Uniwersytet Wrocławski

Bazy danych 2022

Wykład 7

Powtórzenie podstaw HTML i PHP; połączenia z MySQL w PHP

Bartosz Brzostowski

Wydział Fizyki i Astronomii UWr
semestr letni r. akad. 2021/22

21 kwietnia 2022

Minimalny HTML

```
<html>
  <head>
    <title>...</title>
    <meta ...>
  </head>
  <body>
    <table>
      <tr>
        <th>...</th>...
      </tr>
      <tr>
        <td>...</td>...
      </tr>
      ...
    </table>
  </body>
</html>
```

- Do wyświetlania wyników zapytań można wykorzystywać tabele `<table>...</table>`
- Wewnątrz tabeli definiuje się wiersze `<tr>...</tr>...`
- `...` składające się z komórek zwykłych `<td>...</td>` lub nagłówkowych `<th>...</th>` (różnią się domyślnym stylem)

Interfejs DML: formularze HTML

```
<form action="..." method="...">
  <input type="..." name="..." value="...">
  <textarea name="...">...</textarea>
  <select name="..." [multiple]>
    <option value="..." [selected]>...</option>
    ...
  </select>
  ...
</form>
```

- `action`: adres, pod który zostaną odesłane dane z formularza = adres skryptu, który je przetworzy
- `method`: `GET` (domyślna) albo `POST`
  - Różnica semantyczna: zgodnie z nazwą metody, `POST` użyjemy do modyfikacji danych w bazie, `GET` — do wyświetlenia np. którejś strony wyników zapytania
  - Różnica praktyczna: wartości wysłane `GET` widać w pasku adresu przeglądarki, `POST` — nie

Elementy `<input>`

- `name`: nazwa zmiennej związanej z danym elementem
- Sens i zachowanie zależne od atrybutu `type`:
  - `submit`, `reset`: przyciski do wysyłania i czyszczenia formularza, nie służą do przekazywania danych (`value` — napis na przycisku)
  - `text`, `password`: jednolinijkowe pola tekstowe (`value` — początkowa zawartość)
  - `radio`, `checkbox`: pola jedno- i wielokrotnego wyboru
    - zazwyczaj kilka z tym samym `name` i różnymi `value`, które są wartościami wysyłanymi w zależności od zaznaczeń pól
    - dla `checkbox` może być wysyłanych kilka różnych wartości pod jednym `name`: żeby było kolekcją, musi się kończyć na `[]`
  - `hidden`: niewidoczny dla użytkownika, służy do przesłania konkretnej wartości danej zmiennej, której użytkownik nie powinien modyfikować
  - wiele nowych w HTML5

Elementy `<textarea>` i `<select>`

- `<textarea>...</textarea>`: wielolinijkowe pole tekstowe, zamiast `value` — wnętrze elementu
- `<select>`: menu rozwijane (*drop-down*)
  - Zawiera listę opcji `<option>...</option>` — wnętrze elementu jest wyświetlane, atrybut `value` to wartość, która zostanie przypisana zmiennej zadeklarowanej w `name`... elementu `select`
  - Z atrybutem `multiple` pole wielokrotnego wyboru (na pecetach: z przytrzymaniem Ctrl); `name` powinno się kończyć na `[]` jak dla `checkbox`

PHP

- Interpretowany język do generowania stron WWW
- Najczęściej „uruchamiany” przez przeglądarkę: serwer interpretuje PHP, przeglądarka — HTML
- Jest też interpreter w linii poleceń:

```
$ cat helloworld.php
<?php
$msg = "Hello, world!";
echo $msg . "\n";
$ php < helloworld.php
Hello, world!
```

- Każdy dokument HTML (czy wręcz tekstowy) jest programem PHP
- Blok kodu rozpoczyna znacznik `<?php`, kończy (opcjonalnie) `?>`
- Język C(++)-podobny: wąsy, średniki, `if`, `for`...
- Zmienne poprzedzone `$`, typowane dynamicznie (jak Python)

