {
        printf('No record found.<br />');
    }
    mysqli_free_result($result);
    $mysqli->close();
?>
</body>
</html>

```

Ostatni w tej sekcji przykład usuwa dane z tabeli za pomocą metody query:

```

<html>
<head>
    <title>Deleting MySQL Table record</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $dbname = 'TUTORIALS';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

    if($mysqli->connect_errno) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    if ($mysqli->query('DELETE FROM tutorials_tbl where tutorial_id = 1')) {
        printf("Table tutorials_tbl record deleted successfully.<br />");
    }
    if ($mysqli->errno) {
        printf("Could not delete record from table: %s<br />", $mysqli->error);
    }

    $sql = "SELECT tutorial_id, tutorial_title, tutorial_author, submission_date FROM tutorials_tbl";
    $result = $mysqli->query($sql);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            printf("Id: %s, Title: %s, Author: %s, Date: %d <br />",
                $row["tutorial_id"],
                $row["tutorial_title"],
                $row["tutorial_author"],
                $row["submission_date"]);
        }
    } else {
        printf('No record found.<br />');
    }
    mysqli_free_result($result);
    $mysqli->close();
?>
</body>
</html>

```

Step 15

Rozważmy tabelę `tcount_tbl`, której zawartość wygląda następująco:

tutorial_author	tutorial_count
Mahesh	3
Suresh	1
Mateusz Miotk	20
Mateusz	NULL
Kazik	NULL

Najpierw utworzymy ją za pomocą zapytania SQL (wraz z dodaniem powyższych danych):

```

create table tcount_tbl(
    tutorial_author VARCHAR(40) NOT NULL,
    tutorial_count int
);
insert into tcount_tbl values('Mahesh', 3);
insert into tcount_tbl values('Suresh', 1);
insert into tcount_tbl values('Mateusz Miotk', 20);
insert into tcount_tbl values('Mateusz', NULL);
insert into tcount_tbl values('Kazik', NULL);

```

Teraz połączmy te tabelę za pomocą `JOIN` przy użyciu metody `query` w PHP:

```

<html>
<head>
    <title>Handling NULL</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $dbname = 'TUTORIALS';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

    if($mysqli->connect_errno ) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    $sql = 'SELECT a.tutorial_id, a.tutorial_author, b.tutorial_count
            FROM tutorials_tbl a, tcount_tbl b
            WHERE a.tutorial_author = b.tutorial_author';

    $result = $mysqli->query($sql);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            printf("Id: %s, Author: %s, Count: %d <br />",
                $row["tutorial_id"],
                $row["tutorial_author"],
                $row["tutorial_count"]);
        }
    } else {
        printf('No record found.<br />');
    }
    mysqli_free_result($result);
    $mysqli->close();
?>
</body>
</html>

```

Korzystając z konstrukcji `if... warunek else`, możemy przygotować obsłużyć zapytanie, które będzie posiadało wartości NULL.

```

<html>
<head>
    <title>Handling NULL</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $dbname = 'TUTORIALS';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    $tutorial_count = null;
    if($mysqli->connect_errno) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');
    if( isset($tutorial_count) ) {
        $sql = 'SELECT tutorial_author, tutorial_count
                FROM tcount_tbl
                WHERE tutorial_count = ' + $tutorial_count;
    } else {
        $sql = 'SELECT tutorial_author, tutorial_count
                FROM tcount_tbl
                WHERE tutorial_count IS NULL';
    }
    $result = $mysqli->query($sql);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            printf("Author: %s, Count: %d <br />",
                $row["tutorial_author"],
                $row["tutorial_count"]);
        }
    } else {
        printf('No record found.<br />');
    }
    $mysqli->close();
?>
</body>
</html>

```

Z MySQL możemy uzyskać trzy rodzaje informacji: o wynikach zapytań (co było pokazane dotychczas), o tabelach i bazach danych oraz o serwerze MySQL. Aby dowiedzieć się jaką jest domyślna użyta baza danych możemy skorzystać z następującego kodu:

```

<html>
<head>
    <title>Getting MySQL Database Info</title>
</head>
<body>
<?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $dbname = 'TUTORIALS';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    $tutorial_count = null;

    if($mysqli->connect_errno) {
        printf("Connect failed: %s<br />", $mysqli->connect_error);
        exit();
    }
    printf('Connected successfully.<br />');

    if ($result = mysqli_query($mysqli, "SELECT DATABASE()")) {
        $row = mysqli_fetch_row($result);
        printf("Default database is %s<br />", $row[0]);
        mysqli_free_result($result);
    }
    $mysqli->close();
?>
</body>
</html>

```

Oprócz `SELECT DATABASE()` mamy również do dyspozycji `SELECT VERSION()`, `SELECT USER()`, `SHOW STATUS` oraz `SHOW VARIABLES`.

7.1 7.1 Teoria

Step 1

Istnieją dwa sposoby uzyskiwania dostępu do baz danych z poziomu PHP. Jednym z nich jest użycie rozszerzenia specyficznego dla bazy danych; drugim jest użycie niezależnej od bazy danych biblioteki **PDO**. Każde podejście ma swoje zalety i wady. Jeśli używamy rozszerzenia specyficznego dla bazy danych, kod jest ściśle powiązany z bazą danych, która jest używana. Na przykład nazwy funkcji, parametry, obsługa błędów itp. Rozszerzenia MySQL są zupełnie inne niż w innych rozszerzeniach bazy danych. Jeśli chcemy przenieść bazę danych z MySQL do PostgreSQL, będzie to wiązało się ze znacznymi zmianami w kodzie. **PDO** ukrywa funkcje specyficzne dla bazy danych za pomocą warstwy abstrakcji, więc poruszanie się między systemami baz danych może być tak proste, jak zmiana jednej linii programu lub pliku **php.ini**. Przenośność warstwy abstrakcji, takiej jak biblioteka PDO, ma jednak swoją cenę ponieważ kod, który go używa, jest również zwykle nieco wolniejszy niż kod, który używa natywnego rozszerzenia specyficznego dla bazy danych.

Rozszerzenie **PHP Data Objects (PDO)** definiuje lekki, spójny interfejs do uzyskiwania dostępu do baz danych w PHP. Każdy sterownik bazy danych, który implementuje interfejs PDO, może udostępniać funkcje specyficzne dla bazy danych jako zwykłe funkcje rozszerzeń. Należy pamiętać, że nie można samodzielnie wykonywać żadnych funkcji bazy danych przy użyciu rozszerzenia PDO; aby uzyskać dostęp do serwera bazy danych, należy użyć sterownika PDO specyficznego dla bazy danych.

Do innych cech PDO należą:

- To natywne rozszerzenie napisane w języku C.
- Wykorzystuje najnowsze wewnętrzne funkcje PHP 7
- Używa buforowanego odczytu danych z zestawu wyników
- Zapewnia podstawowe funkcje bazy danych
- Nadal ma dostęp do funkcji specyficznych dla bazy danych
- Potrafi korzystać z technik transakcyjnych
- Może współdziałać z LOBS (Large Objects) w bazie danych
- Potrafi używać przygotowanych i wykonywalnych instrukcji SQL z powiązanymi parametrami
- Może implementować przewijalne kursorы
- Posiada dostęp do kodów błędów SQLSTATE i bardzo elastyczną obsługę błędów

PDO zawiera sterowniki dla prawie wszystkich istniejących silników baz danych, a te sterowniki, których PDO nie dostarcza, powinny być dostępne przez ogólne połączenie **ODBC**. Moduł PDO musi mieć włączone rozszerzenie PDO specyficzne dla bazy danych, z którą będziemy się łączyć. Na przykład, aby ustawić PDO na systemie Linux w celu interakcji z MySQL, po prostu należy wprowadzić następującą linię do pliku **php.ini** i zrestartować serwer:

```
extension=pdo_mysql
```

Pierwszym wymaganiem dla PDO jest nawiązanie połączenia z daną bazą danych i utrzymanie tego połączenia w zmiennej, jak w poniższym kodzie:

```
<?php  
    $dbuser = 'root';  
    $dbpass = 'silneHaslo123@';  
    $db = new PDO("mysql:host=localhost;dbname=TUTORIALS", $dbuser,$dbpass)  
?>
```

Połączenie nawiązane za pomocą obiektu PDO pozostaje aktywne przez cały czas życia obiektu i jest kończone przy usuwaniu obiektu z pamięci. Zostanie to wykonane automatycznie po zakończeniu pracy skryptu lub po przypisaniu wartości `null` zmiennej obiektowej przechowującej odwołanie do obiektu.

Step 2

Wróćmy do bazy `classics`, którego zawartość tabeli wygląda następująco:

author	title	category	year	isbn
Charles Dickens	The Old Curiosity Shop	Classic Fiction	1841	9780099533474
William Shakespeare	Romeo and Juliet	Play	1594	9780192814968
Charles Darwin	The Origin of Species	Non-Fiction	1856	9780517123201
Jane Austen	Pride and Prejudice	Classic Fiction	1811	9780582506206
Mark Twain	The Adventures of Tom Sawyer	Classic Fiction	1876	9781598184891

Po nawiązaniu połączenia z silnikiem bazy danych i bazą danych, z którą chcesz współdziałać, możemy użyć tego połączenia do wysyłania poleceń SQL do serwera. Prosta instrukcja `UPDATE` wyglądałaby następująco:

```
<html>
<head>
    <title>Getting MySQL Database Info with PDO</title>
</head>
<body>
<?php
    $dbuser = 'root';
    $dbpass = 'silneHaslo123@';
    $db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
    $db->query("Update classics SET author='New Mark Twain' WHERE year=1876");

?>
</body>
</html>
```

author	title	category	year	isbn
Charles Dickens	The Old Curiosity Shop	Classic Fiction	1841	9780099533474
William Shakespeare	Romeo and Juliet	Play	1594	9780192814968
Charles Darwin	The Origin of Species	Non-Fiction	1856	9780517123201
Jane Austen	Pride and Prejudice	Classic Fiction	1811	9780582506206
New Mark Twain	The Adventures of Tom Sawyer	Classic Fiction	1876	9781598184891

Ten kod po prostu aktualizuje tabelę książek i zwalnia zapytanie. Pozwala to na wysyłanie prostych poleceń SQL (np. `UPDATE`, `DELETE`, `INSERT`) bezpośrednio do bazy danych.

Metoda `query` zwraca obiekt typu `PDOStatement` pozwalający na odczyt danych po wykonaniu zapytania, gdy zakończyło się sukcesem, lub też wartość `false` w przeciwnym razie.

Wspomniany obiekt zawiera metodę `fetch`, która udostępnia pobrane dane. Jej ogólne wywołanie ma postać:

```
fetch([typ_wyniku])
```

Zwracaną wartością jest kolejny wiersz z wyników zapytania lub wartość `false`, jeżeli kolejnego wiersza nie uda się pobrać (np. zostały już odczytane wszystkie dane). Postać zwróconych danych zależy od stanu argumentu `typ_wyniku`, który może przyjmować następujące wartości:

- `PDO::FETCH_ASSOC` – zwraca tablicę asocjacyjną, w której nazwy kolumn wynikowych są kluczami,
- `PDO::FETCH_BOTH` – zwraca tablicę indeksowaną zarówno numerycznie, jak i asocjacyjnie (jest to wartość domyślna),
- `PDO::FETCH_BOUND` – zwraca wartość `true` oraz przypisuje wartość z kolumn wyniku do zmiennych PHP ustalonych wcześniej za pomocą wywołania metody `bindColumn()`,
- `PDO::FETCH_CLASS` – zwraca nową instancję klasy, dokonując mapowania kolumn wynikowych na właściwości klasy,
- `PDO::FETCH_INTO` – uaktualnia istniejącą instancję klasy, dokonując mapowania kolumn wynikowych na właściwości klasy,
- `PDO::FETCH_LAZY` – kombinacja `PDO::FETCH_BOTH` i `PDO::FETCH_OBJ`,
- `PDO::FETCH_NUM` – zwraca tablicę indeksowaną numerycznie,
- `PDO::FETCH_OBJ` – zwraca obiekt z właściwościami o nazwach i wartościach odpowiadającym kolumnom wynikowym zapytania

Aby zmienić domyślny tryb obowiązujący dla wszystkich zapytań (czyli standardowe `PDO::FETCH_BOTH`), należy wykorzystać metodę `setFetchMode`, której wywołanie ma postać:

```
setFetchMode([domyślny_typ_wyniku])
```

gdzie `domyślny_typ_wyniku` to jedna z wartości wymienionych wyżej. Jeśli chcemy na przykład, aby domyślnym typem wyniku była tablica indeksowana numerycznie, zastosujemy wywołanie:

```
$result->setFetchMode(PDO::FETCH_NUM);
```

```
<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
$db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
$query = "SELECT * FROM classics";
$result = $db->query($query);
if (!$result){
    echo "BŁĄD w zapytaniu";
    exit;
}
while($row = $result->fetch(PDO::FETCH_ASSOC)){
    print_r($row);
}
?>
```

```
Array
(
    [author] => Charles Dickens
    [title] => The Old Curiosity Shop
    [category] => Classic Fiction
    [year] => 1841
    [isbn] => 9780099533474
)
Array
(
    [author] => William Shakespeare
    [title] => Romeo and Juliet
    [category] => Play
    [year] => 1594
    [isbn] => 9780192814968
)
Array
(
    [author] => Charles Darwin
    [title] => The Origin of Species
    [category] => Non-Fiction
    [year] => 1856
    [isbn] => 9780517123201
)
Array
(
    [author] => Jane Austen
    [title] => Pride and Prejudice
    [category] => Classic Fiction
    [year] => 1811
    [isbn] => 9780582506206
)
Array
(
    [author] => New Mark Twain
    [title] => The Adventures of Tom Sawyer
    [category] => Classic Fiction
    [year] => 1876
    [isbn] => 9781598184891
)
```

```

<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
$db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
$query = "SELECT * FROM classics";
$result = $db->query($query);
if (!$result){
    echo "BŁĄD w zapytaniu";
    exit;
}
$result->setFetchMode(PDO::FETCH_OBJ);
while($row = $result->fetch()){
    print_r($row);
}
?>

```

```

stdClass Object
(
    [author] => Charles Dickens
    [title] => The Old Curiosity Shop
    [category] => Classic Fiction
    [year] => 1841
    [isbn] => 9780099533474
)
stdClass Object
(
    [author] => William Shakespeare
    [title] => Romeo and Juliet
    [category] => Play
    [year] => 1594
    [isbn] => 9780192814968
)
stdClass Object
(
    [author] => Charles Darwin
    [title] => The Origin of Species
    [category] => Non-Fiction
    [year] => 1856
    [isbn] => 9780517123201
)
stdClass Object
(
    [author] => Jane Austen
    [title] => Pride and Prejudice
    [category] => Classic Fiction
    [year] => 1811
    [isbn] => 9780582506206
)
stdClass Object
(
    [author] => New Mark Twain
    [title] => The Adventures of Tom Sawyer
    [category] => Classic Fiction
    [year] => 1876
    [isbn] => 9781598184891
)

```

Zwykle będziemy używać tzw. **prepared statements** w PDO. Rozważmy następujący kod:

```

<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
$db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
$statement = $db->prepare("SELECT * FROM classics");
$statement->execute();
while($row = $statement->fetch()){
    print_r($row);
}
$statement = null;
?>

```

```
<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
$db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
$statement = $db->prepare("SELECT * FROM classics");
$statement->execute();
var_dump($statement->fetchAll());
$statement = null;
?>
```

W tym kodzie kod SQL jest przygotowywany, a następnie wykonywany. Następnie przechodzimy przez wynik z kodem while i na koniec zwalniamy obiekt, przypisując mu wartość `null`. To może nie wyglądać tak potężnie w tym prostym przykładzie, ale istnieją inne funkcje, których można użyć z przygotowanymi instrukcjami. Rozważmy teraz ten kod:

```
<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
$db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
$statement = $db->prepare("INSERT INTO classics(author, title, category, year, isbn) "
"VALUES (:author, :title, :category, :year, :isbn)");
$statement->execute(array(
    "author" => "Mateusz Miotk",
    "title" => "Introduce to Programming in PHP",
    "category" => "IT",
    "year" => 2021,
    "isbn" => "9781598184895",
));
?>
```

author	title	category	year	isbn
Charles Dickens	The Old Curiosity Shop	Classic Fiction	1841	9780099533474
William Shakespeare	Romeo and Juliet	Play	1594	9780192814968
Charles Darwin	The Origin of Species	Non-Fiction	1856	9780517123201
Jane Austen	Pride and Prejudice	Classic Fiction	1811	9780582506206
New Mark Twain	The Adventures of Tom Sawyer	Classic Fiction	1876	9781598184891
Mateusz Miotk	Introduce to Programming in PHP	IT	2021	9781598184895

Tutaj przygotowujemy instrukcję SQL z nazwanymi symbolami zastępczymi: `author`, `title`, `category`, `year` oraz `isbn`. W tym przypadku są to te same nazwy, co kolumny w bazie danych, ale jest to zrobione tylko dla przejrzystości - nazwy symboli zastępczych mogą być dowolne. W momencie wywołania zapytania zamieniamy te symbole zastępcze na rzeczywiste dane, które chcemy wykorzystać w tym konkretnym przypadku. Jedną z zalet prepared statements jest to, że można wykonać to samo polecenie SQL i za każdym razem przekazać różne wartości przez tablicę. Możemy również wykonać tego typu przygotowanie instrukcji z pozycyjnymi symbolami zastępczymi (nie nazywając ich w rzeczywistości), oznaczanymi przez symbol `?`, który jest pozycją do zastąpienia, co pokazuje następny kod:

