

## Informatyka, Aplikacje internetowe i mobilne, semestr 5

### Projektowanie serwisów internetowych

#### Laboratorium nr 5

#### PHP – obsługa sesji i praca z bazą danych PostgreSQL

Celem laboratorium jest zapoznanie się z obsługą sesji i bazy danych PostgreSQL z wykorzystaniem materiału z poprzednich laboratoriów tj. obiektowego PHP, HTML i CSS.

Uwaga: W każdym zadaniu w przypadku generowanego kodu HTML i używanego kodu CSS sprawdź czy jest poprawny za pomocą odpowiedniego walidatora W3C.

***W pliku `sql_książki.zip` zawarto niezbędne struktury i dane dla bazy PGSQL dla schematu „książki”. Schemat o nazwie „książki” należy utworzyć w swoim lokalnym postgresie w bazie o nazwie postgres. Jeśli już taki istnieje oraz struktury wraz z danymi to ten etap można opuścić***

PHP - postgresql

<https://www.php.net/manual/en/book.pgsql.php>

<https://www.php.net/manual/en/function.pg-fetch-array.php>

PHP – PGSQL Tutorial

[https://www.tutorialspoint.com/postgresql/postgresql\\_php.htm](https://www.tutorialspoint.com/postgresql/postgresql_php.htm)

## Obsługa sesji w PHP

Praca z sesją wymaga, aby po otrzymaniu żądania klienta, PHP rozpoczynał nową lub wznowiał istniejącą sesję sprawdzając czy był już do niej przypisany ID sesji. Mechanizm wbudowany w PHP może działać na dwa sposoby:

- automatycznie, jeśli w konfiguracji PHP włączona została opcja `auto_start`,
- „ręcznie” czyli włączany przez projektanta aplikacji PHP przy wywołaniu funkcji `session_start()`.

Jeśli sesja istnieje, PHP odczytuje zmienne zarejestrowane w tej sesji. Jeśli nie, generowany jest nowy, unikalny identyfikator sesji. Można go odczytać w kodzie PHP za pomocą funkcji `session_id()`. Tablica przechowująca zmienne sesyjne o nazwie `$_SESSION` zostaje utworzona bez żadnych elementów. Programista w ramach sesji aplikacji dodaje do tej tablicy potrzebne elementy (zmienne i ich wartości). Więcej informacji na temat obsługi sesji można znaleźć pod adresem:

<https://www.php.net/manual/en/reserved.variables.session.php>.

## Ćwiczenie 1a – id sesji

Zbuduj poniższy skrypt PHP o nazwie ***sesje\_test\_a.php*** oraz sprawdź jego działanie.

```

1  <?php
2  session_start();
3  /* rozpoczęcie nowej lub wznowienie istniejącej sesji */
4  $_SESSION["moja_zmienna"]="Twoje imię i nazwisko";
5  /*zmienna sesyjna o nazwie moja_zmienna został utworzona*/
6  echo "Jestem ".$_SESSION["moja_zmienna"];
7  /* zostaje wydrukowany tekst: Moja zmienna */
8  ?>

```

Dopisz do niego jeszcze wyświetlenie ID sesji i przetestuj działanie.

## Ćwiczenie 1b – tablice

Zbuduj poniższy skrypt PHP o nazwie **sesje\_test\_b.php**, który wylosuje 10 liczb całkowitych z przedziału od -100 do 100 i umieści je w tablicy (array) o nazwie \$ts, a następnie wyświetli na stronie w formie tabeli HTML. Następnie tablica ta zostanie przypisana do sesji jako zmienna sesji o nazwie tablica. Po przeładowaniu strony wyświetl obie tablice: jedną - pobraną z sesji oraz drugą - nowo wylosowaną. Zapisz w sesji obie i ponownie wyświetl. Po kolejnym przeładowaniu w sesji powinny być już trzy tablice itd.

Jak można zauważyć, za pomocą sesji można przechowywać zmienne o typach prostych jak i bardziej złożonych, jak tablice czy obiekty.

