
PRÓBNY EGZAMIN POTWIERDZAJĄCY KWALIFIKACJE W ZAWODZIE

Zima 2021, część praktyczna

EE.09 – Programowanie, tworzenie i administrowanie stronami internetowymi i
bazami danych

Przed rozpoczęciem pracy z arkuszem, zapoznaj się z instrukcją podaną poniżej:

1. Zanim zaczniesz pracę, sprawdź, czy arkusz zawiera 7 stron, czy wszystkie wydruki są czytelne, czy nie ma żadnych błędów. Ewentualne błędy zgłoś nauczycielowi.
2. Zapoznaj się z treścią zadań. Pamiętaj o dokładnej analizie treści zadania.
3. Przed rozpoczęciem zapoznaj się z instrukcją do wypakowania plików. Na pulpicie będzie znajdowało się archiwum zabezpieczone hasłem podanym w arkuszu.
4. Przed oddaniem testu, sprawdź, czy udzieliłeś odpowiedzi na wszystkie pytania.
5. Jeżeli zadanie wydaje się mieć błędy merytoryczne, zgłoś wątpliwość nauczycielowi po zakończeniu egzaminu.

Powodzenia!

Instrukcja

Po zalogowaniu na konto użytkownika, na pulpicie znajdziesz plik *ezdamin_3.zip*.

Wypakuj je dostępnym narzędziem do wypakowywania archiwum (np. WinRar, 7Zip) podając hasło **m!kol1#r**. Utwórz folder w którym umieścisz **wszystkie pliki**, które są związane z Twoim zadaniem.

Uruchom usługę MySQL oraz Apache korzystając z programu Xampp lub Laragon.

Zaloguj się do aplikacji PHPMyAdmin i wykonaj w niej zadanie 1. Zadanie wymagające utworzenia pliku .php testuj, zapisując je w folderze **htdocs**.

Zadanie 1.

Twoim zadaniem jest wykonanie na bazie danych MySQL poniższych zadań:

- a) utwórz bazę danych o nazwie *pociagi*
- b) zaimportuj do niej plik *pociagi.sql* z archiwum *ezdamin_3.zip*. Zrób odpowiednie zrzuty ekranu. Na zrzucie widoczny musi być pasek zadań.
- c) wykonaj zapytania SQL na bazie danych:
 - i) Wybierz *miasto_poczatkowe*, *miasto_koncowe*, *numer_pociagu*, *godzina_odjazdu*, *kategoria_pociagu* z tabeli **polaczenia**, gdzie *miasto_poczatkowe* ma w sobie słowo "Wrocław"
 - ii) Wybierz *miasto_poczatkowe*, *miasto_koncowe*, *numer_pociagu*, *godzina_odjazdu* z tabeli **polaczenia** i odpowiadające każdemu rekordowi *opoznienie* z tabeli **statusy**, posortowane rosnąco według *godzina_odjazdu*, wyłącznie 4 rekordy (ograniczenie narzucone w kwerendzie).
 - iii) Dodaj nowe połączenie do tabeli **polaczenia**, dla kolejnych kolumn: "Łódź Fabryczna", "Berlin Hbf.", "DB3003", "04:33", "międzynarodowy".
 - iv) Aktualizujące w tabeli **statusy** pole *opoznienie* na 210 dla połączenia o id równym 10.

Zadanie 2.

Przygotuj witrynę internetową według specyfikacji podanej poniżej.

TrainTravel

Sprawdź połączenia kolejowe z Twojego miasta

Skąd

→

Dokąd

Sprawdź połączenie

Dostępne połączenie kolejowe

Stacja początkowa	Stacja końcowa	Numer pociągu	Godzina odjazdu
Kraków Główny	Katowice	B34330	14:30 (+4)
Katowice	Warszawa Cent.	5540	14:45
Rzeszów Główny	Białystok	17717	14:59
Gdańsk Główny	Wrocław Główny	30300	15:11 (+ 130)

Specyfikacja kodu witryny

- Witryna napisana jest w języku PHP. Nazwa pliku strony to **pociagi.php**.
- Strona posiada standard kodowania polskich znaków.
- Na karcie przeglądarki widoczny jest tytuł “TrainTravel – portal kolejowy”
- Arkusz stylów **pociagi.css** prawidłowo umiejscowiony i działający
- Strona podzielona na sekcje:
 - a) u góry znajduje się sekcja nawigacji, a w niej:
 - i) obrazek o nazwie *traintravel.png*,
 - b) sekcja główna, a w niej:
 - i) nagłówek 3. stopnia o treści “Sprawdź połączenia kolejowe z Twojego miasta!”,
 - ii) poniżej kontener, a w nim:

