Kurs WWW

Paweł Rajba

pawel@ii.uni.wroc.pl
http://pawel.ii.uni.wroc.pl/

Spis treści

Korzystanie z bazy PostgreSQL

- Połączenie z bazą danych
  - Składnia funkcji
    - pg_connect($connstring, [$typ_polaczenia])
  - Opcje
    - Akceptowane opcje w $connstring
      host, hostaddr, port, dbname, user, password,
      connect_timeout, options, sslmode, service
      - Przykład łańcucha
        "host=sheep dbname=mary user=lamb password=foo"
    - Jeśli jako typ połączenia podamy
      PGSQL_CONNECT_FORCE_NEW,
      za każdym razem będzie tworzone nowe połączenie

Korzystanie z bazy PostgreSQL

- Wykonanie zapytania ver. 1
  - Składnia funkcji
    - pg_query(string $connection, string $query)
  - Opcje
    - $connection – wynik zwrócony przez pg_connect()
    - $query – zapytanie
  - Komentarz
    - $connection można pominąć, ale nie będziemy tak robić

Korzystanie z bazy PostgreSQL

- Wykonanie zapytania ver. 2
  - Składnia funkcji
    - pg_query_params(string $connection, string $query, array $params)
  - Opcje
    - $connection – wynik zwrócony przez pg_connect()
    - $query – zapytanie z parametrami: $1, $2, ...
    - $params – wartości parametrów (liczba musi się zgadzać)
  - Komentarz
    - $connection można pominąć, ale nie będziemy tak robić

Korzystanie z bazy PostgreSQL

- Wykonanie zapytania ver. 3
  - Przygotowanie zapytania
    - pg_prepare(resource $connection, string $stmtname, string $query)
      - parametry to $1, $2, ...
  - Wykonanie zapytania
    - pg_execute(resource $connection, string $stmtname, array $params)
      - $params – tablica z wartościami parametrów

Korzystanie z bazy PostgreSQL

- Pobieranie błędów po wykonaniu zapytania
  - pg_last_error($connection)
  - pg_last_notice ($connection)
- Pobieranie liczby wierszy wyniku
  - pg_num_rows($result)
    - Zapytania typu SELECT
  - pg_affected_rows($result)
    - Zapytania typu INSERT, UPDATE, DELETE
  - pg_num_fields ($result)
    - Zwraca liczbę wierszy wyniku

Korzystanie z bazy PostgreSQL

- Pobieranie wierszy wyniku
  - pg_fetch_array (resource $result, int $row [, int $result_type])
    - Zwraca pojedynczy wiersz jako tablicę
    - $result_type przyjmuje wartości: PGSQL_ASSOC, PGSQL_NUM i PGSQL_BOTH (domyślnie BOTH)
      - W przypadku PGSQL_NUM kolumny są numerowane od 0
  - pg_fetch_assoc (resource $result [, int $row])
    - Podobne do pg_fetch_array z parametrem PGSQL_ASSOC
  - pg_fetch_row (resource $result, int $row)
    - Podobne do pg_fetch_array z parametrem PGSQL_NUM

Korzystanie z bazy PostgreSQL

- Pobieranie wierszy wyniku
  - array pg_fetch_all (resource $result)
    - Wartości null będą miały
- Kilka innych funkcji
  - int pg_field_is_null (resource $result, int $row, mixed $field)
    - nry wierszy od 0, kolumna jako indeks lub nazwa
  - bool pg_free_result (resource $result)
- Zamykanie połączenia
  - bool pg_close ([resource $połączenie])

Korzystanie z bazy PostgreSQL

- Inny sposób wykonywania zapytań
  - Dodawanie rekordów
    - bool pg_insert (resource $connection, string $table_name, array $assoc_array)
  - Usuwanie rekordów
    - long pg_delete (resource $connection, string $table_name, array $assoc_array)
  - Wybieranie rekordów
    - array pg_select (resource $connection, string $table_name, array $assoc_array

Korzystanie z bazy PostgreSQL

- Przykłady
  - postgres1.php
  - postgres2.php
  - postgres3.php
  - postgres4.php

Korzystanie z bazy MySQL

- Mamy dostępne dwa interfejsy do MySQLa
  - MySQL (inaczej mysql)
  - Improved MySQL (inaczej mysqli)
- Zmiany w wersji improved
  - Zgodność z MySQL 4.1 i wyższe
  - Instrukcje preparowane i parametry dowiązywane
  - Interfejs zorientowany obiektowo
  - Połączenia zabezpieczone SSLem

Korzystanie z bazy MySQL

- Połączenie z bazą danych
  - Wersja proceduralna
    - mysqli mysqli_connect ([string $host [, string $username [, string $passwd [, string $dbname [, int $port]]]]])
  - Wersja obiektowa
    - class mysqli { __construct ([string $host [, string $username [, string $passwd [, string $dbname [, int $port]]]]]) }
- Pobieranie błędów połączenia
  - int mysqli_connect_errno (void)
  - string mysqli_connect_error (void)