Tablice w PHP

- Inicjalizacja: `$t = array();` lub `$t = [];`
- Z konkretnymi wartościami: `$t = array(v1, v2, ...);` lub `$t = [v1, v2, ...];`
- Domyślnie klucze całkowitoliczbowe: `$t[0]`, `$t[1]`, ...
- Mogą zachowywać się jak słowniki:
  `$t = array(k1 => v1, k2 => v2, ...);` lub
  `$t = [k1 => v1, k2 => v2, ...];`
- Mutowalne: `$t[] = val;` dopisuje pod kolejnym kluczem
- `unset($t[klucz])` usuwa element, `unset($t)` — tablicę
- Iterator `foreach($t as $v) {...}` oraz `foreach($t as $k => $v) {...}`
- `implode($sep, $t)` zwija tablicę do napisu z separatorem `$sep` (elementy muszą być konwertowalne na napisy)
- `array_map(funkcja, $t1, $t2, ...)` zwraca tablice „przepuszczone” przez funkcję (tyloargumentową, ile tablic); funkcja może być anonimowa

PHP cd.

- Obsługa formularza: przysłane wartości znajdują się w tablicy `$_POST` lub `$_GET`, zależnie od metody
- Klucze — takie jak atrybuty `name` pól formularza, dla pól złożonych (`checkbox`, `select multiple`) — zagnieżdżenie
- Funkcja `print_r($t)` drukuje strukturę tablicy, najlepiej umieścić jej wynik w elemencie `<pre>...</pre>`
- W ramach pracy własnej: obsługa sesji (zmienna `$_SESSION`) — zwłaszcza dla osób piszących aplikację dla wielu użytkowników z logowaniem; funkcje biblioteczne

Sterowniki bazodanowe w PHP

- Rozszerzenia PHP do obsługi baz danych: warstwy abstrakcji (ODBC, PHP Data Objects) lub dedykowane dla specyficznych DBMS
- Dedykowane rozszerzenia dla MySQL, PostgreSQL (`pgsql`), SQLite[3], MS SQL (`mssql`)..., także dla NoSQL: MongoDB, Tokyo Cabinet (`tokyo_tyrant`); więcej: `https://www.php.net/manual/en/refs.database.php`
- MySQL:
  - `mysql` — przestarzały, niedostępny od PHP7
  - MySQLi (`mysqli`) — *improved*; pełna dokumentacja: `https://www.php.net/manual/en/book.mysqli.php`
- W MySQLi: obiekty klas `mysqli`, `mysqli_result` i in., można pisać w stylu proceduralnym lub obiektowym

Nawiązywanie połączenia z bazą

- `$link = mysqli_connect(...);` lub
  `$link = new mysqli(...);`
- Argumenty kolejno: host, nazwa użytkownika, hasło, nazwa bazy, port; domyślne wartości zależne od konfiguracji PHP (poza nazwą bazy — tu: pusta); przykład:
  `$link = new mysqli("localhost", "plg", "...", "plg_projekt");`
- Sprawdzanie poprawności połączenia:
  - Proceduralnie: `if (!$link) die("Nie udało się...");` (tutaj `$link` jest `FALSE`)
  - Obiektowo: `if ($link->connect_errno) die("...");`

albo: mysqli.connect.errno()

Modyfikowanie połączenia

- Zmiana bieżącej bazy (jak `use` w kliencie MySQL)
  - Proceduralnie: `mysqli_select_db($link, "nowa_baza");`
  - Obiektowo: `$link->select_db("nowa_baza");`
- Zmiana użytkownika
  - `mysqli_change_user($link, "nowy_user", "hasło", "nowa_baza");`
  - `$link->change_user("nowy_user", "hasło", "nowa_baza");`
- Zamknięcie połączenia (choć i tak wykonywane na końcu skryptu)
  - `mysqli_close($link);`
  - `$link->close();`
- Wszystkie powyższe zwracają `TRUE` lub `FALSE` w zależności od powodzenia

Wysyłanie zapytań

Tutaj chodzi o poprawność niektórych zapytań, ale też o częściowe zabezpiecznie przed SQL injection - można też

- `mysqli_query($link, $zapytanie);` lub używać PREPARED `$link->query($zapytanie);` STATEMENTS patrz klasa mysqli_stmt
- `$zapytanie` jest napisem zawierającym zapytanie w SQL
- Przy tworzeniu zapytań należy dbać o znaki specjalne, np.

```
$esc_var = $link->real_escape_string($var);
$result = $link->query("SELECT * FROM tabela WHERE
nazwa = '$esc_var'");
```