```
<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
$db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
$statement = $db->prepare("INSERT INTO classics(author, title, category, year, isbn) "
"VALUES (?, ?, ?, ?, ?)");
$statement->execute(array(
    "Mateusz Miotk",
    "Introduce to Programming in PHP",
    "IT",
    2022,
    "9781598184845",
));
?>
```

author	title	category	year	isbn
Charles Dickens	The Old Curiosity Shop	Classic Fiction	1841	9780099533474
William Shakespeare	Romeo and Juliet	Play	1594	9780192814968
Charles Darwin	The Origin of Species	Non-Fiction	1856	9780517123201
Jane Austen	Pride and Prejudice	Classic Fiction	1811	9780582506206
Mateusz Miotk	Introduce to Programming in PHP	IT	2022	9781598184845
New Mark Twain	The Adventures of Tom Sawyer	Classic Fiction	1876	9781598184891
Mateusz Miotk	Introduce to Programming in PHP	IT	2021	9781598184895

Rezultat powyższego kodu jest taki sam jak poprzednio, ale przy mniejszej ilości kodu, ponieważ obszar wartości instrukcji SQL nie określa nazw elementów do zastąpienia, a zatem tablica w instrukcji `execute` musi przesyłać tylko surowe dane bez nazw.

Step 3

Niektóre systemy bazodanowe obsługują **transakcje**, w których można zatwierdzić serię zmian w bazie danych (wszystkie zastosowane jednocześnie) lub wycofać (odrzucone, bez żadnych zmian wprowadzonych do bazy danych). Na przykład, gdy bank obsługuje przelew pieniężny, wypłata z jednego konta i wpłata na inne musi nastąpić razem - ani jedno, ani drugie powinno się odbywać bez drugiego i nie powinno być opóźnienia między tymi dwoma działaniami. PDO elegancko obsługuje transakcje dzięki strukturze `try ... catch`, tak jak poniżej:

```
<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
try {
    $db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
} catch (Exception $error){
    die("Connection failed: " . $error->getMessage());
}
try{
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); //Do raportowania błędów
    $db->beginTransaction();
    $db->exec("DELETE FROM classics WHERE author='Mateusz Miotk'");
    $db->commit();
} catch (Exception $error) {
    $db->rollBack();
    echo "Transaction not completed: " . $error->getMessage();
}
?>
```

Jeśli całość transakcji nie może zostać zakończona, żadna z nich nie zostanie zakończona i zostanie zgłoszony wyjątek. Jeśli wywołamy `commit()` lub `rollback()` w bazie danych, która nie obsługuje transakcje, metody zwracają będą `DB_ERROR`.

Interfejs PDO udostępnia metodę wyświetlania szczegółów instrukcji PDO, co może być przydatne do debugowania, jeśli coś pójdzie nie tak.

```
<?php
$dbuser = 'root';
$dbpass = 'silneHaslo123@';
try {
    $db = new PDO("mysql:host=localhost;dbname=publications", $dbuser,$dbpass);
} catch (Exception $error){
    die("Connection failed: " . $error->getMessage());
}
$author = "Mateusz Miotk";
$statement = $db->prepare("SELECT title FROM classics WHERE author=?");
$statement->bindParam(1,$author,PDO::PARAM_STR);
$statement->execute();
$statement->debugDumpParams();
?>
```

Wywołanie metody `debugDumpParams()` wyświetla następujące rezultaty:

```
SQL: [41] SELECT title FROM classics WHERE author=?  
Sent SQL: [55] SELECT title FROM classics WHERE author='Mateusz Miotk'  
Params: 1  
Key: Position #0:  
paramno=0  
name=[0] ""  
is_param=1  
param_type=2
```

Step 4

Teraz użyjmy innego systemu bazodanowego jakim jest **PostgreSQL** (zainstalowany między innymi na serwerze szuflandia). Odpalenie konsoli **PostgreSQL** odbywa się za pomocą komendy `psql`. Najpierw utworzymy sobie tabelę o nazwie `customers`.

```
CREATE TABLE customers (firstname VARCHAR(128), secondname VARCHAR(128));
```

Następnie dodamy parę rekordów danych:

```
INSERT INTO customers(firstname, secondname) VALUES ('Mateusz', 'Miotk');  
INSERT INTO customers(firstname, secondname) VALUES ('Jan', 'Kowalski');  
INSERT INTO customers(firstname, secondname) VALUES ('Anna', 'Nowak');  
INSERT INTO customers(firstname, secondname) VALUES ('Misio', 'Pysio');
```

Wyświetlenie danych (polecenie `SELECT * FROM customers`) wyświetli wówczas następujące dane:

firstname	secondname
Mateusz	Miotk
Mateusz	Miotk
Jan	Kowalski
Anna	Nowak
Misio	Pysio

Teraz utwórzmy formularz w PHP, który będzie dodawał do bazy danych imię i nazwisko. Do podłączenia się z bazą danych skorzystamy z PDO.

```
<?php  
    //Ten kod musi być umieszczony na serwerze szuflandia  
    $dbuser = 'mmiotk';  
    $dbpass = '*****';  
    $db = new PDO("pgsql:host=localhost;port=5434", $dbuser, $dbpass);  
    $query = "SELECT * FROM customers";  
    $result = $db->query($query);  
    if (!$result){  
        echo "BŁĄD w zapytaniu";  
        exit;  
    }  
    $result->setFetchMode(PDO::FETCH_OBJ);  
    while($row = $result->fetch()){  
        print_r($row);  
    }  
?>
```

Uwaga. Aby korzystać na lokalnym komputerze z `pgsql` należy mieć zainstalowane (w systemie Linux) pakiet `php-pgsql`. Aby dowiedzieć się na jakim porcie uruchomiona jest baza danych należy w konsoli użyć polecenia `\conninfo`.

```
<html>
<head><title>Creating MySQL Database</title></head>
<body>
<form id="formularz" method="post">
    <label>Imię: </label> <input type="text" name="firstname"> <br/>
    <label>Nazwisko: </label> <input type="text" name="secondname"> <br/>
    <input type="submit">
</form>
<?php
    if ((isset($_POST['firstname'])) && (isset($_POST['secondname']))){
        //Ten kod musi być umieszczony na serwerze szuflandia
        $dbuser = 'mmiotk';
        $dbpass = '*****';
        $db = new PDO("pgsql:host=localhost;port=5434", $dbuser, $dbpass);
        $query = "INSERT INTO customers(firstname, secondname) VALUES (?,?)";
        $statement = $db->prepare($query);
        $statement->execute(array(
            $_POST['firstname'],
            $_POST['secondname'],
        ));
    }
    else{
        die("Error!!!");
    }
?>
</body>
</html>
```

Uwaga. W razie problemów z autoryzacją z pgsql, najprościej po prostu zmienić hasło za pomocą komendy (w konsoli psql):

