

Bazy Danych

Ćwiczenie 12: dostęp do danych zgromadzonych w bazie MySQL z poziomu przeglądarki internetowej z użyciem PHP

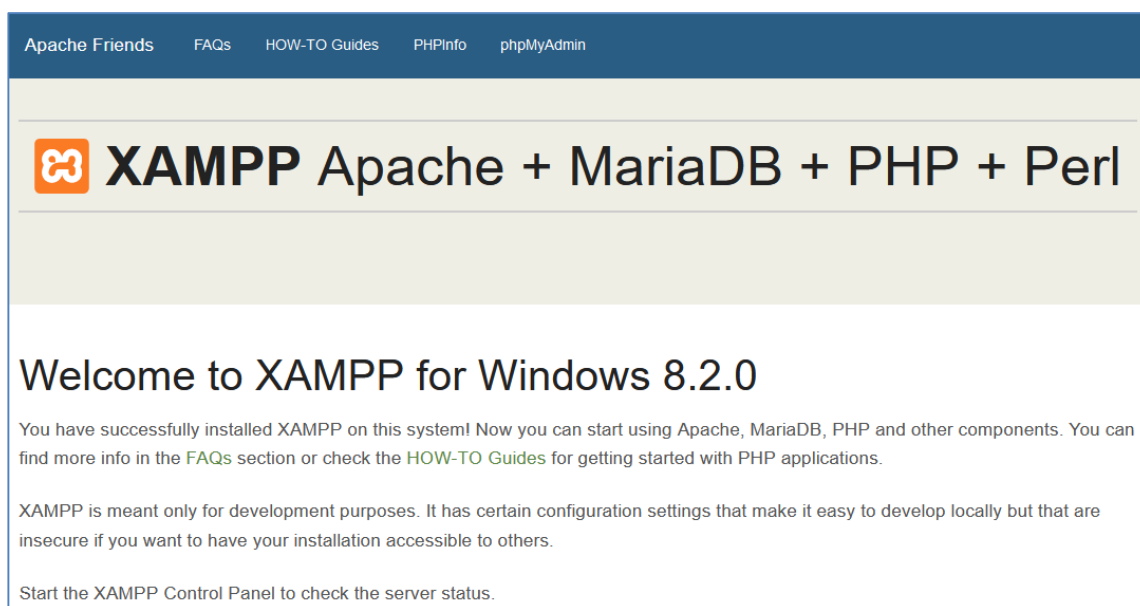
opracował: dr inż. Artur Gramacki (a.gramacki@issi.uz.zgora.pl)

1. Uwagi wstępne


Aby poprawnie wykonać ćwiczenie należy upewnić się, czy poprawnie działają:

- serwer MySQL,
- serwer WWW Apache,
- interpreter języka PHP.

Po uruchomieniu panelu kontrolnego XAMPP (patrz ćwiczenie 1) należy załadować do przeglądarki stronę główną, czyli:



Następnie należy wybrać z menu pozycję *PHPInfo*, co powinno spowodować wyświetlenie strony z różnymi szczegółowymi informacjami na temat zainstalowanego interpretera PHP, bazy MySQL i wielu, wielu innych działających modułów. Poprawnie uruchomienie się programu *phpMyAdmin* świadczy o tym, że baza MySQL została poprawnie uruchomiona i działa poprawnie.

PHP Version 8.2.4		
System	Windows NT DELL-ARTUR 10.0 build 19044 (Windows 10) AMD64	
Build Date	Mar 14 2023 17:50:26	
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]	
Compiler	Visual C++ 2019	
Architecture	x64	
Configure Command	cscript /nologo /e:javascript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=.\\..\\..\\instantclient\\sdk,shared" "--with-oci8-19=.\\..\\..\\instantclient\\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgsql"	
Server API	Apache 2.0 Handler	
Virtual Directory Support	enabled	
Configuration File (php.ini) Path	no value	
Loaded Configuration File	C:\Programy\xampp_8.2.4\php\php.ini	

2. Utworzyć w katalogu *htdocs* (podkatalog w katalogu głównym XAMPP-a) dowolny podkatalog. W tym katalogu będziemy przechowywali wszystkie własne *skrypty php* (oraz inne wymagane pliki, jak np. pliki z kodami *JavaScript*, grafikę, we. inne). Dostęp do tego katalogu w poziomie przeglądarki WWW będzie możliwy po wpisaniu adresu:

<http://hostname:port/podkatalog-jaki-utworzyles-w-katalogu-htdocs>

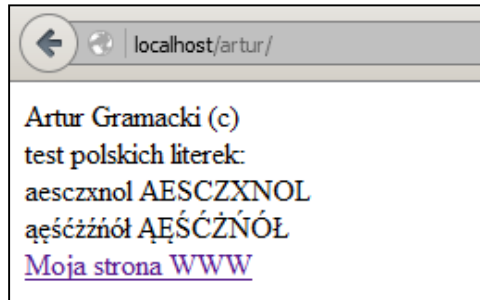
Nie powinniśmy naszych programów w PHP przechowywać bezpośrednio w katalogu *htdocs* (choć technicznie jest to w pełni możliwe), aby niepotrzebnie go nie zaśmiecać.

3. Utworzyć a następnie wgrać do katalogu pierwszy plik o nazwie *index.html*. Wyświetla on w przeglądarce statyczne dane, więc nie ma na razie żadnego połączenia do bazy MySQL. Poprawne wyświetlenie zawartości tego pliku potwierdza nam, że środowisko serwera WWW działa poprawnie, plik został wgrany do właściwego katalogu i w przeglądarce wpisaliśmy poprawny adres. Poprawne wyświetlenie wszystkich polskich liter potwierdza, że nie ma problemów z kodowaniem znaków.

```

index.html
1  <!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf8">
5  <meta name="Keywords" content="słowo 1, słowo2">
6  <meta name="Description" content="Stronka testowa">
7  <meta name="Author" content="Artur Gramacki">
8  <meta name="Copyright" content="(c) by Artur Gramacki">
9  <meta http-equiv="Reply-to" content="a.gramacki@issi.uz.zgora.pl">
10 <meta name="Language" content="pl">
11 <meta name="Distribution" content="Global">
12 <meta name="Robots" content="all">
13 <meta name="Classification" content="Education">
14
15 <title>(c) by Artur Gramacki</title>
16 </head>
17 <body>
18   Artur Gramacki (c) <br>
19   test polskich literek: <br>
20   aesczxnol AESCZYNOL <br>
21   ąęśćźńół AĘŚĆŹŃÓŁ <br>
22   <a href="http://www.uz.zgora.pl/~agramack">Moja strona WWW</a>
23 </body>
24 </html>
25

```



Do tworzenia i edycji plików źródłowych sugerujemy używać edytora, który potrafi podświetlać składnię, co bardzo ułatwia tworzenie programów i na starcie eliminuje wiele prostych błędów. Bardzo dobrym edytorem z tej grupy jest **Notepad++** lub **Sublime Text**.

Powyższy plik zapisano w stronie kodowej UTF8 i w zasadzie nie ma powodu, aby nasze pliki kodować inaczej. Ta strona kodowa jest najbardziej uniwersalna.

4. Utworzyć dwa skrypty php, które zademonstrują różnice pomiędzy używaniem w języku php apostrofów oraz cudzysłówów. Zaobserwuj dokładnie działanie obu plików i zrozum w każdym przypadku dlaczego wyniki, jakie widzimy na ekranie są takie a nie inne.

```
<?php
    $test = "jakiś tekst <br>";
    echo "$test";
    // Na ekranie wyświetli się: jakiś tekst
?>
```

oraz

```
<?php
    $test = "jakiś tekst";
    echo '$test';
    // na ekranie wyświetli się: $test
?>
```

5. Utworzyć dwa skrypty php, demonstrujące zasadę pracy z tzw. *formularzami html* (ang. *html forms*). Jeden z nich wykorzystuje metodę *GET* a drugi *POST* do przesyłania danych użytkownika od przeglądarki do serwera WWW oraz w drugą stronę.

```
<html>
<head>
    <title>(c) by Artur Gramacki</title>
</head>
<body>
<H3> Przykład formularza </H3>
<?php
if (isset($_GET['moj_tekst'])) { // gdy cos wpisano
    print "Wpisano: $_GET[moj_tekst] <br>";
    print '<a href = "form_get.php">Powrót do formularza</a>';
}
}
```

```

else { // gdy nic nie wpisano
    print '<form action="form_get.php" method="get">';
    print '<input type="text" name="moj_tekst">';
    print '<input type="submit" value="Wyslij">';
    print '</form>';
}
?>
</body>
</html>

