Współpraca bazy danych ze stroną internetową, etap IV

Źródło: domena publiczna.

Współczesne sposoby reagowania na sytuacje wyjątkowe w czasie komunikacji bazy danych ze stroną internetową wykraczają poza klasyczne podejście oparte na przerwaniu wykonywania skryptu PHP. Dużo bardziej subtelną i pożądaną przez klienta witryny metodą jest tzw. obsługa wyjątków. Pozwala ona w razie pojawienia się problemów z połączeniem wyświetlić lepiej wyglądające w przeglądarce rezultaty żądania HTTP.

Programista webowy, starając się poprawić jakość kodu, powinien także poznać obiektowy algorytm współpracy bazy danych z witryną. Ułatwia to zarządzanie dużymi projektami. Warto także dokonać refaktoryzacji kodu odpowiedzialnego za przetwarzanie rezultatów wykonania zapytania `SELECT`, zastępując klasyczną pętlę `while` dużo prostszą w zapisie konstrukcją `foreach`.

W skład serii wchodzą również e-materiały:

- Współpraca bazy danych ze stroną internetową, etap I,
- Współpraca bazy danych ze stroną internetową, etap II,
- Współpraca bazy danych ze stroną internetową, etap III.

> **Twoje cele**
>
> - Poznasz korzyści wynikające z wyodrębnienia do zewnętrznego pliku danych dostępowych do bazy.

- Ocenisz różnicę między zwyczajnym przerwaniem dalszego wykonania skryptu w razie wystąpienia błędu, a podejściem nowoczesnym, opartym na tzw. wychwytywaniu wyjątków.
- Skonstruujesz obiektowy algorytm współpracy bazy danych z witryną i poznasz przetwarzanie obiektowe rekordów: `fetch_object()`.
- Używając pętli `foreach`, zrealizujesz w bardziej czytelny sposób przetwarzanie rezultatów odpowiedzi serwera na zapytanie `SELECT`.

Przeczytaj

Współczesne techniki współpracy witryny z bazą danych

W tym e-materiale poznamy cztery współczesne, istotne usprawnienia algorytmu komunikacji skryptu PHP z bazą danych.

- **Umieszczenie danych konfiguracyjnych** połączenia **w zewnętrznym pliku**, możliwym do wywołania w pozostałych skryptach komunikujących się z bazą danych.
- **Obsługa wyjątków** `try .. catch` zamiast „twardego" zatrzymania wykonania skryptu funkcją `or die()`.
- **Wariant obiektowy** współpracy z bazą danych (zamiast klasycznej wersji proceduralnej), w tym nowa metoda fetchowania obiektowego o nazwie `fetch_object()`.
- **Wykorzystanie pętli** typu `foreach` do przetwarzania rezultatów wyszukiwania w bazie.

Znajomość tych metod jest ważna z punktu widzenia **optymalizacji tworzonych kodów źródłowych**. Omówimy zatem nie tylko sposób wdrożenia tych metod, lecz także logiczne **przyczyny** wprowadzenia usprawnień i wynikające z nich **konsekwencje** – zarówno dla programisty webowego, jak i użytkownika tworzonej witryny.

Umieszczenie danych dostępowych do bazy w zewnętrznym pliku

Pierwszym etapem realizowania skryptu współpracującego z bazą danych MySQL jest zawsze podanie **informacji dostępowych** potrzebnych do **nawiązania połączenia z bazą**. Początkowo w skryptach zazwyczaj dokonujemy tego, używając danych dostępowych jako **argumentów** zapisanych wprost wewnątrz funkcji `mysqli_connect()`:

```
1 $c = mysqli_connect("localhost", "root", "", "nazwa_bazy");
```

Jednak w przypadku tworzenia **całych witryn**, a więc **wielu skryptów** realizujących w aplikacji przeróżne zadania, wielokrotne powtarzanie tych danych dostępowych wydaje się nieoptymalne. W przypadku konieczności **zmiany** konfiguracji serwisu (na przykład wskutek przeniesienia na inny serwer) jesteśmy zmuszeni poprawić dane dostępowe **w każdym skrypcie osobno**.