- W przypadku niepowodzenia zwraca `FALSE`, w przypadku powodzenia:
  - dla zapytań `INSERT`, `UPDATE`, `DELETE`: `TRUE`
  - dla pozostałych zapytań (w tym `SELECT`): obiekt klasy `mysqli_result`
- Liczba „dotkniętych” wierszy: `mysqli_affected_rows($link)` lub `$link->affected_rows`

Iterowanie po obiekcie `mysqli_result`

- `mysqli_fetch_row($result)` lub `$result->fetch_row()` zwraca *kolejny* wiersz z wyniku (lub `NULL`, jeśli więcej brak)
- Typowe użycie:
  ```
  while ($row = $result->fetch_row()) {...}
  ```
- Wynikowa tablica ma klucze numeryczne; alternatywnie:
  - `fetch_assoc` — klucze o nazwach jak w zapytaniu
  - `fetch_array` — oba zestawy kluczy

Przykład

tu podajemy nasze hasło do MySQL, np. na panoramx-ie, jeśli nic nie zmienialiśmy, zostało ono zapisane w pliku ~/.mysql

```
<?php

$link = mysqli_connect("localhost", "plg", "...", "plg_sprawdzian");

if(!$link) die("Brak połączenia z bazą");

echo "<pre>";

$result = $link->query("SELECT * FROM dzielo");

while ($row = $result->fetch_row()) // lub assoc, lub array
  print_r($row);

$result->close();

echo "</pre>";

$link->close();
```

nasza nazwa użytkownika

nazwa bazy z którą chcemy pracować

Efekt dla `fetch_row()`

```
Array
(
    [0] => 1
    [1] => Feynmana wykłady z fizyki - Mechanika kwantowa
    [2] => Feynman, Richard
    [3] => fizyka kwantowa
    [4] => 1964
)
Array
(
    [0] => 2
    [1] => Pan raczy żartować, panie Feynman!
    [2] => Feynman, Richard
    [3] => non-fiction
    [4] => 1985
)
...
```

Efekt dla `fetch_assoc()`

```
Array
(
    [idd] => 1
    [tytul] => Feynmana wyklady z fizyki - Mechanika kwantowa
    [autor] => Feynman, Richard
    [kategoria] => fizyka kwantowa
    [rokpowstania] => 1964
)
Array
(
    [idd] => 2
    [tytul] => Pan raczy żartować, panie Feynman!
    [autor] => Feynman, Richard
    [kategoria] => non-fiction
    [rokpowstania] => 1985
)
...
```

Efekt dla `fetch_array()`

```
Array
(
    [0] => 1
    [idd] => 1
    [1] => Feynmana wykłady z fizyki - Mechanika kwantowa
    [tytul] => Feynmana wykłady z fizyki - Mechanika kwantowa
    [2] => Feynman, Richard
    [autor] => Feynman, Richard
    [3] => fizyka kwantowa
    [kategoria] => fizyka kwantowa
    [4] => 1964
    [rokpowstania] => 1964
)
Array
(
    [0] => 2
    [idd] => 2
...
```

Iterowanie cd.

- `$result->fetch_object()` zwraca obiekt zadanej klasy z atrybutami zainicjalizowanymi wartościami z kolejnego wiersza wyniku
- `$result->fetch_all()` zwraca tablicę, której elementy to wiersze wyniku; zestaw kluczy w wierszach wskazuje opcjonalny argument o wartości `MYSQLI_NUM` (domyślnie) / `MYSQLI_ASSOC` / `MYSQLI_BOTH`
- `$result->data_seek($n)` przechodzi do zadanego argumentem wiersza (numeracja od 0)
- `$result->close()` uwalnia pamięć

print_r($res->fetch_all(MYSQLI_ASSOC))

```
Array
(
    [0] => Array
        (
            [idd] => 1
            [tytul] => Feynmana wykłady z fizyki - Mechanika kwantowa
            [autor] => Feynman, Richard
            [kategoria] => fizyka kwantowa
            [rokpowstania] => 1964
        )

    [1] => Array
        (
            [idd] => 2
            [tytul] => Pan raczy żartować, panie Feynman!
            [autor] => Feynman, Richard
            [kategoria] => non-fiction
            [rokpowstania] => 1985
        )
...
```