```
ALTER USER user_name WITH PASSWORD 'new_password';
```

8.1 8.1 Teoria

Step 1

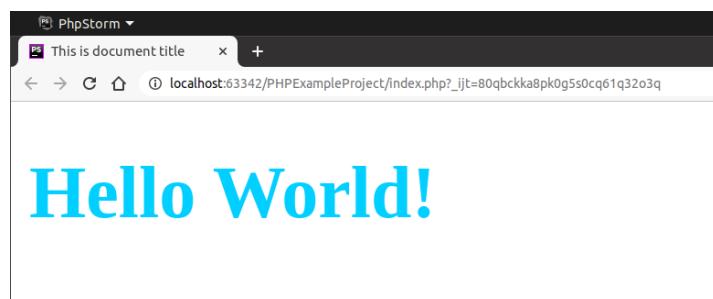
CSS służy do kontrolowania stylu strony internetowej w prosty i łatwy sposób. CSS jest skrótem od "Cascading Style Sheet", co w tłumaczeniu na język polski oznacza **Kaskadowe arkusze stylów**. Jest to prosty język mający na celu uproszczenie procesu tworzenia reprezentacyjnych stron internetowych. CSS jest koniecznością dla studentów i profesjonalistów pracujących jako web development. Do zalet CSS należą między innymi:

- **Tworzenie oszałamiającej witryny sieci Web** – CSS obsługuje wygląd i działanie części strony internetowej. Za pomocą CSS można kontrolować kolor tekstu, styl czcionek, odstępy między akapitami, sposób, w jaki kolumny są dopasowywane i rozmieszczone, jakie obrazy tła lub kolory są używane, projekty układu, różnice w wyświetlaniu dla różnych urządzeń i rozmiarów ekranu, a także wiele innych efektów.
- **Control web** - CSS jest łatwy do nauczenia się i zrozumienia, ale zapewnia potężną kontrolę nad prezentacją dokumentu HTML. Najczęściej CSS jest łączony z językami znaczników HTML lub XHTML.
- **Ucz się innych języków** - Gdy poznamy podstawy HTML i CSS, inne powiązane technologie, takie jak javascript, php stają się łatwiejsze do zrozumienia.
- **CSS oszczędza czas** - Można napisać CSS raz, a następnie ponownie użyć tego samego arkusza na wielu stronach HTML. Można zdefiniować styl dla każdego elementu HTML i zastosować go do dowolnej liczby stron sieci Web.
- **Strony ładują się szybciej** - Jeśli używany jest CSS, nie trzeba za każdym razem pisać atrybutów tagu HTML. Wystarczy napisać jedną regułę CSS tagu i zastosować ją do wszystkich wystąpień tego tagu. Więc mniej kodu oznacza krótszy czas pobierania.
- **Łatwa konserwacja** - Aby wprowadzić globalne zmiany, wystarczy zmienić styl, a wszystkie elementy na wszystkich stronach internetowych zostaną automatycznie zaktualizowane.
- **Zgodność wielu urządzeń** – arkusze stylów umożliwiają optymalizację zawartości dla więcej niż jednego typu urządzenia. Korzystając z tego samego dokumentu HTML, można przedstawić różne wersje witryny sieci Web dla urządzeń przenośnych, takich jak tablety, telefony komórkowe lub wynik strony do druku.
- **Globalne standardy sieci web** – atrybuty HTML są przestarzałe i dlatego zaleca się używanie CSS. Dlatego warto zacząć używać CSS na wszystkich stronach HTML, aby były kompatybilne z przyszłymi przeglądarkami.

Hello world napisany w HTML przy użyciu CSS może wyglądać następująco:

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is document title</title>
    <style>
      h1 {
        color: #36CFFF;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Strona wówczas wygląda następująco:



CSS jest tworzony i utrzymywany przez grupę osób w ramach **W3C** (<https://www.w3.org/>) o nazwie CSS Working Group. CSS Working Group tworzy dokumenty zwane specyfikacjami. Gdy specyfikacja została omówiona i oficjalnie ratyfikowana przez członków W3C, staje się ona zaleceniem. Te ratyfikowane specyfikacje są nazywane zaleceniami, ponieważ W3C nie ma kontroli nad faktycznym wdrożeniem języka. Niezależne firmy i organizacje tworzą to oprogramowanie.

UWAGA – World Wide Web Consortium, lub W3C to grupa, która wydaje zalecenia dotyczące tego, jak działa Internet i jak powinien on ewoluować.

Kaskadowe arkusze stylów poziomu 1 (**CSS1**) powstały jako rekommendacja z W3C w grudniu 1996 roku. Ta wersja opisuje język CSS, a także prosty model formatowania wizualnego dla wszystkich tagów HTML. **CSS2** stał się zaleceniem W3C w maju 1998 roku i opiera się na **CSS1**. Ta wersja dodaje obsługę arkuszy stylów specyficznych dla nośnika, takich jak drukarki i urządzenia fonetyczne, czcionki do pobrania, pozycjonowanie elementów i tabele. Obecnie korzysta się ze standardu **CSS3**, który składa się z bardzo wiele tzw. modułów.

Strona główna do standardu CSS w ramach W3C znajduje się pod adresem: <https://www.w3.org/Style/CSS/Overview.en.html> (<https://www.w3.org/Style/CSS/Overview.en.html>).

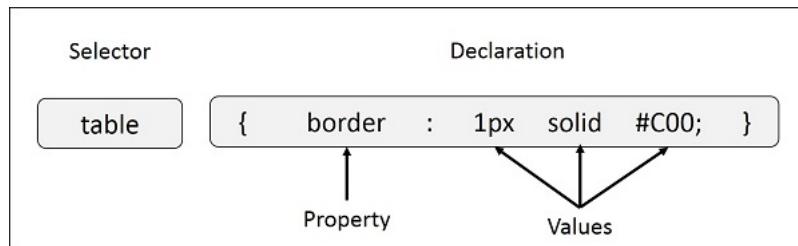
Step 2

Css składa się z reguł stylu, które są interpretowane przez przeglądarkę, a następnie stosowane do odpowiednich elementów w dokumencie. Reguła stylu składa się z trzech części –

- **Selektor (ang. selector)** – Selektor to znacznik HTML, pod którym zostanie zastosowany styl. Może to być dowolny tag, taki jak `<h1>` lub `<table>` itp.
- **Właściwość (ang. property)** – Właściwość jest typem atrybutu tagu HTML. Mówiąc prościej, wszystkie atrybuty HTML są konwertowane na właściwości CSS. Mogą to być `color`, `border` itp.
- **Wartość (ang. value)** – Wartości są przypisywane do właściwości. Na przykład `color` właściwość może mieć wartość `red` lub `#F1F1F1` itp.

Składnia reguły stylu CSS można umieścić w następujący sposób:

```
selector { property: value }
```



Powyzsze obramowanie tabeli w języku CSS można zapisać następująco:

```
table{ border :1px solid #C00; }
```

W tym miejscu `table` jest **selektem**, a `border` jest **właściwością**, a podana wartość `1px solid #C00` jest wartością tej właściwości. Istnieją różne rodzaje selektorów. Są nimi między innymi:

- **Selektory typów** - to taki jak wyżej. Można zapisać go bardziej przejrzystie, a mianowicie:

```
h1 {  
    color: #36CFFF;  
}
```

- **Selektory uniwersalne** - po prostu pasuje do nazwy dowolnego typu elementu.

```
* {  
    color: #000000;  
}
```

Ta reguła powoduje, że zawartość każdego elementu w naszym dokumencie jest czarna.

- **Selektory podrzędne** - Założymy, że chcemy zastosować regułę stylu do określonego elementu tylko wtedy, gdy znajduje się wewnątrz określonego elementu. Jak podano w poniższym przykładzie, reguła stylu będzie stosowana do elementu programu `` tylko wtedy, gdy znajduje się wewnątrz tagu ``.

```

<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        ul em {
            color: #1c74cb;
        }
    </style>
</head>
<body>
<h1>Hello World!</h1>
<ul>
    <p>Ala ma kota</p><br/>
    <em>Kot ma ale</em>
</ul>
</body>
</html>

```

- **Selektory klas** - Reguły stylu można zdefiniować na podstawie atrybutu klasy elementów. Wszystkie elementy posiadające tę klasę zostaną sformatowane zgodnie ze zdefiniowaną regułą.

```

.black {
    color: #000000;
}

```

Ta reguła renderuje zawartość w kolorze czarnym dla każdego elementu z atrybutem klasy *ustawionym* na `black` w naszym dokumencie.

```

h1.black {
    color: #000000;
}

```

Ta reguła renderuje zawartość w kolorze czarnym tylko dla elementów `<h1>` z atrybutem klasy *ustawionym* na `black`.

```

<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        h1.color {
            color: #e51366;
        }
    </style>
</head>
<body>
<h1>Hello World!</h1>
<h1 class="color">Kolorowy tytuł</h1>
</body>
</html>

```

- **Selektory identyfikatorów** - Reguły stylu można definiować na podstawie atrybutu `id` elementów. Wszystkie elementy o tym `id` zostaną sformatowane zgodnie ze zdefiniowaną regułą.

```

#black {
    color: #000000;
}

```

Ta reguła renderuje zawartość w kolorze czarnym dla każdego elementu z atrybutem `id` ustawionym na `black` w naszym dokumencie.

```

h1#black {
    color: #000000;
}

```

Ta reguła renderuje zawartość w kolorze czarnym tylko dla elementów `<h1>` z atrybutem `id` ustawionym na `black`. Prawdziwa moc selektorów `id` jest wtedy, gdy są one używane jako podstawa dla selektorów podrzędnych.

```

#black h2 {
    color: #000000;
}

```

W tym przykładzie wszystkie elementy `<h2>` będą wyświetlane w kolorze czarnym, gdy te nagłówki będą znajdować się w tagach o `id` z atrybutem ustawionym na `black`.

```

<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        #color h2 {
            color: #ea0a0a;
        }
    </style>
</head>
<body>
<h1>Hello World!</h1>
<h2>Nagłówek drugiego poziomu</h2>
<div id="color">
    <h2>Kolorowy nagłówek drugiego poziomu</h2>
</div>
</body>
</html>

```

- Child selectors - Podobne co do zasad jak selektory podrzędne.

```

body > p {
    color: #000000;
}

```

Ta reguła spowoduje, że wszystkie akapity będą czarne, jeśli są bezpośrednim elementem podrzędnym elementu `<body>`. Inne akapity umieszczone wewnątrz innych elementów, takich jak `<div>` lub `<td>` nie miałyby żadnego wpływu na tę regułę.

```

<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        body > p {
            color: chartreuse;
        }
    </style>
</head>
<body>
<h1>Hello World!</h1>
<h2>Nagłówek drugiego poziomu</h2>
<p>
    Piszę w innym kolorze
</p>
<div>
    <p> Piszę w domyślnym kolorze</p>
</div>
</body>
</html>

```

- Selektory atrybutów - Style można również stosować do elementów HTML z określonymi atrybutami.

```

input[type = "text"] {
    color: #000000;
}

```

Powyższa reguła stylu będzie zgodna ze wszystkimi elementami wejściowymi posiadającymi atrybut typu z wartością `text`. Zaletą tej metody jest to, że element `<input type = "submit" />` pozostaje nienaruszony, a kolor zastosowany tylko do żądanych pól tekstowych.

```

<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        input[type = "text"] {
            color: #f10606;
        }
    </style>
</head>
<body>
    <h1>Hello World!</h1>
    <input type="submit"/>
    <input type="text"/>
</body>
</html>

```

Istnieją następujące reguły stosowane do selektora atrybutów.

- p[lang]** – Wybiera wszystkie elementy akapitu z atrybutem `lang`.
- p[lang="fr"]** – Wybiera wszystkie elementy akapitu, których atrybut `lang` ma wartość dokładnie "fr".

- **p[lang~="fr"]** – Wybiera wszystkie elementy akapitu, których atrybut `lang` zawiera słowo "fr".
- **p[lang|= "en"]** – Wybiera wszystkie elementy akapitu, których atrybut `lang` zawiera wartości, które są dokładnie "en" lub zaczynają się od "en-".

Step 3

Może być konieczne zdefiniowanie wielu reguł stylu dla pojedynczego elementu. Reguły te można zdefiniować w celu połączenia wielu właściwości i odpowiadających im wartości w pojedynczy blok, zgodnie z definicją, co pokazuje poniższy przykład:

```
<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        h1 {
            color: #36C;
            font-weight: normal;
            letter-spacing: .4em;
            margin-bottom: 1em;
            text-transform: lowercase;
        }
    </style>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

Tutaj wszystkie pary właściwości i wartości są oddzielone **średnikiem (;**). Można je przechowywać w jednym wierszu lub wielu wierszach. Aby uzyskać lepszą czytelność, najrozsądniej przechowywać je w oddzielnym liniach.