> **Ciekawostka**

> Jeżeli podczas przenoszenia witryny na inny serwer dojdzie do przeoczenia konieczności wprowadzenia zmian w którymkolwiek z wielu plików, powstanie serwis internetowy, w którym poprawnie działa tylko wybrana część skryptów (a my w ogóle nie będziemy o tym wiedzieć). Konieczność **testowania każdej funkcjonalności** osobno okaże się zadaniem **wyjątkowo czasochłonnym**.

Współcześnie postępujemy więc inaczej: opis **aktu nawiązania połączenia** oraz **wszystkie dane dostępowe** umieszczamy w specjalnie wydzielonym, **zewnętrznym pliku**. W razie konieczności dokonania zmian w danych dostępowych poprawki aplikujemy **tylko w jednym skrypcie**, zamiast w wielu rozproszonych zasobach. Specjalnie utworzony, dodatkowy plik możemy nazwać np. `connect.php`. Oto jego przykładowa zawartość:

```
<?php
    // Konfiguracja dostępu do bazy danych
    $host = "localhost";
    $user = "root";
    $pass = "";
    $db = "nazwa_bazy";
?>
```

Następnie **w każdym** skrypcie PHP, w którym **łączymy się z bazą danych**, użyjemy **instrukcji jednokrotnego żądania** specjalnego pliku:

```
<?php
    require_once "connect.php";
?>
```

W ten sposób dane konfiguracyjne zostaną „podpięte” we wszystkich potrzebnych dokumentach. Jest to idea znana z technologii CSS – style elementów witryny również przenosimy do **zewnętrznego arkusza**, aby w razie chęci dokonania zmian uniknąć wprowadzania dużej liczby poprawek w wielu miejscach jednocześnie.

> **Ciekawostka**
>
> Testowanie współpracy z bazą danych również staje się dzięki opisanej technice łatwe: wystarczy sprawdzenie jednego ze skryptów korzystających z bazy. Wiele współczesnych frameworków przestrzega tzw. **podejścia DRY**, umieszczając **kluczowe informacje konfiguracyjne** w zewnętrznym pliku. W przypadku popularnego systemu Wordpress jest to zasób o nazwie `wp-config.php`.

Przyczyny oraz **sposób implementacji** zewnętrznego pliku zostały także przedstawione w filmie zawartym w sekcji „Film samouczek”.

Obsługa wyjątków: `try .. catch`

Instrukcję takiej postaci stosujemy współcześnie zamiast klasycznej obsługi błędów komunikacji z bazą danych z użyciem funkcji `or die()`. Tak zwane **wyłapywanie wyjątków** pozwala dużo elastyczniej reagować na poszczególne problemy w aplikacji webowej, zamiast po prostu „zabijać" wykonanie skryptu.

Używając instrukcji `try`, podejmujemy próbę nawiązania połączenia z bazą; w razie wystąpienia problemów (gdy kod błędu jest inny niż zero) dokonujemy tzw. **rzucenia wyjątku**. Korzystamy przy tym z zapisu `throw new Exception`. Taki wyjątek, mający **unikalny identyfikator**, można później „złapać" w specjalnej sekcji `catch`:

```
<?php
    try {
        $c = mysqli_connect("localhost", "root", "", "nazwa_bazy"

        if ($c->connect_errno!=0) {
            throw new Exception(mysqli_connect_errno());
        }

    } catch (Exception $error) {
        echo 'Błąd bazy danych: '.$error->getMessage();
    }
?>
```