## Obsługa bazy danych PostgreSQL w PHP

W tabeli poniżej znajduje się lista najczęściej używanych funkcji PostgreSQL.

Nazwa funkcji / metody	Opis
pg_connect(String \$conn_string, int \$flags = 0): PgSql\Connection false	otwiera połączenie z bazą danych PostgreSQL określoną przez zmienną \$conn_string. Przykład: \$conn_string = "host=postgres port=5432 dbname=postgres user=postgres"; \$conn = pg_connect(\$conn_string);
pg_query(PgSql\Connection \$connection, string \$query): PgSql\Result false	wykonuje zapytanie na określonym połączeniu z bazą danych. \$query = 'SELECT * FROM ksiazki.ksiazka'; \$result = pg_query(\$conn,\$query); Komentarz: usunięcie książki lub wstawienie rekordu, albo jego aktualizacja wiąże się z użyciem odp. Instrukcji SQL z użycie DELETE, INSERT lub UPDATE.
pg_affected_rows (\$result);	zwraca liczbę rekordów/wierszy, na które mają wpływ zapytania INSERT, UPDATE,DELETE i SELECT (od wersji >= 9.0). Przykład: \$ile_wierszy=pg_affected_rows (\$result);
pg_fetch_array(PgSql\Result \$result, ?int \$row = null, int \$mode = PGSQL_BOTH): array false	Pobiera wiersz jako tablicę. Przykład: while (\$line = pg_fetch_array(\$result, null, PGSQL_ASSOC)) { foreach (\$line as \$col_value) { echo "\$col_value\n"; } }
<a href="#">pg_fetch_all()</a> , <a href="#">pg_fetch_row()</a>	Inne funkcje do przetwarzania rezultatu funkcji zwróconej przez funkcję pg_query().

<a href="#">pg_free_result()</a>	Ta funkcja musi być wywoływana tylko wtedy, gdy zużycie pamięci podczas wykonywania skryptu stanowi problem. W przeciwnym razie cała pamięć wyników zostanie automatycznie zwolniona po zakończeniu skryptu.
<code>pg_close(?Pgsql\Connection \$connection = null): bool</code>	zamyka nietrwałe połączenie z bazą danych PostgreSQL skojarzoną z daną instancją połączenia. Przykład: <code>pg_close(\$conn);</code>
Inne funkcje Postgresql	Więcej informacji – <a href="#">funkcje postgresql</a> .

Podczas pracy z bazą PgSQL warto zajrzeć do podręcznika: <https://php.net/manual/en/book.pgsql.php>.

## Komentarz od DevEnv / Postgresql

Obejrzyj zawartość pliku konfiguracyjnego C:\DevEnv\xampp\php\php.ini znajdź poniższy fragment:

```
935 ;extension=pdo_oci
936 extension=pdo_odbc
937 extension=pdo_pgsql
938 extension=pdo_sqlite
939 extension=pgsql
```

**Do czego służy plik php.ini?**

**Jak myślisz do czego zapisy w formie `extension = ???` służą w kontekście wykonania poniższego Ćwiczenia 2.**

## Ćwiczenie 2 –praca z bazą danych PostgreSQL

Uruchom lokalnie swój serwer PostgreSQL (za pomocą skryptu `pgsql_start.bat` z katalogu C:\DevEnv\xampp). Przygotuj przykładowy skrypt php o nazwie **ksiazki.php**, którego zadaniem jest wyświetlenie zawartości tabeli **ksiazka** z bazy postgres z użyciem znanego schematu **ksiazki**.