Można też zastosować styl do wielu selektorów, jeśli mamy taką potrzebę. Wystarczy oddzielić selektory przecinkiem.

```
<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        h1, h2, h3 {
            color: #36C;
            font-weight: normal;
            letter-spacing: .4em;
            margin-bottom: 1em;
            text-transform: lowercase;
        }
    </style>
</head>
<body>
    <h1>Tytuł pierwszego poziomu</h1>
    <h2>Tytuł drugiego poziomu</h2>
    <h3>Tytuł trzeciego poziomu</h3>
</body>
</html>
```

Ta reguła zdefiniowania stylu będzie dotyczyć również elementów `h1`, `h2` i `h3`. Kolejność listy jest bez znaczenia. Wszystkie elementy w selektorze będą miały odpowiednie deklaracje zastosowane do nich. Można połączyć różne selektory *identyfikatorów*.

```
<!DOCTYPE html>
<html>
<head>
    <title>This is document title</title>
    <style>
        #content, #footer, #supplement {
            position: absolute;
            left: 510px;
            width: 200px;
        }
    </style>
</head>
<body>
    <article id="content">
        Tekst w content
    </article>
    <section id="supplement">
        Tekst w supplement
    </section>
    <footer id="footer">
        Tekst w footer
    </footer>
</body>
</html>
```

Reguły CSS można umieścić w dokumencie HTML za pomocą elementu `<style>`. Ten znacznik jest umieszczany wewnętrznie w elementach `<head>... </head>`. Reguły zdefiniowane przy użyciu tej składni zostaną zastosowane do wszystkich elementów dostępnych w dokumencie. Nie jest to jedyna metoda na osadzanie elementów CSS w dokumencie HTML.

Do zdefiniowania reguł stylu można użyć atrybutu `style` dowolnego elementu HTML. Reguły te będą stosowane tylko do tego elementu.

```
<!DOCTYPE html>
<html lang="pl">
<head>
</head>

<body>
<h1 style = "color:#36C;">
    This is inline CSS
</h1>
</body>
</html>
```

Element `<link>` może służyć do dołączania do dokumentu HTML zewnętrznego pliku arkusza stylów. Zewnętrzny arkusz stylów to oddzielnny plik tekstowy z **rozszerzeniem .css**. Definiowane są tam wszystkie reguły stylu w tym pliku tekstowym, a następnie można dołączyć ten plik do dowolnego dokumentu HTML za pomocą elementu `<link>`. Rozważmy osobny plik style.css:

```
/*Style.css - Tak się pisze komentarze w CSS*/
h1, h2, h3 {
    color: #36C;
    font-weight: normal;
    letter-spacing: .4em;
    margin-bottom: 1em;
    text-transform: lowercase;
}
```

Aby dołączyć go do naszego dokumentu html należy użyć następującego kodu:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <link href = "style.css" rel="stylesheet" />
    <title>Hello world!!!</title>
</head>

<body>
    <h1>Tytuł pierwszego poziomu</h1>
    <h2>Tytuł drugiego poziomu</h2>
    <h3>Tytuł trzeciego poziomu</h3>
    <h4>Tytuł czwartego poziomu</h4>
</body>
</html>
```

W przypadku używania różnych form definiowania stylów, należy pamiętać o następujących regułach:

- Każdy wbudowany arkusz stylów ma najwyższy priorytet. Spowoduje to zastąpienie dowolnej reguły zdefiniowanej w `<style>... </style>` tagu lub reguły zdefiniowane w dowolnym zewnętrznym pliku arkusza stylów.
- Dowolna reguła zdefiniowana w `<style>...</style>` zastąpią reguły zdefiniowane w dowolnym zewnętrznym pliku arkusza stylów.
- Każda reguła zdefiniowana w pliku arkusza stylów zewnętrznych ma najniższy priorytet, a reguły zdefiniowane w tym pliku będą stosowane tylko wtedy, gdy powyższe dwie reguły nie mają zastosowania.

Step 4

CSS obsługuje szereg pomiarów, w tym jednostki bezwzględne, takie jak całe, centymetry, punkty i tak dalej, a także względne miary, takie jak wartości procentowe i jednostki `em`. Te wartości są potrzebne przy jednoczesnym określeniu różnych pomiarów w regułach stylu. Przykłady użycia jednostek oraz miar przedstawia poniższa tabela.

Jednostki	Opis
%	Definiuje pomiar jako wartość procentową względem innej wartości, zazwyczaj otaczającego elementu.
cm	Definiuje pomiar w centymetrach.
em	Względny pomiar wysokości czcionki w spacjach em. Ponieważ jednostka em jest odpowiednikiem rozmiaru danej czcionki, jeśli przypiszemy każda jednostka "em" będzie 12pt; tak, 2em będzie 24pt.

ex	Ta wartość definiuje pomiar względem wysokości x czcionki. Wysokość x jest określana na podstawie wysokości litery x czcionki.
in	Definiuje pomiar w calach.
mm	Definiuje pomiar w milimetrach.
pc	Definiuje pomiar w picas. Pica odpowiada 12 punktom; w ten sposób, istnieje 6 picas na cal.
pt	Definiuje pomiar w punktach. Punkt jest zdefiniowany jako 1/72 cala.
px	Definiuje pomiar w pikselach ekranu.

Aby ustawić tła na różnych elementach HTML można skorzystać z następujących właściwości:

- Właściwość `background-color` służy do ustawiania koloru tła elementu.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style="background-color: yellow">Tekst na żółtym tle.</p>
</body>
</html>
```

- Właściwość `background-image` służy do ustawiania obrazu tła elementu.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style>
        body {
            background-image: url("back.jpeg");
            background-color: #cccccc;
        }
    </style>
</head>

<body>
    <p style="background-color: yellow">Tekst na żółtym tle.</p>
</body>
</html>
```

- Właściwość `background-repeat` służy do kontrolowania powtarzania obrazu w tle. Domyślnie właściwość `background-repeat` będzie miała wartość `repeat`.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style>
        body {
            background-image: url("back.jpeg");
            background-color: #cccccc;
            background-repeat: no-repeat;
        }
    </style>
</head>

<body>
    <p style="background-color: yellow">Tekst na żółtym tle.</p>
</body>
</html>
```

- Właściwość `background-position` służy do kontrolowania położenia obrazu w tle.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style>
        body {
            background-image: url("back.jpeg");
            background-position:100px;
        }
    </style>
</head>

<body>
    <p style="background-color: yellow">Tekst na żółtym tle.</p>
</body>
</html>
```

- Właściwość `background-attachment` służy do kontrolowania przewijania obrazu w tle.

- Właściwość `background` jest używana jako skrót do określenia wielu innych właściwości tła.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "background:url(back.jpeg) repeat fixed;">
        This paragraph has fixed repeated background image.
    </p>
</body>
</html>
```

Step 5

Aby ustawić czcionki dostępne w elemencie HTML można skorzystać z następujących właściwości:

- Właściwość `font-family` służy do zmiany powierzchni czcionki.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "font-family:georgia,garamond,serif;">
        This text is rendered in either georgia, garamond, or the
        default serif font depending on which font you have at your system.
    </p>
</body>
</html>

```

- Właściwość `font-style` jest używana do tworzenia między innymi kursywy czcionki. Możliwe wartości są `normal`, `italic` i `oblique`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "font-style:italic;">
        This text will be rendered in italic style
    </p>
    <p style = "font-style:normal;">
        This text will be rendered in normal style
    </p>
    <p style = "font-style:oblique;">
        This text will be rendered in oblique style
    </p>
</body>
</html>

```

- Właściwość `font-variant` służy do tworzenia efektu kapitaliki. Możliwe wartości to `small-caps` oraz `normal`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "font-variant:small-caps;">
        This text will be rendered in small-caps style
    </p>
    <p style = "font-variant:normal;">
        This text will be rendered in normal style
    </p>
</body>
</html>

```

- Właściwość `font-weight` służy do zwiększania lub zmniejszania pogrubienia lub światła czcionki. Możliwymi wartościami są: `normal`, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, `900`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "font-weight:bold;">
        This font is bold.
    </p>

    <p style = "font-weight:bolder;">
        This font is bolder.
    </p>

    <p style = "font-weight:500;">
        This font is 500 weight.
    </p>
</body>
</html>

```

- Właściwość `font-size` służy do zwiększenia lub zmniejszenia rozmiaru czcionki. Możliwymi wartościami są `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`, `smaller`, `larger`, `rozmiar w pikselach` lub `%`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "font-size:20px;">
        This font size is 20 pixels
    </p>