W profesjonalnych **aplikacjach webowych** przetworzeniem obiektu o nazwie `$error` można się zająć za pomocą specjalnie napisanych własnych funkcji (a w przypadku zastosowania technik obiektowych używając metod, czyli funkcji zdefiniowanych wewnątrz klas). **Kod błędu** pobiera metoda o nazwie `mysqli_connect_errno()`, wywołana już w momencie rzucania wyjątku. Wewnątrz sekcji `catch` dostęp do tego kodu uzyskujemy dzięki metodzie `getMessage()` wywołanej na rzecz obiektu `$error`

Wykrycie błędu, połączone z **unikatową reakcją na jego rodzaj** (osiąganą za sprawą identyfikatora numerycznego), daje możliwość zaprogramowania inteligentnej reakcji skryptu na problemy z komunikacją z serwerem. Nie musimy już zwyczajnie „zabijać" dalszego przetwarzania dokumentu – możemy użytkownika **ostrzec**, **przekierować** albo **wyświetlić zawartość zastępczą**. Współczesne zaawansowane aplikacje internetowe zapewniają internaucie właśnie taką „miękką" reakcję na kłopoty z bazą danych, zamiast brutalnie uniemożliwiać przeglądanie witryny.

Użycie konstrukcji `try` .. `catch` zazwyczaj łączymy z pierwszą opisaną techniką, czyli z **przeniesieniem aktu nawiązania połączenia z bazą do zewnętrznego pliku**. Więcej

szczegółów na temat zrealizowania obsługi połączenia w taki sposób podajemy w filmie samouczku.

Wariant obiektowy współpracy witryny z bazą danych

We współczesnym programowaniu wyróżniamy **dwa podstawowe paradygmaty**, czyli przyjęte konwencje budowania aplikacji. W podejściu proceduralnym stosujemy **funkcje**, czyli wydzielone fragmenty kodu, które realizują konkretne zadania. Konwencja obiektowa jest bardziej abstrakcyjna i złożona, gdyż używamy **obiektów** skonstruowanych na bazie szablonów zwanych **klasami**. Każdy obiekt ma przypisane pewne **atrybuty** (zmienne) oraz **metody** (funkcje zdefiniowane wewnątrz klas, a służące do przetwarzania zmiennych).

Programowanie **zorientowane obiektowo** jest trudniejsze do zrozumienia, jednak to właśnie za sprawą rozbudowanej struktury logicznej zapewnia utrzymanie **porządku w kodzie**, **chroni dane przed niepożądanym nadpisaniem** (dzięki tzw. hermetyzacji), a ponadto **ułatwia rozbudowę aplikacji** (tzw. skalowanie). Współpraca witryny z bazą danych za pośrednictwem biblioteki `mysqli` także może zostać zrealizowana obiektowo.

Nawiązanie połączenia z bazą danych

W wersji obiektowej pojawia się **dodatkowa klauzula** `new`, która tworzy nowy obiekt o nazwie `$c`. Metoda `mysqli()` to tzw. **konstruktor obiektu**. Kolejność argumentów wewnątrz nawiasów okrągłych pozostaje w obu wariantach identyczna.

wersja proceduralna	wersja obiektowa
`$c = mysqli_connect(...);`	`$c = new mysqli(...);`

Obsługa polskich znaków

Podobnie jak w wersji proceduralnej, odpowiednią linię kodu umieszczamy **już po nawiązaniu aktywnego połączenia**. Tym razem jednak zapis ma następującą postać: `obiekt->metoda("argument");`

wersja proceduralna	wersja obiektowa
`mysqli_set_charset($c,"utf8");`	`$c->set_charset("utf8");`

Wykonanie zapytania SQL

Zakładamy, że treść kwerendy do wykonania została uprzednio umieszczona w pomocniczej zmiennej `$q`.