```

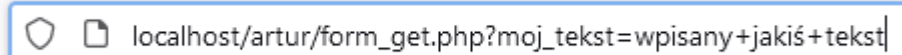
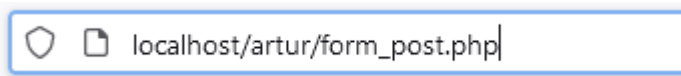
oraz

```

<html>
<head>
    <title>(c) by Artur Gramacki</title>
</head>
<body>
<H3>Przykład formularza </H3>
<?php
if (isset($_POST['moj_tekst'])) { // gdy cos wpisano
    print "Wpisano: $_POST[moj_tekst] <br>";
    print '<a href = "form_post.php">Powrót do formularza</a>';
}
else { // gdy nic nie wpisano
    print '<form action="form_post.php" method="post">';
    print '<input type="text" name="moj_tekst">';
    print '<input type="submit" value="Wyslij">';
    print '</form>';
}
?>
</body>
</html>

```

Zaobserwuj pasek adresowy w przeglądarce w czasie działania obu skryptów. Jak mógłbyś opisać/skomentować/wyjaśnić zaobserwowane różnice?

- Utwórz w bazie MySQL poniższą tabelę. Bedzie ona podstawą dalszych ćwiczeń.

```

# przełącz się na właściwą bazę danych poleceniem: use moja-baza-danych

DROP TABLE IF EXISTS prac;

CREATE TABLE prac (
id          INT UNSIGNED NOT NULL PRIMARY KEY,
imie       CHAR(20) NOT NULL,
nazwisko   CHAR(30) NOT NULL,
zarobki    FLOAT(6,2) NOT NULL,
data_ur    DATE

```

```

) ENGINE=InnoDB DEFAULT CHARACTER SET = utf8 COLLATE = utf8_bin;

INSERT INTO prac VALUES
(1, 'Artur', 'Gramacki', 1000, '1999-01-01'),
(2, 'Gal', 'Anonim', 2000, '1999-01-01'),
(3, 'ąęśćźźńół', 'ĄĘŚĆŹŹŃÓŁ', 9999, '2006-01-01');

```

Zwróć uwagę, że jawnie podaliśmy w poleceniu CREATE TABLE stronę kodową (UTF8). Plik musi więc być zapisany w tej samej stronie kodowej, aby polskie znaki zostały poprawnie zakodowane w bazie danych. Polecenie, które zostało użyte do utworzenia tabeli w każdej chwili można wyświetlić w konsoli mysql:

```

mysql> show create table prac;
+-----+-----+
| Table | Create Table
|
+-----+-----+
| prac  | CREATE TABLE `prac` (
  `id` int(10) unsigned NOT NULL,
  `imie` char(20) NOT NULL,
  `nazwisko` char(30) NOT NULL,
  `zarobki` float(6,2) NOT NULL,
  `data_ur` date DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin |
+-----+-----+
1 row in set (0.000 sec)

```

Potwierdź, że polskie znaki prawidłowo wyświetlają się w konsoli mysql.

```

MariaDB [artur]> select * from prac;
+----+-----+-----+-----+-----+
| id | imie   | nazwisko | zarobki | data_ur |
+----+-----+-----+-----+-----+
| 1  | Artur  | Gramacki | 1000.00 | 1999-01-01 |
| 2  | Gal    | Anonim   | 2000.00 | 1999-01-01 |
| 3  | ąęśćźźńół | ĄĘŚĆŹŹŃÓŁ | 9999.00 | 2006-01-01 |
+----+-----+-----+-----+-----+
3 rows in set (0.000 sec)

MariaDB [artur]>

```

Testowo zmień domyślną stronę kodową w konsoli mysql, uruchamiając ją poleceniem:

```
shell> mysql -u lab -p --default-character-set=latin1
```

Wówczas polskie litery nie wyświetlą się prawidłowo

```

MariaDB [artur]> select * from prac;
+-----+-----+-----+-----+-----+
| id | imie      | nazwisko | zarobki | data_ur |
+-----+-----+-----+-----+-----+
| 1 | Artur     | Gramacki | 1000.00 | 1999-01-01 |
| 2 | Gal       | Anonim   | 2000.00 | 1999-01-01 |
| 3 | ???????~? | ???????Ë? | 9999.00 | 2006-01-01 |
+-----+-----+-----+-----+-----+
3 rows in set (0.000 sec)

```

Listę bieżących wartości parametrów bazy MySQL otrzymamy wydając polecenie:

```
mysql> show variables;
```

