Podstawy programowania III

WYKŁAD 2

Jan Kazimirski

Komunikacja z bazami danych

PHP i bazy danych

- PHP zapewnia dostęp do wielu popularnych baz danych.
- Kilka poziomów abstrakcji:
  - Funkcje obsługujące konkretne bazy danych
  - Biblioteka PEAR DB
  - Biblioteka ADOdb
  - Warstwa PDO

PostgreSQL - połączenie

- Nawiązywanie połączenia:

resource ***pg_connect****(string)*

- Funkcja wymaga podania parametrów połączenia
- Funkcja zwraca uchwyt do bazy danych lub wartość FALSE.

PostgreSQL – parametry połączenia

- Łańcuch znakowy w formie "klucz=wartość ..."
- Parametry:
  - host – nazwa serwera bazy danych
  - port – numer portu bazy danych
  - dbname – nazwa bazy danych
  - user – nazwa użytkownika
  - password – hasło użytkownika

PostgreSQL – koniec połączenia

- Zakończenie połączenia zwykle nie jest konieczne (połączenie zamykane jest automatycznie po zakończeniu skryptu)
- Zamknięcie połączenia:

bool ***pg_close****([resource])*

- Funkcja zwraca wartość TRUE (powodzenie) lub FALSE (niepowodzenie).

PostgreSQL – wysłanie zapytania

- Po uzyskaniu połączenia z bazą danych można do niej przesłać zapytanie SQL
- Wysłanie zapytania:

  resource ***pg_query****(resource* ***db****,string* ***query****)*

- Rezultatem jest uchwyt do rezultatu wykonania kwerendy lub FALSE (niepowodzenie)

PostgreSQL – pobieranie informacji

- Rezultat zapytania zwrócony przez funkcję pg_query jest “zapakowany” - do uzyskania rekordów można użyć różnych funkcji:
  - pg_fetch_row – zwraca wiersz w postaci tablicy
  - pg_fetch_assoc – zwraca wiersz w postaci tablicy asocjacyjnej
  - pg_fetch_object – zwraca wiersz w postaci obiektu

PostgreSQL - pg_fetch_row

- Zwraca rekord (wiersz) rezultatu zapytania w postaci tablicy indeksowanej numerycznie

  *array **pg_fetch_row**(resource **rez** [, int **wiersz**])*

- Numer wiersza jest opcjonalny. Jeżeli nie podany to zwracany jest kolejny wiersz.
- W przypadku niepowodzenia funkcja zwraca FALSE.

Przykład 1

```php
<?php

$db = pg_connect("host=localhost port=54321 dbname=kazimirs".
                 " user=kazimirs password=student113");

$res = pg_query($db,"SELECT * FROM samochody;");

while($row=pg_fetch_row($res)) {
    echo $row[0],"  ";  // nr_s
    echo $row[1],"  ";  // marka
    echo $row[2],"\n";  // typ
};

?>
```

PostgreSQL - pg_fetch_assoc

- Zwraca rekord (wiersz) rezultatu zapytania w postaci tablicy asocjacyjnej

  array ***pg_fetch_assoc****(resource* ***rez*** *[, int* ***wiersz****])*

- Numer wiersza jest opcjonalny. Jeżeli nie podany to zwracany jest kolejny wiersz.
- W przypadku niepowodzenia funkcja zwraca FALSE.

Przykład 2

```php
<?php

$db = pg_connect("host=localhost port=54321 dbname=kazimirs".
                 " user=kazimirs password=student113");

$res = pg_query($db,"SELECT * FROM samochody;");

while($row=pg_fetch_assoc($res)) {
    echo $row["nr_s"],"  ";  // nr_s
    echo $row["marka"],"  ";  // marka
    echo $row["typ"],"\n";  // typ
};

?>
```

PostgreSQL - pg_fetch_object

- Zwraca rekord (wiersz) rezultatu zapytania w postaci obiektu (kolumny w postaci atrybutów)

  object ***pg_fetch_object***(*resource* ***rez*** *[, int* ***wiersz****])*

- Numer wiersza jest opcjonalny. Jeżeli nie podany to zwracany jest kolejny wiersz.
- W przypadku niepowodzenia funkcja zwraca FALSE.