---

wersja proceduralna	wersja obiektowa
`$result = mysqli_query($c, $q);`	`$result = $c->query($q);`

Pobranie liczby zwróconych rekordów

W tym przypadku mamy do czynienia nie z **metodą** obiektu, tylko z jego **atrybutem** o nazwie `num_rows` (zwróć uwagę na brak nawiasów okrągłych, właściwych zapisowi metody):

wersja proceduralna	wersja obiektowa
`$ile = mysqli_num_rows($result);`	`$ile = $result->num_rows;`

Fetchowanie pojedynczych rekordów

W poprzednich e-materiałach poznaliśmy **trzy sposoby fetchowania** rekordów do tablicy jednowymiarowej:

- `mysqli_fetch_row()` – **przetwarzanie liczbowe** – ponumeruj (od zera) szufladki tablicy jednowymiarowej;
- `mysqli_fetch_assoc()` – **wyjmowanie asocjacyjne (skojarzeniowe)** – ponazywaj słownie szufladki tablicy jednowymiarowej, nadając im zawsze taki sam identyfikator, jaki mają kolumny w bazie danych;
- `mysqli_fetch_array()` – **przetwarzanie równocześnie numeryczne i asocjacyjne** (dostępne są oba rodzaje indeksów).

Zapis obiektowy tych metod (tutaj na przykładzie przetwarzania liczbowego) wygląda następująco:

```
1 $c = mysqli_connect("localhost", "root", "", "nazwa_bazy");
2 $q = "SELECT * FROM tabela";
3 $result = $c->query($q);
4
5 while ($row = $result->fetch_row()) {
6     echo $row[0];
7 }
```

Dla każdej z trzech klasycznych metod pozbywamy się przedrostka `mysqli_` występującego w nazwie instrukcji, a ponadto `$result` nie jest już argumentem funkcji, tylko obiektem umieszczonym po lewej stronie operatora: `->`.

Na rzecz obiektu `$result` wywołujemy metodę `fetch_row()`, która kopiuje do tablicy jednowymiarowej **pojedynczy rekord**. W bieżącym e-materiale poznamy jeszcze możliwość przetwarzania rekordów nową, czwartą metodą o nazwie `fetch_object()`.

Fetchowanie obiektowe `fetch_object`

Podejście obiektowe umożliwia także przetwarzanie rekordów nie w postaci **tablic jednowymiarowych**, lecz **obiektów mających atrybuty**:

```
$c = mysqli_connect("localhost", "root", "", "nazwa_bazy");
$q = "SELECT login, email FROM users";

if ($result = $c->query($q)) {
    while ($obj = $result->fetch_object()) {
        echo $obj->login;
        echo $obj->email;
    }
}
```

Obiekt o nazwie `$obj` nada atrybuty noszące **nazwy takie same,** jak **kolumny w tabeli** bazy danych. Zamiast podawania indeksów tablicy z użyciem nawiasów kwadratowych, stosujemy w tym przypadku zapis `obiekt->atrybut`.

> **Ciekawostka**
>
> Idea przechowywania atrybutów wydobytych z bazy danych **wewnątrz obiektów**, a nie **tablic**, stała się bardzo rozpowszechniona w wielu współczesnych **frameworkach webowych**. Warto więc jak najszybciej przyzwyczaić się do takiego modelu fetchowania rekordów – czas na to poświęcony zwróci się później z nawiązką.

Zwolnienie pamięci zajmowanej przez rekordy

Kolejnym logicznym usprawnieniem działania skryptu współpracującego z bazą danych jest **zwolnienie pamięci** zajmowanej przez wyjęte rekordy. Oczywiście może to nastąpić dopiero w momencie, gdy dane znajdujące się w `$result` **nie są już w skrypcie dłużej potrzebne**. Takie wyczyszczenie pamięci ma największy sens w skryptach wykonujących wiele zapytań, a generujących duże liczby zwracanych rekordów.

wersja proceduralna	wersja obiektowa
`mysqli_free_result($result);`	`$result->free_result();`