Podczas analizy kodu zwróć uwagę na konstrukcje:

- Połączenia z bazą PgSQL – `pg_connect`,
- Wykonanie zapytania – `pg_query`,
- Opracowanie wyników – przedstawienie ich w formie HTML,
- Zwolnienie pamięci,
- Zamknięcie połączenia – `pg_close()`, **uwaga: nie zapominaj o tej czynności w przyszłości.**



Katedra Systemów Informatycznych  
Uniwersytet Morski w Gdyni

```
<!DOCTYPE html>
<html lang="pl-PL">
<head>
<meta charset="UTF-8">
<title>Książki</title>
</head>
<body>

<?php
// definicja parametrów połączenia.
$host = 'localhost';
$port = '5432';
$database = 'postgres';
$user = 'postgres';
$password = 'postgres'; // hasło należy zweryfikować;

// zdefiniowanie połączenia w formie łańcucha
$conn_string = "host=$host port=$port dbname=$database user=$user password=$password";
$conn = pg_connect($conn_string) or die ("Nie można podłączyć się do Postgresql\n");
// Zbudowanie zapytania SQL query do tabeli książka w schemacie książki
$query = 'SELECT * FROM książki.książka';
$result = pg_query($conn,$query) or die('Błąd zapytania: ' . pg_last_error());
// wydruk rezultatu w formie tabeli HTML
echo "<table>\n";
while ($line = pg_fetch_array($result, null, PGSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

// zwolnienie pamięci
pg_free_result($result);

// zamknięcie połączenia
pg_close($conn);
?>
</body>
</html>
```

## Zadanie 1

### Założenia podstawowe:

Należy zorganizować internetową rekrutację pracowników z wykorzystaniem PHP, HTML5 oraz CSS3, mechanizmu sesji oraz zapisu do bazy danych PostgreSQL. Dane wypełniamy nie w jednym formularzu/stronie, ale w kolejnych krokach – stronach / podformularzach. Tak wypełnianie dane powinny być zapamiętywane na poszczególnych etapach za pomocą zmiennej obiektowej typu klasy pracownik (w której do określania wartości składowych samej klasy wykorzystuje się mechanizm obsługi sesji PHP - zapamiętywać wartości z poprzedniego formularza). Na końcu działania internetowej rekrutacji dane powinny zostać zapisane do bazy PostgreSQL za pomocą metody z klasy pracownik. W metodach zaprojektowanych w klasie pracownik w przypadku skorzystania np. z dostępu do bazy, zapytania SQL należy korzystać z metod określonych w klasie o nazwie db\_pgsql ( np. połączenia z bazą, uruchomienia zapytania, itp.). Podczas przechodzenia do kolejnego formularza należy wyświetlać na dole zawartość tablicy \$\_SESSION (za pomocą print\_r) oraz ewentualnie numeru sesji za wykorzystaniem funkcji: session\_id().

Dane należy gromadzić w tabeli **pracownicy** zlokalizowanej w nowym schemacie o nazwie „**rekrutacja**” PostgreSQL. Zakres i typy danych w PostgreSQL to: idp typu SERIAL, nazwisko (varchar 50), imię (varchar 25) , wiek (Integer), doświadczenie (TEXT), zainteresowania (TEXT). Wszystkie w tabeli pracownicy są wymagane – NOT NULL.


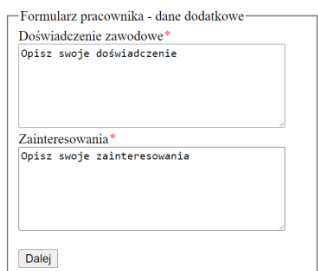
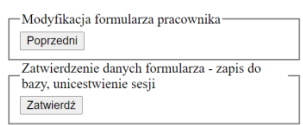
### Założenia techniczne / organizacja plików:

W dalszej części przewodnika w specjalnej tabeli przedstawiono szczegółowe założenia projektowe dla poszczególnych plików projektu – należy je uwzględnić. Niech skrypt główny **fp.php** będzie pełnić rolę internetowego formularza rekrutacji nowego pracownika do zakładu pracy. W rozwiązaniu należy tak zorganizować skrypt główny, aby po odczytaniu za pomocą metody GET numeru strony (parametr o nazwie np. ST) do skryptu fp.php będzie dołączany w zależności od odczytanej numeru strony odpowiednia podstrona/odpowiedni podformularz o nazwie **.f1.html, f2.html, .f3.html, .f4.html**.