Przykład 3

```
<?php

$db = pg_connect("host=localhost port=54321 dbname=kazimirs".
                 " user=kazimirs password=student113");

$res = pg_query($db,"SELECT * FROM samochody;");

while($row=pg_fetch_object($res)) {
    echo $row->nr_s,"  ";  // nr_s
    echo $row->marka,"  ";  // marka
    echo $row->typ,"\n";  // typ
};

?>
```

PostgreSQL – liczba zwróconych wierszy

- Aby określić liczbę wierszy w rezultacie zapytania można użyć funkcji pg_num_rows:

int ***pg_num_rows****(resource* ***rez****)*

- Funkcja ta ma sens tylko w przypadku zapytań zwracających rezultaty (SELECT)

PostgreSQL – liczba kolumn

- Do uzyskania informacji o liczbie kolumn w wyniku zapytania mozna użyć funkcji pg_num_fields

int ***pg_num_fields****(resource* ***rez****)*

- Funkcja ma sens tylko w przypadku zapytań zwracających rezultaty.

Przykład 4

```php
<?php

$db = pg_connect("host=localhost port=54321 dbname=kazimirs".
                 " user=kazimirs password=student113");

$res = pg_query($db,"SELECT marka,typ,kolor FROM samochody;");

while($row=pg_fetch_object($res)) {
    echo $row->marka,"  ";  // marka
    echo $row->typ,"  ";    // typ
    echo $row->kolor,"\n";  // kolor
};

echo "Zapytanie zwrocilo ".pg_num_rows($res)." wierszy.\n";
echo "Kazdy rekord sklada sie z ".pg_num_fields($res)." pol.\n";

?>
```

PostgreSQL – nazwy i typy kolumn

- Informacje o nazwach kolumn można uzyskać za pomocą funkcji pg_field_name:

  string ***pg_field_name****(resource* ***zap****,int* ***nr****)*

- Dodatkowo informację o typie kolumny można uzyskać za pomocą funkcji pg_field_type:

  string ***pg_field_type****(resource* ***zap****, int* ***nr****)*

Przykład 5

```php
<?php

$db = pg_connect("host=localhost port=54321 dbname=kazimirs".
                 " user=kazimirs password=student113");

$res = pg_query($db,"SELECT marka,typ,kolor FROM samochody;");

for($f=0;$f<pg_num_fields($res);$f++)
    echo pg_field_name($res,$f)," ";
echo "\n---------------------------\n";

while($row=pg_fetch_object($res)) {
    echo $row->marka,"  ";  // marka
    echo $row->typ,"  ";    // typ
    echo $row->kolor,"\n";  // kolor
};

?>
```

PostgreSQL – liczba modyfikowanych wierszy

- W zapytaniach typu INSERT, UPDATE, DELETE istotna jest liczba wierszy, które zostały zmodyfikowane (dodane/zmienione/usunięte). Można ją sprawdzić funkcją pg_affected_rows

*int **pg_affected_rows**(resource **zap**)*

- Jeżeli wykonanie zapytania się nie powiodło, funkcja zwróci wartość -1.

PostgreSQL- zwalnianie zasobów zapytania

- Funkcja pg_free_result usuwa rezultaty zapytania i zwalnia zasoby:

  *bool **pg_free_result**(resource **zap**)*

- Zwykle wywołanie tej funkcji nie jest konieczne – zasoby są zwalniane automatycznie w momencie zakończenia skryptu.

Podsumowanie – praca z bazą PostgreSQL

- Typowy schemat korzystania z bazy danych PostgreSQL:
    1. Otwarcie połączenia (pg_connect)
    2. Wykonanie zapytania (pg_query)
    3. Przetworzenie wyników zapytania (pq_fetch_row, pg_fetch_assoc, pg_fetch_object)
    4. [Zamknięcie połaczenia (pg_close)]

MySQL - połączenie

- Połączenie z bazą MySQL

  resource ***mysql_connect****(string* ***serwer****, string* ***user*** *string* ***hasło****, bool* ***nowe_poł****, int* ***opcje****)*
- Rezultatem wykonania funkcji jest uchwyt do bazy danych, lub FALSE w przypadku niepowodzenia.
- Zamykanie połączenia:

  bool ***mysql_close****(resource* ***uchwyt****)*

MySQL – wysyłanie zapytań

- Do przesyłania zapytań do bazy MySQL służy funkcja mysql_query:

  *resource **mysql_query**(string **zap**,resource **db**)*

- Jeżeli wykonywana jest kwerenda wybierająca (SELECT) to funkcja zwróci uchwyt do rezultatów lub FALSE (niepowodzenie). Dla innych typów zapytań funkcja zwraca TRUE lub FALSE

MySQL – pobieranie informacji

- Pobieranie informacji z rezultatu zapytania (uchwyt przekazany przez funkcje mysql_query):

  array ***mysql_fetch_row****(resource* ***zap****)*

  array ***mysql_fetch_assoc****(resource* ***zap****)*

  object ***mysql_fetch_object****(resource* ***zap****)*

MySQL – pobieranie informacji c.d.

- Funkcja mysql_data_seek pozwala przesunąć wewnętrzny indeks położenia rekordu w wynikach zwróconych z zapytania – pozwala pobrać dowolny rekord

  bool ***mysql_data_seek****(resource* ***zap****, int* ***nr****)*

- Funkcja zwraca TRUE (sukces) lub FALSE (niepowodzenie)

MySQL – inne funkcje

- Liczba zwróconych wierszy:
  int ***mysql_num_rows****(resource* ***zap****)*
- Liczba pól w zwróconych rekordach
  int ***mysql_num_fields****(resource* ***zap****)*
- Nazwa tabeli w której jest pole rekordu
  string ***mysql_field_table****(resource* ***zap****,int* ***nr****)*
- Nazwa pola rekordu danych
  string ***mysql_field_name****(resource* ***zap****,int* ***nr****)*

MySQL – inne funkcje c.d.

- Typ pola rekordu danych
  string ***mysql_field_type****(resource* ***zap****, int* ***nr****)*
- Właściwości pola rekordu danych
  object ***mysql_fetch_field****(resource* ***zap****, int* ***nr****)*
  - name – nazwa kolumny
  - table – nazwa tabeli
  - max_length – max. Liczba znaków w kolumnie
  - Inne właściwości: not_null, primary_key, unique_key, multiple_key, numeric, blob, type

MySQL – inne funkcje c.d.

- Określenie liczby zmodyfikowanych wierszy:

  int ***mysql_affected_rows****(resource* ***db****)*

- Zwolnienie zasobów

  bool ***mysql_free_result****(resource* ***zap****)*

PEAR DB

- PEAR::DB jest interfejsem API dostępu do baz danych
- Klasa DB jest częścią biblioteki PEAR
- Biblioteka ta musi być osobno zainstalowana (nie jest częścią PHP)
- PEAR::DB izoluje szczegóły bazy danych od reszty aplikacji

PEAR::DB - Źródło danych

- Przy nawiązywaniu połączenia wykorzystywany jest identyfikator źródła danych (DSN – Data Source Name)

  "*dbtype://user:password@host:port/database*"
  - dbtype – rodzaj bazy danych
  - user:password – użytkownik i hasło
  - host:port – nazwa hosta i numer portu
  - database – nazwa bazy danych

PEAR::DB - Połączenie z bazą danych

- W celu połączenia z bazą danych należy wykorzystać statyczną metodę connect, np:

```
$db =DB::connect($dsn);
```

- Rozłączenie z bazą danych:

```
$db->disconnect();
```

PEAR::DB - Wykonywanie zapytań

- Do realizacji zapytań służą metody:
  - ***query**(string **zap**)* – wysyła do serwera zapytanie nie zwracające rezultatów
  - ***getAll**(string **zap**)* – rezultat zapytania zwracany jest w 2-wymiarowej tablicy
  - ***getCol**(string **zap**)*, ***getRow**(string **zap**)* – dla rezultatów zawierających jedną kolumnę lub wiersz – wynik zwracany w tablicy
  - ***getOne**(string **zap**)* – dla zapytania zwracającego jedną liczbę

PEAR::DB - Inne metody

- Liczba zwróconych wierszy
  *$result->**numRows**()*
- Wynik operacji
  DB::isError($db*)*
  ***DB::isError**($result)*
- Komunikat o błędzie
  *$db->**getMessage**()*
  *$result->**getMessage**()*

PDO – PHP Data Objects

- Warstwa abstrakcji dostępu do baz danych wprowadzona do PHP 5.1
- W pełni obiektowa, wykorzystuje wyjątki do obsługi błędów
- Wykorzystuje zaawansowane techniki baz danych (transakcje, kwerendy parametryczne itp.)