Zamknięcie połączenia z bazą danych

Na rzecz obiektu `$c` reprezentującego nawiązane połączenie wywołujemy metodę o nazwie `close()`. W tym przypadku argument staje się zbędny, gdyż po lewej stronie operatora `->` znajduje się już informacja, które z istniejących połączeń zamierzamy zakończyć:

wersja proceduralna	wersja obiektowa
`mysqli_close($c);`	`$c->close();`

Wykorzystanie pętli `foreach`

Tego typu instrukcja iteracyjna upraszcza wizualnie klasyczny zapis fetchowania rekordów. Nagłówek pętli jest dużo prostszy w porównaniu do analogicznej instrukcji `while`. Po lewej stronie operatora `as` znajduje się **obiekt** `$result` klasy `mysqli_result`, przechowujący **wszystkie wyjęte z bazy rekordy**. Natomiast po prawej stronie tego operatora znajduje się pomocnicza, **jednowymiarowa tablica** `$row`, w której komórkach umieszczamy atrybuty **pojedynczego rekordu**.

```
1 $c = mysqli_connect("localhost", "root", "", "nazwa_bazy");
2 $q = "SELECT login, email FROM users";
3 $result = $c->query($q);
4
5 foreach ($result as $row) {
6     echo $row["login"];
7     echo $row["email"];
8 }
```

> **Ciekawostka**
>
> Zapis `as` (ang. „jako”) możemy logicznie potraktować tak samo, jak znany z języka SQL operator o identycznej nazwie, stosowany w kwerendach – jest to przecież logicznie **nowy alias** właśnie fetchowanego rekordu.

Słownik

> **programowanie obiektowe**
>
> konwencja projektowania oprogramowania zalecająca traktowanie programu komputerowego jako współpracujących ze sobą abstrakcyjnych obiektów, mających pewne atrybuty (określające stan) oraz metody (funkcje definiujące zachowania); obiekty

skonstruowane na podstawie szablonów (klas) współpracują ze sobą w celu wykonywania określonych zadań

programowanie proceduralne

konwencja projektowania oprogramowania zalecająca dzielenie instrukcji na realizujące konkretne zadania funkcje lub procedury (funkcje zwracają wartość, zaś procedury nie); program komputerowy traktujemy jako zbiór współpracujących ze sobą funkcji, wymieniających między sobą wartości nieglobalne