    <p style = "font-size:small;">
        This font size is small
    </p>

    <p style = "font-size:large;">
        This font size is large
    </p>
</body>
</html>

```

- Właściwość `font` jest używana jako skrót do określenia wielu innych właściwości czcionki.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "font:italic small-caps bold 15px georgia;">
        Applying all the properties on the text at once.
    </p>
</body>
</html>

```

Aby manipulować tekstem przy użyciu właściwości CSS, można skorzystać z następujących właściwości:

- Właściwość `color` służy do ustawiania koloru tekstu.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "color:red;">
        This text will be written in red.
    </p>
</body>
</html>

```

- Właściwość `direction` służy do ustawiania kierunku tekstu.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "direction:rtl;">
        This text will be rendered from right to left
    </p>
</body>
</html>

```

- Właściwość `letter-spacing` służy do dodawania lub odejmowania odstępu między literami, które tworzą wyraz.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "letter-spacing:5px;">
        This text is having space between letters.
    </p>
</body>
</html>

```

- Właściwość `word-spacing` służy do dodawania lub odejmowania odstępu między wyrazami zdania.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "word-spacing:5px;">
        This text is having space between words.
    </p>
</body>
</html>

```

- Właściwość `text-indent` służy do wcięcie tekstu akapitu.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "text-indent:1cm;">
        This text will have first line indented by 1cm and this line will remain at
        its actual position this is done by CSS text-indent property.
    </p>
</body>
</html>

```

- Właściwość `text-align` służy do wyrównywania tekstu dokumentu. Możliwe wartości to `left, right, center, justify`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "text-align:right;">
        This will be right aligned.
    </p>

    <p style = "text-align:center;">
        This will be center aligned.
    </p>

    <p style = "text-align:left;">
        This will be left aligned.
    </p>
</body>
</html>

```

- Właściwość `text-decoration` służy do podkreślenia lub przekreślenia tekstu. Możliwe wartości to `none, underline, overline, line-through, blink`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "text-decoration:underline;">
        This will be underlined
    </p>

    <p style = "text-decoration:line-through;">
        This will be striked through.
    </p>

    <p style = "text-decoration:overline;">
        This will have a over line.
    </p>

    <p style = "text-decoration:blink;">
        This text will have blinking effect
    </p>
</body>
</html>

```

- Właściwość `text-transform` służy do wielkich liter lub konwertowania tekstu na wielkie lub małe litery. Możliwe wartości to `none, capitalize, uppercase, lowercase`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "text-transform:capitalize;">
        This will be capitalized
    </p>

    <p style = "text-transform:uppercase;">
        This will be in uppercase
    </p>

    <p style = "text-transform:lowercase;">
        This will be in lowercase
    </p>
</body>
</html>

```

- Właściwość `white-space` służy do kontrolowania przepływu i formatowania tekstu. Możliwe wartości to `normal`, `pre`, `nowrap`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "white-space:pre;">
        This text has a line break and the white-space pre setting
        tells the browser to honor it just like the HTML pre tag.
    </p>
</body>
</html>

```

- Właściwość `text-shadow` służy do ustawiania cienia tekstu wokół tekstu.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "text-shadow:4px 4px 8px blue;">
        If your browser supports the CSS text-shadow property,
        this text will have a blue shadow.
    </p>
</body>
</html>

```

Step 6

Obrazy odgrywają ważną rolę na każdej stronie internetowej. Chociaż nie zaleca się dodawania wielu obrazów, ale nadal ważne jest, aby używać dobrych obrazów, gdziekolwiek jest to wymagane. CSS odgrywa dobrą rolę do kontrolowania wyświetlania obrazu. Następujące właściwości obrazu można ustawić za pomocą css.

- Właściwość `border` służy do ustawiania szerokości obramowania obrazu.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <img style = "border:0px;" src = "train1.jpg" />
    <br />
    <img style = "border:3px dashed red;" src = "train1.jpg" />
</body>
</html>

```

- Właściwość `height` służy do ustawiania wysokości obrazu.
- Właściwość `width` służy do ustawiania szerokości obrazu.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <img style = "border:1px solid red; width:250px; height: 125px" src = "train1.jpg" />
</body>
</html>
```

W przypadku hiperłącz (linków) można ustawić następujące właściwości:

- `:link` oznacza nieodwiedzane hiperłącza.
- `:visited` oznacza odwiedzone hiperłącza.
- `:hover` oznacza element, na który znajduje się wskaźnik myszy użytkownika, na który znajduje się wskaźnik myszy.
- `:active` oznacza element, na którym użytkownik aktualnie kliką.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style type = "text/css">
        a:link {color:#000000}
        a:visited {color: #006600}
        a:hover {color: #FFCC00}
        a:active {color: #FF00CC}
    </style>
</head>

<body>
    <a href="https://google.pl">Wyszukaj w google!!!</a>
</body>
</html>
```

Należy pamiętać, że `a:hover` musi pochodzić po `a:link` oraz `a:active` musi przyjść po `a:hover`. Inaczej działanie tych stylów może być nieprawidłowe.

Tabele są nieroziłącznym elementem stron internetowych, ponieważ bardzo często łączy się je za pomocą formularzy. Za pomocą CSS można ustawić następujące właściwości tabeli:

- `border-collapse` określa, czy przeglądarka powinna kontrolować wygląd przyległych obramowań, które się ze sobą dotykają, czy też każda komórka powinna zachować swój styl.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style type = "text/css">
        table.one {border-collapse:collapse;}
        table.two {border-collapse:separate;}

        td.a {
            border-style:dotted;
            border-width:3px;
            border-color:#000000;
            padding: 10px;
        }
        td.b {
            border-style:solid;
            border-width:3px;
            border-color:#333333;
            padding:10px;
        }
    </style>
</head>

<body>
    <table class = "one">
        <caption>Collapse Border Example</caption>
        <tr><td class = "a"> Cell A Collapse Example</td></tr>
        <tr><td class = "b"> Cell B Collapse Example</td></tr>
    </table>
    <br />

    <table class = "two">
        <caption>Separate Border Example</caption>
        <tr><td class = "a"> Cell A Separate Example</td></tr>
        <tr><td class = "b"> Cell B Separate Example</td></tr>
    </table>
</body>
</html>

```

- Odstępy `border-spacing` określają szerokość, która powinna być wyświetlana między komórkami tabeli.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style type = "text/css">
        table.one {
            border-collapse:separate;
            width:400px;
            border-spacing:10px;
        }
        table.two {
            border-collapse:separate;
            width:400px;
            border-spacing:10px 50px;
        }

        td.a {
            border-style:dotted;
            border-width:3px;
            border-color:#000000;
            padding: 10px;
        }
        td.b {
            border-style:solid;
            border-width:3px;
            border-color:#333333;
            padding:10px;
        }
    </style>
</head>

<body>
    <table class = "one">
        <caption>Collapse Border Example</caption>
        <tr><td class = "a"> Cell A Collapse Example</td></tr>
        <tr><td class = "b"> Cell B Collapse Example</td></tr>
    </table>
    <br />

    <table class = "two">
        <caption>Separate Border Example</caption>
        <tr><td class = "a"> Cell A Separate Example</td></tr>
        <tr><td class = "b"> Cell B Separate Example</td></tr>
    </table>
</body>
</html>

```

- Podpisy `caption-side` są prezentowane w elemencie `<caption>`. Domyślnie są one renderowane nad tabelą w dokumencie. Właściwości *po stronie podpisu* służą do kontrolowania rozmieszczenia podpisu tabeli.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style type = "text/css">
        caption.top {caption-side:top}
        caption.bottom {caption-side:bottom}
    </style>
</head>

<body>
    <table style = "width:400px; border:1px solid black;">
        <caption class = "top">
            This caption will appear at the top
        </caption>
        <tr><td> Cell A</td></tr>
        <tr><td> Cell B</td></tr>
    </table>
    <br />

    <table style = "width:400px; border:1px solid black;">
        <caption class = "bottom">
            This caption will appear at the bottom
        </caption>
        <tr><td> Cell A</td></tr>
        <tr><td> Cell B</td></tr>
    </table>
    <br />
</body>
</html>

```

- `empty-cells` określają, czy obramowanie powinno być wyświetlane, jeśli komórka jest pusta.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style type = "text/css">
        table.empty {
            width:350px;
            border-collapse:separate;
            empty-cells:hide;
        }
        td.empty {
            padding:5px;
            border-style:solid;
            border-width:1px;
            border-color:#999999;
        }
    </style>
</head>

<body>
    <table class = "empty">
        <tr>
            <th></th>
            <th>Title one</th>
            <th>Title two</th>
        </tr>

        <tr>
            <th>Row Title</th>
            <td class = "empty">value</td>
            <td class = "empty">value</td>
        </tr>