Proszę zwrócić uwagę, że nazwy plików rozpoczynają się od kropek co w systemie linux przekłada się na właściwość ukrytych plików.

Każdy formularz/podstrona (oprócz .f4.html) powinna posiadać w sekcji **<form action = 'URL' method='POST' ??? inne parametry>** , gdzie zamiast URL ustawiamy adres **fp.php?ST=[nr\_strony]**. Oznaczeniu **[nr\_strony]** przypisujemy odpowiednie wartości 1,2,3,4 w zależności od formularza jaki budujemy i na jaki kolejny formularz/stronę kierujemy użytkownika. Kod danej strony/formularza (oprócz .f4.html) należy tak zorganizować, aby oprócz samych pól można było podczas wypełniania danych w kolejnych odpowiednich etapach zawsze cofnąć się lub przejść dalej do kolejnego etapu za pomocą przycisku „Dalej”, „Poprzedni” z założeniem, że powrót oznacza dla nas odczytanie za pomocą obiektu klasy pracownik oraz mechanizmu sesji wcześniej uzupełnionych wartości na poprzednich formularzach i ustawianiu ich w odpowiednich polach danego formularza, który chcemy wyświetlić. Wypełnienie pól w formularzu powinny być wymagane (właściwość required). Nie skupiamy się w tym zadaniu na typowej walidacji pól w PHP, czy JAVASCRIPT np. za pomocą wyrażeń regularnych.

Projekt powinien zawierać następującą strukturę plików. Jeśli zajdzie potrzeba można dołożyć odpowiednio własne pliki jednak z zachowaniem poniższej podstawowej oczekiwanej struktury.