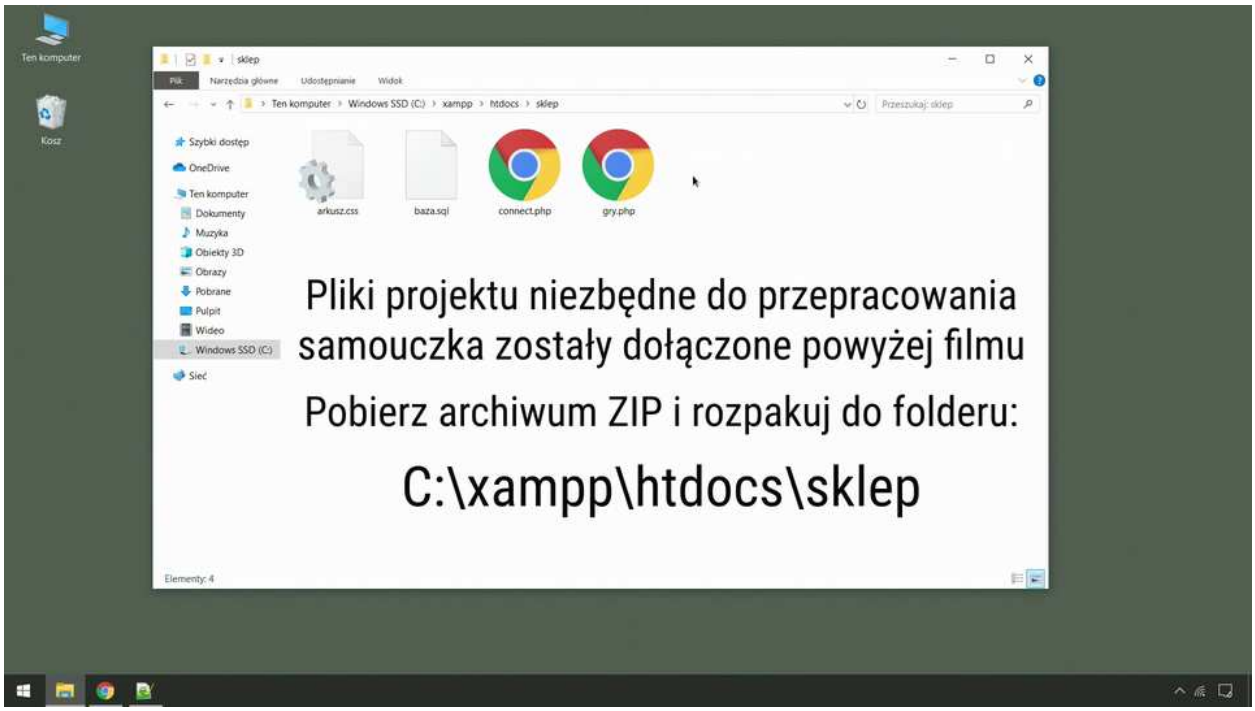
zasada DRY

(od ang. *Don't Repeat Yourself* – „nie powtarzaj się") reguła projektowania oprogramowania, która zaleca unikanie stosowania tych samych fragmentów kodów źródłowych (lub bardzo zbliżonych) w wielu miejscach; lepszą praktyką jest wyodrębnienie powtarzającego się fragmentu z ewentualnym zastąpieniem wartości parametrami i jedynie odwoływanie się do niego w licznych wystąpieniach

Film samouczek

> **Polecenie 1**
>
> Zapoznaj się z filmem instruktażowym, realizując kolejno przedstawione czynności na własnym stanowisku komputerowym. Niezbędne do wykonania zadania pliki w archiwum ZIP znajdziesz poniżej:

Plik o rozmiarze 1.89 KB w języku polskim



Film dostępny pod adresem /preview/resource/RCJZ1uwJHL32Y

Film nawiązujący do treści lekcji dotyczącej baz danych.

Polecenie 2

Wykonaj kolejną podstronę serwisu, która zawiera **listę wyboru** `<select>` wypełnioną **gatunkami gier** wydobytymi (dzięki słowu kluczowemu `DISTINCT`) z tabeli `gry` oraz **przycisk** podsumowujący formularz.

Po kliknięciu przycisku wypisane zostaną na **liście nienumerowanej** `<ul>`, dodanej pod formularzem, wszystkie gry komputerowe znajdujące się w bazie **należące do wybranego gatunku** gier.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Przyporządkuj instrukcje biblioteki `mysqli` do odpowiedniej grupy.

Wariant proceduralny `mysqli`

Wariant obiektowy `mysqli`

```
$c->close();
$c = new mysqli();
mysqli_close($c);
$c = mysqli_connect();
$c->set_charset("utf8");
$c->query($q);
mysqli_query($c, $q);
mysqli_set_charset($c,"utf8
```

Ćwiczenie 2

Z tabeli w bazie danych wyjęto kwerendą tylko login użytkownika. Połącz w pary rodzaje fetchowania z odpowiednim sposobem dostępu do danych.

`mysqli_fetch_assoc()`	`$row[0]`
`fetch_object()`	`$row["login"]`
`mysqli_fetch_row()`	`$row->login`

Ćwiczenie 3

Wskaż, którą funkcję należy dodać, aby włączyć do kodu skryptu PHP zawartość zewnętrznego pliku `file.php` zawierającego kod źródłowy.