- 1) formularz wysyłający dane metodą jawną dla użytkownika, a w nim:
 - (a) nagłówek 4. stopnia o treści “Skąd”
 - (b) pole tekstowe przyjmujące miasto początkowe
 - (c) zdjęcie o nazwie “strzałka.png”
 - (d) nagłówek 4. stopnia o treści “Dokąd”
 - (e) przycisk wysyłający formularz o treści “Sprawdź połączenie”
- iii) nagłówek 3. stopnia o treści “Dostępne połączenie kolejowe”
- iv) tabela o id *legenda*, a w niej
 - 1) cztery komórki nagłówkowe o treści kolejno: “Stacja początkowa”, “Stacja końcowa”, “Numer pociągu”, “Godzina odjazdu”
 - 2) wiersze, będące wynikiem działania **skryptu 1**

Specyfikacja arkusza stylów

- a) czcionka dla całej strony: Montserrat
- b) nawigacja: szerokość 100%, wysokość 160px, margines dolny 8px
- c) obrazek w nawigacji: margines lewy 25px, margines górny 20px
- d) nagłówek 3. stopnia: kolor czcionki #17466E, wielkość czcionki 18px, pogrubiony, margines lewy 50px
- e) kontener w sekcji głównej: szerokość 100%, wysokość 150px, tło #F2F2FF, marginesy wewnętrzne górny i dolny 15px, margines dolny 50px
- f) formularz: margines lewy 64px, margines górny 24px, obiekty ustawione w linii
- g) obrazek w formularzu: margines górny 40px, szerokość automatyczna, wysokość 30px, margines prawy 32px
- h) nagłówek 4. stopnia: rozmiar czcionki 12px, kolor czcionki #17466E, margines dolny 4px
- i) pole tekstowe: szerokość 330px, wysokość 45px, obramowanie 1px linią ciągłą o kolorze #D9D9D9, margines prawy 48px

- j) przycisk: szerokość 215px, wysokość 60px, tło #17466E, zaokrąglenie rogów 15px, czcionka pogrubiona, kolor czcionki biały, rozmiar czcionki 16px, margines lewy 96px, margines górny 24px
- k) tabela o id *legenda*: szerokość 100%, kolor czcionki czarny, czcionka pogrubiona, margines górny 32px, obramowanie tabeli i komórki ma być połączone (collapse)
- l) komórka nagłówkowa: wysokość 45px, wielkość czcionki 12px, margines wewnętrzny lewy 48px, kolor tła #F2F2FF, tekst ustawiony do lewej
- m) wiersz tabeli: szerokość 100%
- n) komórka tabeli: wysokość 45px, obramowanie 1px linią ciągłą o kolorze #D9D9D9, zaokrąglenie rogów 5px, obramowanie górne 5px linią ciągłą o kolorze #EDAF1F, kolor czcionki #17466E, margines wewnętrzny lewy 48px
- o) gdy pociąg jest opóźniony: kolor czcionki #C10000

Specyfikacja skryptu

Skrypt 1 powinien zostać napisany w języku PHP i spełniać wymagania podane poniżej.

Skrypt 1:

- a) łączy się z bazą danych na serwerze o adresie localhost, użytkownikiem jest root bez hasła, nazwa bazy danych – **pociagi**
- b) odbiera dane z formularza na stronie, jeżeli pola edycyjne są puste, następuje sprawdzenie ciasteczek:
 - i) jeżeli ciasteczko jest puste – do bazy danych wysyłane jest **zapytanie 2**, zgodne z treścią zadania bazodanowego na stronie 2
 - ii) jeżeli ciasteczko zawiera dane – do bazy danych wysyłane jest zmodyfikowane **zapytanie 2**, tak, że powinny zostać zwrócone połączenia dla stacji początkowej podanej w ciasteczku
- c) jeżeli nie są puste, do bazy danych wysyłane jest zmodyfikowane **zapytanie 2**, tak, że powinny zostać zwrócone połączenia dla stacji początkowej i stacji końcowej podanych z formularza

d) powinny zostać wygenerowane bloki połączeń kolejowych, o schemacie opisanym wyżej, gdzie:

- i) stacja początkowa odpowiada polu *stacja_poczkowa*, stacja docelowa odpowiada polu *stacja_docelowa*, numer pociągu odpowiada polu *numer_pociagu*, godzina odjazdu odpowiada polu *godzina_odjazdu*
- ii) jeżeli opóźnienie jest większe niż zero, godzina odjazdu powinna zostać wyświetlona w formacie *<godzina_odjazdu (+ <opoznienie>)>*
- iii) gdy odwiedzający wejdzie na stronę po raz pierwszy i formularz przesłał dane, tworzone jest ciasteczko z czasem trwania 4h o nazwie **stacjap**, o wartości równej **stacji początkowej** przesłanej formularzem, w innym wypadku nie tworzy ciasteczka
- iv) po zakończeniu działania skrypt zamyka połączenie z serwerem.

Pamiętaj o zapisaniu zrzutów ekranu z kwerend oraz wszystkich plików ze skryptami, wraz z plikami arkuszy stylów oraz skryptem!

Wybrane funkcje MySQLi:

`mysqli_connect(serwer, użytkownik, hasło, nazwa_bazy)` – łączy się z serwerem i zwraca id połączenia, lub zwraca FALSE, gdy nie uda się połączyć

`mysqli_select_db(id_połączenia, nazwa_bazy)` – zwraca TRUE, jeżeli baza istnieje, FALSE jeżeli baza nie istnieje

`mysqli_query(id_połączenia, zapytanie)` – wysyła zapytanie do bazy danych, zwraca wynik zapytania

`mysqli_fetch_row(wynik_zapytania)` – tablica z danymi danego wiersza wyniku zapytania, gdzie kolejne indeksy to kolumny zapytania

`mysqli_num_rows(wynik_zapytania)` – zwraca ilość wierszy w wyniku danego zapytania

`mysqli_close(id_połączenia)` – zamyka połączenie z bazą danych

bool setcookie (string \$name string \$value = "" , int \$expire = 0 , string \$path = "" , string \$domain = "" , bool \$secure = false, bool \$httponly = false)

`setcookie()` defines a cookie to be sent along with the rest of the HTTP headers. Like other headers, cookies must be sent before any output from your script (this is a protocol restriction). This requires that you place calls to this function prior to any output, including `<html>` and `<head>` tags as well as any

whitespace.

Once the cookies have been set, they can be accessed on the next page load with the `$_COOKIE` array. Cookie values may also exist in `$_REQUEST`.

name - The name of the cookie.

value - The value of the cookie. This value is stored on the clients computer; do not store sensitive information. Assuming the name is 'cookienam', this value is retrieved through `$_COOKIE['cookienam']`

expire - The time the cookie expires. This is a Unix timestamp so is in number of seconds since the epoch. In other words, you'll most likely set this with the `time()` function plus the number of seconds before you want it to expire. Or you might use `mktime()`. `time()+60*60*24*30` will set the cookie to expire in 30 days. If set to 0, or omitted, the cookie will expire at the end of the session (when the browser closes).

path - The path on the server in which the cookie will be available on. If set to '/', the cookie will be available within the entire domain. If set to '/foo/', the cookie will only be available within the /foo/ directory and all sub-directories such as /foo/bar/ of domain. The default value is the current directory that the cookie is being set in.

domain - The (sub)domain that the cookie is available to. Setting this to a subdomain (such as 'www.example.com') will make the cookie available to that subdomain and all other sub-domains of it (i.e. w2.www.example.com). To make the cookie available to the whole domain (including all subdomains of it), simply set the value to the domain name ('example.com', in this case).

secure - Indicates that the cookie should only be transmitted over a secure HTTPS connection from the client. When set to TRUE, the cookie will only be set if a secure connection exists. On the server-side, it's on the programmer to send this kind of cookie only on secure connection (e.g. with respect to `$_SERVER["HTTPS"]`).

httponly - When TRUE the cookie will be made accessible only through the HTTP protocol. This means that

the cookie won't be accessible by scripting languages, such as JavaScript. It has been suggested that this setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers), but that claim is often disputed. Added in PHP 5.2.0. TRUE or FALSE