PDO - Połączenie

- Nawiązywanie połączenia:

```
$db = new PDO($dsn,$user,$password)
```

- $dsn – data source name np.
  'mysql:host=localhost;dbname=test'
  'mysql:dbname=testdb;host=127.0.0.1'

- Aby zamknąć połączenie wystarczy przypisać do obiektu $db wartość null.

PDO – Połączenie c.d.

- Za pomocą parametru połączenia PDO::ATTR_PERSISTENT można uzyskać stałe połączenie z bazą
- Stałe połączenie z bazą nie jest zamykane po zakończeniu skryptu
- Zastosowanie stałych połączeń pozwala na efektywniejsze działanie serwera (brak narzutu na otwieranie i zamykanie połączeń)

PDO - Transakcje

- PDO udostępnia obsługę transakcji (dla baz, które je posiadają) za pomocą metod:

  PDO->**beginTransaction**()

  PDO->**commit**()

  PDO->**rollBack**()

PDO – Przekazywanie zapytań do bazy

- PDO->**exec**($zap) – przekazuje do bazy zapytanie i zwraca liczbę zmodyfikowanych wierszy
- PDO->**query**($zap) – przekazuje do bazy zapytanie i zwraca rezultat w postaci obiektu typu PDOStatement.
- Rezultaty zawarte w obiekcie PDOStatement mogą być uzyskane za pomocą jego metod **fetch**() i **fetchAll**().

PDO – przygotowywanie zapytań

- Wielokrotnie używane zapytanie można przygotować za pomocą metody PDO->**prepare**() i symboli zastępczych
- Raz przygotowane zapytanie można wykonywać wielokrotnie za pomocą jego metody **execute**()
- Metoda **execute**() ma argument w postaci tablicy – kolejne elementy tablicy zastepują kolejne symbole zastępcze.

PDO – Przykład 1

```
<?php

$db = new PDO('pgsql:host=localhost;dbname=jankazim',
                "jankazim", "secret");

$res = $db->query("SELECT * FROM samochody");

$table = $res->fetchAll();

var_dump($table);

?>
```

PDO – Przykład 2

```php
<?php

$db = new PDO('pgsql:host=localhost;dbname=jankazim',
                "jankazim", "secret");

$query = $db->prepare("SELECT * FROM samochody");
$query->execute();

$res = $query->fetchAll();

var_dump($res);

?>
```

PDO – Przykład 3

```php
<?php

$db = new PDO('pgsql:host=localhost;dbname=jankazim',
                "jankazim", "secret");

$query = $db->prepare("SELECT * FROM samochody where kolor= ?");
$query->execute(array("czerwony"));

$res = $query->fetchAll();

var_dump($res);

?>
```

PDO – Przykład 4

```php
<?php

$db = new PDO('pgsql:host=localhost;dbname=jankazim',
                "jankazim", "secret");

$query = $db->prepare("SELECT * FROM samochody where kolor =:kolor");

$kolor = "czerwony";
$query->bindParam(":kolor",$kolor);

$query->execute();

$res = $query->fetchAll();

var_dump($res);

?>
```

PDO – Przykład 5

```php
<?php

try {
  $db = new PDO('pgsql:host=localhost;dbname=jankazim',"jankazim", "secret");
} catch (PDOException $e) {
    print "Blad!: " . $e->getMessage() . "\n";
    die();
}

$res = $db->query("SELECT * FROM samochody");
$table = $res->fetchAll();
var_dump($table);

?>
```

Podsumowanie

- Zagadnienia poruszane na wykładzie:
  - Dostęp do bazy PostgreSQL ze skryptu PHP
  - Dostęp do bazy MySQL ze skryptu PHP
  - Wykorzystanie biblioteki PEAR DB do obsługi baz danych z PHP
  - Wykorzystanie mechanizmu PDO do obsługi baz danych z PHP