- ◯ `require_once "file.php"`
- ◯ `fget "file.php"`
- ◯ `uses "file.php";`
- ◯ `return_once "file.php";`

Ćwiczenie 4

Wskaż, jaką ma postać pobranie liczby zwróconych do obiektu `$result` rekordów w wariancie obiektowym biblioteki `mysqli`.

- ◯ `num_rows($result);`
- ◯ `$result->num_rows;`
- ◯ `num_rows->$result;`
- ◯ `mysqli_num_rows($result);`

Ćwiczenie 5

Uzupełnij elementy algorytmu nawiązania połączenia z bazą danych, używającego konstrukcji `try .. catch`.

```
try {
    $c = [          ] mysqli($host, $user, $pass, $db);

    if ($c->[          ]!=0) {
      throw new Exception(mysqli_connect_errno());
    }

    $c->set_charset("[          ]");
} catch (Exception [          ]) {
    echo 'Błąd bazy danych: '.$error;
}
```

$error | $c | connect_errno | utf8 | new | localhost

Ćwiczenie 6

Wskaż, którą instrukcję w języku PHP definiuje zapis `foreach()`.

- ◯ Przypisania, dla atrybutów obiektu.
- ◯ Wyboru, dla elementów tablicy.
- ◯ Warunkową, dla atrybutów obiektu.
- ◯ Pętli, dla elementów tablicy.

Ćwiczenie 7

Połącz w pary podane zapisy z odpowiadającymi im rodzajami instrukcji obiektowych.

`$result->fetch_object();`	konstruktor
`$result->num_rows;`	metoda
`$c = new mysqli();`	atrybut

Ćwiczenie 8

Wskaż, który ciąg liczb zostanie wypisany w wyniku działania pętli zapisanej w języku PHP.

```php
$numbers = [1,7,13];
foreach($numbers as $number)
{
    if ($number >= 7)
    {
        $number = (-1) * $number;
        echo $number." ";
    }
}
```

- ◯ 7 13
- ◯ -1 -7
- ◯ -7 -13
- ◯ -1 -7 -13

Dla nauczyciela

Autor: Mirosław Zelent
Przedmiot: Informatyka

Temat: Współpraca bazy danych ze stroną internetową, etap IV

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:
> > >
> > > > e) programuje elementy strony internetowej współpracujące z sieciową bazą danych;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Poznasz korzyści wynikające z wyodrębnienia do zewnętrznego pliku danych dostępowych do bazy.

- Ocenisz różnicę między zwyczajnym przerwaniem dalszego wykonania skryptu w razie wystąpienia błędu, a podejściem nowoczesnym, opartym na tzw. wychwytywaniu wyjątków.
- Skonstruujesz obiektowy algorytm współpracy bazy danych z witryną i poznasz przetwarzanie obiektowe rekordów: `fetch_object()`.
- Używając pętli `foreach`, zrealizujesz w bardziej czytelny sposób przetwarzanie rezultatów odpowiedzi serwera na zapytanie `SELECT`.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- telefony z dostępem do internetu.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Współpraca bazy danych ze stroną internetową, etap IV". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów zajęć, przejście do wspólnego ustalenia kryteriów sukcesu.

2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Przeczytaj". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek". Uczniowie oglądają film instruktażowy, realizując kolejno przedstawione czynności na swoim stanowisku komputerowym. Pliki potrzebne do wykonania zadania znajdują się w archiwum ZIP.
3. **Ćwiczenie umiejętności.** Uczniowie realizują indywidualnie ćwiczenia nr 1–8, po ich wykonaniu porównują otrzymane wyniki z inną osobą.

Faza podsumowująca:

1. Nauczyciel zadaje pytania podsumowujące, np.
   – czym się różni programowanie proceduralne od obiektowego?
   – na czym polega reguła zwana zasadą DRY?
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 2 z sekcji „Film samouczek". Ich zadaniem jest wykonanie kolejną podstrony serwisu, która posiada listę wyboru

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.