7. Utworzyć skrypt php, który:

- połączy się do bazy danych MySQL (odpowiednik polecenia: `mysql -u user -p`)
- wskaże bieżącą bazę danych (odpowiednik polecenia: `use db_name`)
- wyświetli zawartość tabeli `prac` (odpowiednik polecenia: `SELECT * FROM prac;`)

Uwaga 1: we wszystkich niżej przytaczanych skryptach pomijamy część nagłówkową html. Czynimy tak wyłącznie, aby zaoszczędzić miejsce. Każdorazowe dublowanie tego nagłówka w instrukcji do ćwiczeń jest niepotrzebne. Pamiętajmy jednak, że w docelowej aplikacji część nagłówkowa powinna być zawsze obecna (mimo, że współczesne przeglądarki potrafią się obejść bez nagłówka oraz potrafią pewnych informacji domyśleć się, np. strony kodowej pliku).

Uwaga 2: w kodach stosujemy styl *proceduralny*. Drugi dostępny styl (nowszy, bardziej „nowoczesny”) to *obiektowy*. W zasadzie ma żadnych różnic wydajnościowych między obu stylami programowania. Współczesne zaawansowane tzw. framework-i PHP (pozwalają szybciej i wygodniej tworzyć złożone aplikacje w php) korzystają niemal wyłącznie z podejścia obiekтового.

```

<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf8">
  <meta name="Keywords" content="słowo 1, słowo2">
  <meta name="Description" content="Stronka testowa">
  <meta name="Author" content="Artur Gramacki">
  <meta name="Copyright" content="(c) by Artur Gramacki">
  <meta http-equiv="Reply-to" content="a.gramacki@issi.uz.zgora.pl">
  <meta name="Language" content="pl">
  <meta name="Distribution" content="Global">
  <meta name="Robots" content="all">
  <meta name="Classification" content="Education">
  <title>(c) by Artur Gramacki</title>
</head>
<body>
  ...
</body>
</html>

```

Skrypt wygląda następująco:

```

<body>
<h3>Tabela pracownicy</h3>

<?php
// ZAMIEŃ host, użytkownika, hasło i bazę
// NA WŁAŚCIWE PARAMETRY TWOJEGO KONTA MySQL
$link = mysqli_connect ('localhost', 'artur', 'artur')
  or die ('Nie można połączyć się z MySQL.');
```

```

mysqli_select_db($link, 'artur')
  or die ('Nie można połączyć się z bazą.');
```

```

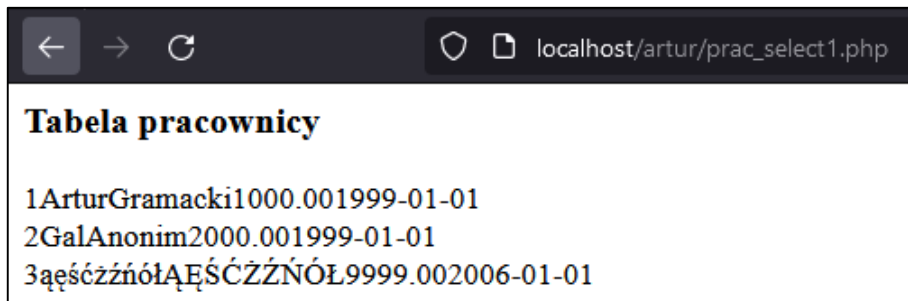
$wynik = mysqli_query ($link, 'SELECT * FROM prac;')
  or die ('Bład w zapytaniu do bazy.');
```

```

while ($rekord = mysqli_fetch_array ($wynik, MYSQLI_NUM)) {
  $id =      $rekord[0];
  $imie =   $rekord[1];
  $nazwisko = $rekord[2];
  $zarobki = $rekord[3];
  $data_ur = $rekord[4];
  print $id . $imie . $nazwisko . $zarobki . $data_ur;
  print '<br>';
}
?>
</body>

```

Wynik jaki uzyskujemy w przeglądarce nie jest jednak satysfakcjonujący. Jest on fatalnie sformatowany (a w zasadzie nie jest w ogóle sformatowany).



Uwaga: gdy nie wyświetlą się poprawnie polskie znaki należy w pierwszej kolejności zajrzeć do pliku `..\mysql\bin\my.ini` i odnaleźć w nim sekcję *UTF 8 Settings* i upewnić się, że są tam wpisane wartości, jak poniżej:

```

## UTF 8 Settings
character-set-server=utf8mb4
collation-server=utf8mb4_general_ci

```

Pierwszy parametr określa tzw. *domyślną metodę porównywania napisów* a drugi *domyślne kodowanie znaków* serwera MySQL. Po tych zmianach polskie litery zaczną się wyświetlać prawidłowo.