        <tr>
            <th>Row Title</th>
            <td class = "empty">value</td>
            <td class = "empty"></td>
        </tr>
    </table>
</body>
</html>

```

Właściwości `border` umożliwiają określenie wyglądu obramowania pola reprezentującego element. Istnieją trzy właściwości obramowania, które można zmienić:

- Kolor `border-color` określa kolor obramowania. Umożliwia zmianę koloru obramowania otaczającego element. Można indywidualnie zmieniać kolor dolnej, lewej, górnej i prawej strony obramowania elementu za pomocą następujących właściwości
 - `border-bottom-color` zmienia kolor dolnej krawędzi.
 - `border-top-color` kolor górnej obramowania.
 - `border-left-color` zmienia kolor lewej krawędzi.
 - `border-right-color` zmienia kolor prawej krawędzi.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
    <style type = "text/css">
        p.example1 {
            border:1px solid;
            border-bottom-color:#009900; /* Green */
            border-top-color:#FF0000;     /* Red */
            border-left-color:#330000;   /* Black */
            border-right-color:#0000CC;  /* Blue */
        }
        p.example2 {
            border:1px solid;
            border-color:#009900;       /* Green */
        }
    </style>
</head>

<body>
    <p class = "example1">
        This example is showing all borders in different colors.
    </p>

    <p class = "example2">
        This example is showing all borders in green color only.
    </p>
</body>
</html>
```

- Styl `border-style` określa, czy obramowanie powinno być trwałe, linia przerywana, podwójna linia, czy jedna z innych możliwych wartości. Dostępne są następujące style obramowania:

Można indywidualnie zmieniać styl dolnej, lewej, górnej i prawej krawędzi elementu, korzystając z następujących właściwości

- `none` – Brak obramowania. (Odpowiednik szerokości obramowania:0)
- `solid` – Obramowanie jest pojedynczą linią ciągłą.
- `dotted` – Obramowanie to seria kropek.
- `dashed` – Obramowanie to seria krótkich linii.
- `double` – Obramowanie to dwie linie stałe.
- `groove` – Obramowanie wygląda tak, jakby było wyryte na stronie.
- `ridge` – Obramowanie wygląda jak grzbiet
- `inset` – Obramowanie sprawia, że pole wygląda tak, jakby było osadzone na stronie.
- `outset` – Obramowanie sprawia, że pudełko wygląda tak, jakby wychodziło z płotna.
- `hidden` – Tak samo jak żadna, z wyjątkiem rozwiązywania konfliktów granicznych w odniesieniu do elementów tabeli.
- `border-bottom-style` zmienia styl dolnej krawędzi.
- `border-top-style` zmienia styl górnej obramowania.
- `border-left-style` zmienia styl lewej granicy.
- `border-right-style` zmienia styl prawej granicy.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "border-width:4px; border-style:none;">
        This is a border with none width.
    </p>

    <p style = "border-width:4px; border-style:solid;">
        This is a solid border.
    </p>

    <p style = "border-width:4px; border-style:dashed;">
        This is a dashed border.
    </p>

    <p style = "border-width:4px; border-style:double;">
        This is a double border.
    </p>

    <p style = "border-width:4px; border-style:groove;">
        This is a groove border.
    </p>

    <p style = "border-width:4px; border-style:ridge">
        This is a ridge border.
    </p>

    <p style = "border-width:4px; border-style:inset;">
        This is a inset border.
    </p>

    <p style = "border-width:4px; border-style:outset;">
        This is a outset border.
    </p>

    <p style = "border-width:4px; border-style:hidden;">
        This is a hidden border.
    </p>

    <p style = "border-width:4px;
        border-top-style:solid;
        border-bottom-style:dashed;
        border-left-style:groove;
        border-right-style:double;">
        This is a a border with four different styles.
    </p>
</body>
</html>

```

- **border-width** określa szerokość obramowania. Można indywidualnie zmieniać szerokość dolnej, górnej, lewej i prawej krawędzi elementu, korzystając z następujących właściwości :

- **border-bottom-width** zmienia szerokość dolnej granicy.
- **border-top-width** zmienia szerokość górnej obramowania.
- **border-left-width** zmienia szerokość lewej krawędzi.
- **border-right-width** szerokość prawej krawędzi.

Właściwość **margin** definiuje spację wokół elementu HTML. Istnieje możliwość użycia wartości ujemnych do nakładania się zawartości.

Wartości właściwości margin nie są dziedziczone przez elementy podzielone. Należy pamiętać, że sąsiednie marginesy pionowe (górnego i dolnego marginesów) zwijają się na siebie, tak aby odległość między blokami nie była sumą marginesów, ale tylko większą z dwóch marginesów lub taką samą wielkością jak jeden margines, jeśli oba są równe.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "border-width:4px; border-style:solid;">
        This is a solid border whose width is 4px.
    </p>

    <p style = "border-width:4pt; border-style:solid;">
        This is a solid border whose width is 4pt.
    </p>

    <p style = "border-width:thin; border-style:solid;">
        This is a solid border whose width is thin.
    </p>

    <p style = "border-width:medium; border-style:solid;">
        This is a solid border whose width is medium;
    </p>

    <p style = "border-width:thick; border-style:solid;">
        This is a solid border whose width is thick.
    </p>

    <p style = "border-bottom-width:4px; border-top-width:10px;
        border-left-width: 2px; border-right-width:15px; border-style:solid;">
        This is a border with four different width.
    </p>
</body>
</html>

```

Właściwość `margin` definiuje spację wokół elementu HTML. Istnieje możliwość użycia wartości ujemnych do nakładania się zawartości. Wartości właściwości `margin` nie są dziedziczone przez elementy podrzędne. Należy pamiętać, że sąsiednie marginesy pionowe (górnego i dolnego marginesów) zwijają się na siebie, tak aby odległość między blokami nie była sumą marginesów, ale tylko większą z dwóch marginesów lub taką samą wielkością jak jeden margines, jeśli oba są równe. Aby ustawić margines elementu można skorzystać z następujących właściwości:

- `margin` określa skróconą właściwość do ustawiania właściwości marginesu w jednej deklaracji.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "margin: 15px; border:1px solid black;">
        all four margins will be 15px
    </p>

    <p style = "margin:10px 2%; border:1px solid black;">
        top and bottom margin will be 10px, left and right margin will be 2%
        of the total width of the document.
    </p>

    <p style = "margin: 10px 2% -10px; border:1px solid black;">
        top margin will be 10px, left and right margin will be 2% of the
        total width of the document, bottom margin will be -10px
    </p>

    <p style = "margin: 10px 2% -10px auto; border:1px solid black;">
        top margin will be 10px, right margin will be 2% of the total
        width of the document, bottom margin will be -10px, left margin
        will be set by the browser
    </p>
</body>
</html>

```

- `margin-bottom` określa dolny margines elementu.
- `margin-top` określa górny margines elementu.
- `margin-left` określa lewy margines elementu.
- `margin-right` określa prawy margines elementu

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "margin-bottom: 15px; border:1px solid black;">
        This is a paragraph with a specified bottom margin
    </p>

    <p style = "margin-bottom: 5%; border:1px solid black;">
        This is another paragraph with a specified bottom margin in percent
    </p>
    <p style = "margin-top: 15px; border:1px solid black;">
        This is a paragraph with a specified top margin
    </p>

    <p style = "margin-top: 5%; border:1px solid black;">
        This is another paragraph with a specified top margin in percent
    </p>
    <p style = "margin-left: 15px; border:1px solid black;">
        This is a paragraph with a specified left margin
    </p>

    <p style = "margin-left: 5%; border:1px solid black;">
        This is another paragraph with a specified top margin in percent
    </p>
    <p style = "margin-right: 15px; border:1px solid black;">
        This is a paragraph with a specified right margin
    </p>
    <p style = "margin-right: 5%; border:1px solid black;">
        This is another paragraph with a specified right margin in percent
    </p>
</body>
</html>

```

Listy są bardzo pomocne w przekazywaniu zestawu ponumerowanych lub punktorów. W przypadku kontrolowania list za pomocą CSS mamy następujące właściwości:

- `list-style-type` umożliwia sterowanie kształtem lub wyglądem znacznika. Dla nieuporządkowanych list mamy następujące wartości: `none`, `disc`, `circle` oraz `square`. W przypadku uporządkowanych list do dyspozycji mamy: `decimal`, `decimal-leading-zero`, `lower-alpha`, `upper-alpha`, `lower-roman`, `upper-roman`, `lower-greek`, `lower-latin`, `upper-latin`, `hebrew`, `armenian`, `georgian`, `cjk-ideographic`, `hiragana`, `katakana`, `hiragana-iroha`, `katakana-iroha`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <ul style = "list-style-type:circle;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ul>

    <ul style = "list-style-type:square;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ul>

    <ol style = "list-style-type:decimal;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ol>

    <ol style = "list-style-type:lower-alpha;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ol>

    <ol style = "list-style-type:lower-roman;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ol>
</body>
</html>

```

- `list-style-position` określa, czy dlugi punkt zawijany do drugiego wiersza powinien być wyrównany z pierwszym wierszem, czy zaczynał się pod początkiem znacznika. Dostępne wartości to `none`, `inside` oraz `outside`.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <ul style = "list-style-type:circle; list-style-position:outside;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ul>

    <ul style = "list-style-type:square;list-style-position:inside;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ul>

    <ol style = "list-style-type:decimal;list-style-position:outside;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ol>

    <ol style = "list-style-type:lower-alpha;list-style-position:inside;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ol>
</body>
</html>
```

- `list-style-image` określa obraz znacznika, a nie punktora lub liczby.
- `list-style` służy jako skrót dla poprzednich właściwości.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <ul style = "list-style: inside square;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ul>