Nazwa pliku	Opis wykorzystania
fp.php	<pre> 1 &lt;!DOCTYPE html&gt; 2 &lt;html lang="pl-PL"&gt; 3 &lt;head&gt; 4 &lt;meta charset="UTF-8"&gt; 5 &lt;title&gt;Formularz pracownika&lt;/title&gt; 6 &lt;link rel="stylesheet" href="fp.css" type="text/css"&gt; 7 &lt;/head&gt; 8 &lt;body&gt; 9 &lt;?php 10 // rozpoczęcie sesji nowej lub jej kontynuacja 11 session_start(); 12 /* dołączenie klasy pracownik - w kolejnym etapie kodowania 13 // require_once 'pracownik.php'; 14 // utworzenie obiektu klasy pracownik - w kolejnym etapie kodowania 15 // \$p = new pracownik(); 16 */ 17 if (isset(\$_GET['ST'])) 18     \$page = 1; else \$page=\$_GET['ST']; 19 \$form = ".fp.\$page.".html"; 20 include_once(\$form); 21 // zaprojektuj zabezpieczenie przed użyciem nr ST spoza zbioru: 1,2,3,4 22 ?&gt; 23 &lt;/body&gt; 24 &lt;/html&gt; </pre> <p>Kod źródłowy formularza pracownika fp.php z komentarzami sSkrypt główny PHP zawierający odpowiednie elementy HTML5, dołączony arkusz styli fp.css). W wierszach od 12 do 16 umieszczono w komentarzu przykład kodu dołączania pliku z definicją klasy pracownik oraz utworzenia nowego obiektu klasy pracownik.</p>
.f1.html	<p><b>Formularz rekrutacji pracownika - krok 1</b></p>  <p>Zawartość \$_SESSION</p> <p>Array ( )</p> <p>Session ID: 669blmsul1ube8e4ufj59o9vbs</p> <p>Formularz powinien posiadać pola: nazwisko, imię typu text oraz wiek typu number. Wartości domyślne dla tych pól - tak jak przedstawiono na rysunku. Należy przemyśleć w jaki sposób w polach formularza określać „value”. W celu określenia wartości pola np. nazwisko można użyć zaprojektowanej ogólnej metody np. z klasy pracownik o nazwie get_field_value z wykorzystaniem także mechanizmu sesji. Wówczas fragment definicji pola nazwisko w formularzu może mieć postać:</p> <pre>.... value="&lt;?=\$_GET['nazwisko']"&gt;</pre> <p>Pod formularzem należy umieścić informację o zawartości \$_SESSION oraz ID Sesji.</p>
.f2.html	<p><b>Formularz rekrutacji pracownika - krok 2</b></p>  <p>Zawartość \$_SESSION</p> <p>Array ( [nazwisko] =&gt; Podaj nazwisko [imie] =&gt; Podaj imię [wiek] =&gt; 21 )</p> <p>Przed definicją formularza powinno nastąpić zapamiętanie podanych wartości z poprzedniego formularza <b>.f1.html</b>. Można to uzyskać za pomocą metody set_field_value() z klasy pracownik.</p> <p>Formularz <b>.f2.html</b> powinien posiadać pola typu textarea: doświadczenie, zainteresowania. Wartości domyślne dla tych pól - tak jak przedstawiono na rysunku. Należy w polach formularza określać „value” podobnie jak wyżej dla .f1.html. Fragment definicji pola doświadczenie w formularzu może mieć postać:</p> <pre>.... value="&lt;?=\$_GET['doswiadczenie']"&gt;</pre> <p>Formularz powinien posiadać dwa przyciski z odpowiednimi adresami URL – żądania odpowiednich stron/formularzy. Wybranie Poprzedni powoduje powrót do poprzedniego formularza z danymi podstawowymi. Wybranie następny powoduje przejście do formularza .f3.html. Pod formularzem należy umieścić informację o zawartości \$_SESSION oraz ID Sesji.</p>
.f3.html	<p><b>Formularz rekrutacji pracownika - krok 3</b></p> <p><b>Etap końcowy</b></p> <p>Końcowe czynności wypełniania formularza</p>  <p>Zawartość \$_SESSION</p> <p>Array ( [nazwisko] =&gt; Podaj nazwisko [imie] =&gt; Podaj imię [wiek] =&gt; 21 [doswiadczenie] =&gt; Opisz swoje doświadczenie [zainteresowania] =&gt; Opisz swoje zainteresowania )</p> <p>Przed definicją formularza powinno nastąpić zapamiętanie podanych wartości z poprzedniego formularza <b>.f2.html</b>. Można to uzyskać za pomocą metody set_field_value() z klasy pracownik.</p> <p>Formularz <b>.f3.html</b> posiada tylko przycisk:</p> <ul style="list-style-type: none"> <li>- umożliwiający powrót do poprzedniego formularza</li> <li>- zatwierdzający dane – przejście do formularza <b>.f4.html</b> w którym powinno nastąpić zapisanie danych do bazy Postgresql oraz unicestwienie sesji.</li> </ul> <p>Pod formularzem należy umieścić nadal informację o zawartości \$_SESSION oraz ID Sesji.</p>
.f4.html	<p><b>Formularz rekrutacji pracownika - krok 4 / końcowy</b></p> <p>Dane zostały zapisane do bazy</p> <p>Sesja została usunięta / zniszczona</p> <p>Zawartość \$_SESSION</p> <p>Nie istnieje</p> <p>Strona, która generuje odpowiednie akapity z treścią komunikatów świadczących o poprawnym wykonaniu rejestracji pracownika do bazy oraz usunięciu sesji.</p> <p>W wersji produkcyjnej należałoby wyświetlić:</p> <p><i>Dziękujemy za rejestrację. Wkrótce skontaktujemy się Panią / Panem w celu podania dalszych kroków rekrutacji.</i></p>