Aktualne ustawienia parametrów odpowiedzialnych za kodowanie znaków i porównywanie napisów odczytamy w następujący sposób:

```
mysql> show variables where variable_name like '%character%';
+-----+-----+
| Variable_name          | Value                               |
+-----+-----+
| character_set_client   | latin1                              |
| character_set_connection | latin1                              |
| character_set_database | utf8mb4                             |
| character_set_filesystem | binary                             |
| character_set_results  | latin1                              |
| character_set_server   | utf8mb4                             |
| character_set_system   | utf8                                 |
| character_sets_dir     | C:\Programy\xampp_8.2.4\mysql\share\charsets\ |
+-----+-----+
8 rows in set (0.001 sec)
```

```
mysql> show variables where variable_name like '%collation%';
+-----+-----+
| Variable_name          | Value                               |
+-----+-----+
| collation_connection  | latin1_swedish_ci                 |
| collation_database    | utf8mb4_general_ci               |
| collation_server      | utf8mb4_general_ci               |
+-----+-----+
3 rows in set (0.001 sec)
```

8. Gdy z jakiegoś powodu nie możemy modyfikować pliku *my.ini* (nie zawsze przecież jesteśmy administratorami serwera MySQL, często mamy tylko założone tam konto bez uprawnień administratora) musimy stosownie parametry zmienić „w locie”. Poprawimy więc poprzednio utworzony skrypt, gdzie oba problemy zostaną rozwiązane. Poprawne wyświetlanie polskich znaków zapewni nam wysłanie do serwera odpowiednich instrukcji. Właściwe formatowanie uzyskamy wstawiając pobrane z bazy dane do *tabeli html*. Po wprowadzonych zmianach dane z tabeli zostaną wyświetlone w dużo wygodniejszym formacie

test polskich literek:
aesczxnol AESCZXNOL
ąęśćźźńól AĘŚĆŹŹŃÓŁ

Tabela pracownicy

id	imie	nazwisko	zarobki	data_urodzenia
1	Artur	Gramacki	1000.00	1999-01-01
2	Gal	Anonim	2000.00	1999-01-01
3	ąęśćźźńól	AĘŚĆŹŹŃÓŁ	9999.00	2006-01-01


```

<body>
<?php
// ZAMIEŃ host, użytkownika, hasło i bazę
// NA WŁAŚCIWE PARAMETRY TWOJEGO KONTA MySQL
$link = mysqli_connect ('localhost', 'artur', 'artur')
    or die ('Nie można połączyć się z MySQL.');
```

id	imie	nazwisko	zarobki	data_urodzenia

```

mysqli_select_db($link, 'artur')
    or die ('Nie można połączyć się z bazą.');
```

```

$uchwyty1 = mysqli_query($link, "set names 'utf8' collate 'utf8_general_ci'");

$wynik = mysqli_query ($link, 'SELECT * FROM prac;')
    or die ('Błąd w zapytaniu do bazy.');
```

```

print "<table cellpadding=5 border=1>";
print "<tr>";
print "<td><b>id      </b></td>";
print "<td><b>imie    </b></td>";
print "<td><b>nazwisko </b></td>";
print "<td><b>zarobki  </b></td>";
print "<td><b>data_urodzenia </b></td>";
print "</tr>\n";

while ($rekord = mysqli_fetch_array ($wynik, MYSQLI_NUM)) {
    $id =      $rekord[0];
    $imie =   $rekord[1];
    $nazwisko = $rekord[2];
    $zarobki = $rekord[3];
    $data_ur = $rekord[4];

    print "<tr>";
    print "<td>$id      </td>";
    print "<td>$imie    </td>";
    print "<td>$nazwisko</td>";
    print "<td>$zarobki </td>";
    print "<td>$data_ur </td>";
    print "</tr>";
}
print "</table>";
?>
</body>
```

- Wykorzystamy teraz formularz html do pobrania danych od użytkownika i dodania nowego rekordu (polecenie `INSERT`) w tabeli `prac`. Dane z formularza będą przekazywane do serwera WWW z wykorzystaniem metody `GET`. W pętli `if ... else` sprawdzamy, czy mamy wysłać dane do serwera, czy też wyświetlić formularz. Dodatkowo sprawdzamy, czy udało się wstawić nowy rekord.