    <ol style = "list-style: outside upper-alpha;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ol>
</body>
</html>
```

- `marker-offset` określa odległość między znacznikiem a tekstem na liście.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <ul style = "list-style: inside square; marker-offset:2em;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ul>

    <ol style = "list-style: outside upper-alpha; marker-offset:2cm;">
        <li>Maths</li>
        <li>Social Science</li>
        <li>Physics</li>
    </ol>
</body>
</html>
```

Step 8

Właściwość `padding` pozwala określić, ile miejsca ma być wyświetlane między zawartością elementu a jego obramowaniem –

Wartość tego atrybutu powinna być długością, wartością procentową lub słowem `inherit`. Jeśli wartość jest dziedziczona, będzie miała taką samą dopełnienie jak jego element nadrzedny. Jeśli używana jest wartość procentowa, wartość procentowa jest polem zawierającym. Następujące właściwości CSS mogą służyć do kontrolowania list. Można również ustawić różne wartości dopełnienia po każdej stronie pola, korzystając z następujących właściwości –

- `padding-bottom` określa dolną dopełnienie elementu.
- `padding-top` określa górnej dopełnienie elementu.
- `padding-left` określa lewą dopełnienie elementu.
- `padding-right` określa prawe dopełnienie elementu.
- `padding` służy jako skrót dla poprzednich właściwości.

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "padding-bottom: 15px; border:1px solid black;">
        This is a paragraph with a specified bottom padding
    </p>

    <p style = "padding-bottom: 5%; border:1px solid black;">
        This is another paragraph with a specified bottom padding in percent
    </p>
    <p style = "padding-top: 15px; border:1px solid black;">
        This is a paragraph with a specified top padding
    </p>

    <p style = "padding-top: 5%; border:1px solid black;">
        This is another paragraph with a specified top padding in percent
    </p>
    <p style = "padding-left: 15px; border:1px solid black;">
        This is a paragraph with a specified left padding
    </p>

    <p style = "padding-left: 15%; border:1px solid black;">
        This is another paragraph with a specified left padding in percent
    </p>
    <p style = "padding-right: 15px; border:1px solid black;">
        This is a paragraph with a specified right padding
    </p>

    <p style = "padding-right: 5%; border:1px solid black;">
        This is another paragraph with a specified right padding in percent
    </p>
</body>
</html>
```

Możemy kontrolować wymiary za pomocą następujących właściwości:

- Właściwość `height` służy do ustawiania wysokości pola.
- Właściwość `width` służy do ustawiania szerokości pola.
- Właściwość `line-height` służy do ustawiania wysokości wiersza tekstu.
- Właściwość `max-height` służy do ustawiania maksymalnej wysokości, jaką może być pole.
- Właściwość `min-height` służy do ustawiania minimalnej wysokości, jaką może być pole.
- Właściwość `max-width` służy do ustawiania maksymalnej szerokości, jaką może być pole.
- Właściwość `min-width` służy do ustawiania minimalnej szerokości, jaką może być pole.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <title>Hello world!!!</title>
</head>

<body>
    <p style = "width:400px; height:100px; border:1px solid red; padding:5px; margin:10px;">
        This paragraph is 400pixels wide and 100 pixels high
    </p>
    <p style = "width:400px; height:100px; border:1px solid red; padding:5px; margin:10px; line-height:30px;">
        This paragraph is 400pixels wide and 100 pixels high and here line height is 30pixels.
        This paragraph is 400 pixels wide and 100 pixels high and here line height is 30pixels.
    </p>
    <p style = "width:400px; min-height:200px; border:1px solid red; padding:5px; margin:10px;">
        This paragraph is 400px wide and min height is 200px
        This paragraph is 400px wide and min height is 200px
        This paragraph is 400px wide and min height is 200px
        This paragraph is 400px wide and min height is 200px
    </p>
    <p style = "max-width:100px; height:200px; border:1px solid red; padding:5px; margin:10px;">
        This paragraph is 200px high and max width is 100px
        This paragraph is 200px high and max width is 100px
        This paragraph is 200px high and max width is 100px
        This paragraph is 200px high and max width is 100px
        This paragraph is 200px high and max width is 100px
    </p>
</body>
</html>

```

Może się wydawać, że zawartość elementu może być większa niż ilość przydzielonego mu miejsca. Na przykład, biorąc pod uwagę szerokość i wysokość właściwości nie pozwalają wystarczająco dużo miejsca, aby pomieścić zawartość elementu.

CSS zapewnia właściwość o nazwie `overflow`, która informuje przeglądarkę, co zrobić, jeśli zawartość pola jest większa niż samo pole. Ta właściwość może przyjąć jedną z następujących wartości: `visible`, `hidden`, `scroll` oraz `auto`.

```

<!DOCTYPE html>
<html lang="pl">
<head>
    <style type = "text/css">
        .scroll {
            display:block;
            border: 1px solid red;
            padding:5px;
            margin-top:5px;
            width:300px;
            height:50px;
            overflow:scroll;
        }
        .auto {
            display:block;
            border: 1px solid red;
            padding:5px;
            margin-top:5px;
            width:300px;
            height:50px;
            overflow:auto;
        }
    </style>
</head>

<body>
<p>Example of scroll value:</p>
<div class = "scroll">
    I am going to keep lot of content here just to show you how
    scrollbars works if there is an overflow in an element box.
    This provides your horizontal as well as vertical scrollbars.
</div>
<br />

<p>Example of auto value:</p>

<div class = "auto">
    I am going to keep lot of content here just to show you how
    scrollbars works if there is an overflow in an element box.
    This provides your horizontal as well as vertical scrollbars.
</div>
</body>
</html>

```

Responsywna konstrukcja strony internetowej zapewnia optymalne wrażenia, łatwy odczyt i łatwą nawigację przy minimalnej zmiany rozmiaru na różnych urządzeniach, takich jak komputery stacjonarne, telefony komórkowe i karty). Przykładowa strona:

```
<html>
<head>
    <style>
        body {
            font: 600 14px/24px "Open Sans",
            "HelveticaNeue-Light",
            "Helvetica Neue Light",
            "Helvetica Neue",
            Helvetica, Arial,
            "Lucida Grande",
            Sans-Serif;
        }
        h1 {
            color: #9799a7;
            font-size: 14px;
            font-weight: bold;
            margin-bottom: 6px;
        }
        .container:before, .container:after {
            content: "";
            display: table;
        }
        .container:after {
            clear: both;
        }
        .container {
            background: #eaeaed;
            margin-bottom: 24px;
            *zoom: 1;
        }
        .container-75 {
            width: 75%;
        }
        .container-50 {
            margin-bottom: 0;
            width: 50%;
        }
        .container, section, aside {
            border-radius: 6px;
        }
        section, aside {
            background: #2db34a;
            color: #fff;
            margin: 1.858736059%;
            padding: 20px 0;
            text-align: center;
        }
        section {
            float: left;
            width: 63.197026%;
        }
        aside {
            float: right;
            width: 29.3680297%;
        }
    </style>
</head>

<body>

<h1>100% Wide Container</h1>

<div class = "container">
    <section>Section</section>
    <aside>Aside</aside>
</div>

<h1>75% Wide Container</h1>

<div class = "container container-75">
    <section>Section</section>
    <aside>Aside</aside>
</div>

<h1>50% Wide Container</h1>

<div class = "container container-50">
    <section>Section</section>
    <aside>Aside</aside>
</div>

</body>
</html>
```

Bootstrap (<https://getbootstrap.com/>) jest najbardziej popularną platformą do projektowania stron internetowych opartą na skrypcie HTML, CSS i Java i pomaga projektować strony internetowe w sposób responsywny dla wszystkich urządzeń. Przykładowa strona z użyciem bootstrap:

```
<html>
<head>
    <meta charset = "utf-8">
    <meta name = "viewport" content = "width=device-width, initial-scale = 1">
    <link rel = "stylesheet"
        href = "http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
    <style>
        body {
            color:green;
        }
    </style>
</head>

<body>

<div class = "container">

    <div class = "jumbotron">
        <h1>Tutorials point</h1>
        <p>
            Tutorials Point originated from the idea that there exists a class
            of readers who respond better to online content and prefer to learn
            new skills at their own pace from the comforts of their drawing rooms.
        </p>
    </div>

    <div class = "row">
        <div class = "col-md-4">
            <h2>Android</h2>
            <p>
                Android is an open source and Linux-based operating system for mobile
                devices such as smartphones and tablet computers. Android was developed
                by the Open Handset Alliance, led by Google, and other companies.
            </p>
        </div>

        <div class = "col-md-4">
            <h2>CSS</h2>
            <p>
                Cascading Style Sheets, fondly referred to as CSS, is a simple design
                language intended to simplify the process of making web pages presentable.
            </p>
        </div>

        <div class = "col-md-4">
            <h2>Java</h2>
            <p>
                Java is a high-level programming language originally developed by Sun
                Microsystems and released in 1995. Java runs on a variety of platforms,
                such as Windows, Mac OS, and the various versions of UNIX. This tutorial
                gives a complete understanding of Java.
            </p>
        </div>
    </div>
</div>

</body>
</html>
```