db_pgsql.php	<p>W tym pliku należy umieścić definicję klasy <b>db_pgsql</b>, w której powinny zostać zdefiniowane prywatne składowe klasy np.:</p> <pre>private \$host = 'localhost'; private \$port = '5432'; private \$user = 'postgres'; private \$password = 'postgres'; // hasło i login należy zweryfikować private \$database = 'postgres'; private \$conn = null; private \$status_connection = ''; private \$status;</pre> <p>oraz inne niezbędne składowe, a także metody w postaci <b>publicznych funkcji</b>:</p> <p><b>connect()</b> – podłączenie do PostgreSQL z wykorzystaniem w/w składowych, metody <b>pg_connect()</b>, nawiązane/uzyskane połączenie powinno być przypisane do składowej <b>\$conn</b>, poprawność połączenia może być zapamiętana za pomocą składowej <b>\$status</b>, jako wartość boolowska</p> <p><b>disconnect()</b> – rozłączenie połączenia z PostgreSQL z wykorzystaniem <b>pg_close()</b>,</p> <p><b>query(\$sql)</b> – wykonanie zapytania za pomocą <b>pg_query(\$conn,\$sql)</b>, metoda powinna zwracać prawdę lub fałsz w zależności od prawidłowości wykonania zapytania,</p> <p><b>getStatus()</b> – uzyskanie statusu poprawności połączenia z PostgreSQL. Zwraca wartość składowej <b>\$status</b></p>
pracownik.php	<p>Przed definicją klasy pracownik należy zawrzeć instrukcję dołączającą klasę <b>db_pgsql</b>, gdyż metody z tej klasy powinny być wykorzystane w klasie pracownik.</p> <p>Zdefiniowana w PHP klasa pracownik powinna zawierać prywatne składowe odpowiadające rejestrowanym danym za pomocą formularza fp.php i jego podformularzy. Oto przykładowe składowe:</p> <pre>private \$nazwisko; private \$imie; private \$wiek; private \$doswiadczenie; private \$zainteresowania;</pre> <p><b>Metody w klasie pracownik:</b></p> <p>Konstruktor klasy pracownik powinien przypisywać wartości w/w składowych za pomocą weryfikacji wartości <b>\$_SESSION</b> poprzez ustawienie wartości tych składowych na domyślne (w przypadku jak zmienna sesyjna nie istnieje/nie jest ustawiona), albo na takie wartości na jak wskazuje istniejąca zmienna sesyjna.</p> <p>Metody <b>set_field_value(\$field_name)</b>, <b>get_field_name(\$field_name)</b> powinny być tak zaprojektowane, aby umożliwiły odczyt wybranej prywatnej składowej klasy: \$nazwisko, \$imie itp.</p> <p>Dodatkowo warto zaimplementować funkcję zapisu danych pracownika <b>insert_to_pgsql()</b> z wykorzystaniem składowych klasy pracownik oraz metod klasy <b>db_pgsql</b>.</p>
fp.css	<p>Arkusze styli – dołączany do plik głównego skryptu fp.php</p>

## Zadanie 2

Wykorzystując z zadania 1 klasy `db_pgsq`, `pracownik` oraz zawartość `.f1.html`, `.f2.html` zbuduj skrypt PHP z wykorzystaniem HTML5 i CSS3 o nazwie `lista_pracownikow.php`, który będzie wyświetlał listę pracowników (zarejestrowanych przez formularz) z możliwością usunięcia, edycji danych wybranego pracownika z listy. W praktyce może warto rozważyć dodanie odpowiednich metod `lista_pracownikow(...)`, `usun_pracownika(...)` itd. w klasie `pracownik`.

### Uwaga!

**Plik** o nazwie `PSI_LAB5_[nr_alubumu_studenta].zip` zawierający folder z rozwiązaniami:

`sesje_test_a.php`, `sesje_test_b.php`, `ksiazka.php`, pliki `fp.php`, `fp.css`, `.f[1-4].html`, `db_pgsq.php`, `pracownik.php`, `lista_pracownikow.php` + inne niezbędne pliki prześlij w sprawozdaniu w systemie **Sprawer**. W sprawozdanie umieść także linki do zdalnego katalogu na serwerze uczelnianym zawierającego skrypty/pliki z rozwiązaniami w/w zadań oraz ćwiczeń.