← → ↻ localhost/artur/prac_insert.php

Tabela pracownicy

id prac:

imie:

nazwisko:

zarobki:

data_ur:

(format daty: YYYY-MM-DD)

Zwróć uwagę, jak wgląda pasek adresowy w przeglądarce w czasie działania skryptu. Widać, że dane z formularza zostają przekazane do warstwy serwerowej za pomocą metody GET (patrz przykłady wyżej), czyli przez adres URL.

← → ↻ localhost/artur/prac_insert.php?id=5&imie=Jan&

Tabela pracownicy

INSERT INTO prac VALUES (5, 'Jan', 'Nowak', 5000, '200-01-01');

Wstawiono nowy rekord.

[Nowy rekord](#)

```
<body>
<h3>Tabela pracownicy</h3>

<?php
// Tak można wyłączyć wyświetlanie przez PHP ostrzeżeń.
// Ale trzeba uważać, gdyż nie każde ostrzeżenie można
// bezpiecznie zignorować. Więc może lepiej usunąć w kodach
// wszystkie źródła ostrzeżeń niż wyłączać ich wyświetlanie.
// error_reporting(E_ALL & ~E_NOTICE);

$link = mysqli_connect ('localhost', 'artur', 'artur')
    or die ('Nie można połączyć się z MySQL.');
```

```
mysqli_select_db($link, 'summit2')
    or die ('Nie można połączyć się z bazą.');
```

```
$uchwyty1 = mysqli_query($link, "set names 'utf8'");

if (isset($_GET['id']) &&
    isset($_GET['imie']) &&
    isset($_GET['nazw']) &&
    isset($_GET['zaro']) &&
    isset($_GET['d_ur'])) {
    $query = "INSERT INTO prac VALUES ($_GET[id], '$_GET[imie]', '$_GET[nazw]',
```

```

        $_GET[zaro], '$_GET[d_ur]');";
print "<pre>";
print_r($query);
print "</pre>";
if (mysqli_query ($link, $query) == TRUE) {
    print("Wstawiono nowy rekord.<br>");
} else {
    print("Nie udało się wstawić nowego rekordu.<br>");
}
print '<a href="prac_insert.php">Nowy rekord</a>';
} else {
    // W formularzu nie ma ACTION. Jeżeli nie wskazujemy skryptu do
    // obsługi formularza, zostanie użyty skrypt bieżący.
    // Formularz do dodawania nowego rekordu.
    print '<form method = "get">';
    print '<table>';
    print '<tr><td>id prac: </td><td><input type="text" name="id"> </td></tr>';
    print '<tr><td>imie: </td><td><input type="text" name="imie"> </td></tr>';
    print '<tr><td>nazwisko: </td><td><input type="text" name="nazw"> </td></tr>';
    print '<tr><td>zarobki: </td><td><input type="text" name="zaro"> </td></tr>';
    print '<tr><td>data_ur: </td><td><input type="text" name="d_ur"> </td></tr>';
    print '<tr><td>&nbsp; </td><td>(format daty: YYYY-MM-DD)</td></tr>';
    print '</table>';
    print '<input type = "submit" value = "Dodaj nowy rekord">';
    print '</form>';
}
?>
</body>

```

10. Na koniec pokażemy, jak wygląda kod wyświetlający dane z tabeli prac napisany wg. konwencji obiektowej. Widać wyraźnie, że w gruncie rzeczy nie różni się on zbytnio od analogicznego kodu napisanego w konwencji proceduralnej. Dodatkowo używamy tutaj funkcji `printf`, która jest bardziej elastyczna w formatowaniu wyniku, niż funkcja `print`.

```

<body>
<h3>Tabela pracownicy</h3>

<?php
$mysqli = new mysqli("localhost", "artur", "artur", "artur");
if ($mysqli -> connect_errno) {
    printf("Błąd połączenia do bazy: %s\n", $mysqli->connect_error);
    exit();
}

if ($result = $mysqli -> query("SELECT * FROM prac")) {
    printf("Polecenie SELECT zwróciło %d wiersz(e/y).<br>", $result->num_rows);
}

while ($row = $result -> fetch_array(MYSQLI_NUM)) {
    print $row[0] . $row[1] . $row[2] . $row[3] . $row[4];
    print "<br>";
}
$result->free();
$mysqli->close();

```

```
?>
```

```
</body>
```