Korzystanie z bazy MySQL

- Wykonanie zapytania ver. 1
  - Wersja proceduralna
    - mixed mysqli_query (mysqli $link, string $query [, int $resultmode])
  - Wersja obiektowa
    - class mysqli {
      mixed query (string $query [, int $resultmode]) }
  - Parametr $resultmode może mieć wartości
    - MYSQLI_USE_RESULT
    - MYSQLI_STORE_RESULT (domyślnie)
  - Znaczenie parametru $resultmode

Korzystanie z bazy MySQL

- Pobieranie błędów po wykonaniu zapytania ver. 1
  - int mysqli_errno (mysqli $link)
  - class mysqli { int errno }
  - string mysqli_error (mysqli $link)
  - class mysqli { string error }

Korzystanie z bazy MySQL

- Pobieranie wyników ver. 1
  - mixed mysqli_fetch_array (mysqli_result $result [, int $resulttype])
    - Parametr $resulttype może przyjmować wartości
      - MYSQLI_ASSOC
      - MYSQLI_NUM
      - MYSQLI_BOTH (domyślnie)
  - mixed mysqli_fetch_row (mysqli_result $result) class mysqli_result { mixed fetch_row (void) }
  - array mysqli_fetch_assoc (mysqli_result $result) class mysqli_result { array fetch_assoc (void) }

Korzystanie z bazy MySQL

- Wykonanie zapytania ver. 2
  - Utworzenie sparametryzowanego zapytania
    - Parametry przykazujemy przez znak ?
  - Tworzymy pusty uchwyt do przygotowanego zapyt.
    - mysqli_stmt mysqli_stmt_init (mysqli $link)
    - class mysqli { mysqli_stmt stmt_init (void) }
  - Przygotowania zapytania
    - bool mysqli_stmt_prepare (mysqli_stmt $stmt, string $query)
    - class mysqli_stmt { mixed prepare (string $query) }

Korzystanie z bazy MySQL

- Wykonanie zapytania ver. 2
  - Wiązanie parametrów wejściowych
    - bool mysqli_stmt_bind_param (mysqli_stmt $stmt, string $types, mixed &$var1 [, mixed &$...])
    - class mysqli_stmt {
      bool bind_param (string $types, mixed &$var1 [, mixed &$...]) }
      - Parametr types określa typy kolejnych parametrów
      - Typy mogą być następujące
        - i – liczba całkowita
        - d – liczba zmiennoprzecinkowa
        - s – łańcuch znaków
        - b – ,,blob"

Korzystanie z bazy MySQL

- Wykonanie zapytania ver. 2
  - Wiązanie parametrów wyjściowych
    - bool mysqli_stmt_bind_result (mysqli_stmt $stmt, mixed &$var1 [, mixed &$...])
    - class mysqli_stmt { bool bind_result (mixed &$var1 [, mixed &$...]) }
  - Wykonywanie przygotowanego zapytania
    - bool mysqli_stmt_execute (mysqli_stmt $stmt)
    - class mysqli_stmt { bool execute (void) }

Korzystanie z bazy MySQL

- Pobranie wyników zapytania ver. 2
  - bool mysqli_stmt_fetch (mysqli_stmt $stmt)
  - class mysqli_stmt { bool fetch (void) }
    - Wyniki będą w odpowiednio związanych zmiennych
- Pobieranie błędów po wykonaniu zapytania ver. 2
  - int mysqli_stmt_errno (mysqli_stmt $stmt)
    class mysqli_stmt { int errno }
  - string mysqli_stmt_error (mysqli_stmt $stmt)
    class mysqli_stmt { string error }

Korzystanie z bazy MySQL

- Ustalenie liczby zmodyfikowanych wierszy
  - Dla poleceń INSERT, UPDATE i DELETE
    - int mysqli_stmt_affected_rows (mysqli_stmt $stmt)
      class mysqli_stmt { int affected_rows }
  - Dla poleceń SELECT
    - int mysqli_stmt_num_rows (mysqli_stmt $stmt)
      class mysqli_stmt { int num_rows }
- Zwalnianie zasobów
  - void mysqli_stmt_free_result (mysqli_stmt $stmt)
    class mysqli_stmt { void free_result (void) }

Korzystanie z bazy MySQL

- Pobranie liczby parametrów
  - int mysqli_stmt_param_count (mysqli_stmt $stmt)
    class mysqli_stmt { int param_count }
- Zamknięce przygotowanego zapytania
  - bool mysqli_stmt_close (mysqli_stmt $stmt)
    class mysqli_stmt { bool close (void) }
- Zamknięcie połączenia do bazy danych
  - bool mysqli_close (mysqli $link)
    class mysqli { bool close (void) }

Korzystanie z bazy MySQL

- Przykłady
  - mysql1.php
  - mysql2.php
  - mysql3.php
  - mysql4.php

Warstwa abstrakcji

- Co to jest warstwa abstrakcji dla bazy danych i po co to robić?
  - Przenośność pomiędzy różnymi serwerami bd
  - Problem z różnicami pomiędzy serwerami SQL
  - Łatwiejsza możliwość modyfikacji fragmentów kodu dotyczących warstwy bd

Warstwa abstrakcji

- Propozycje warstw abstrakcji
  - database1.php
  - database2.php
  - database3.php

PHP Data Objects (PDO)

- Wprowadzenie
  - PDO jest implementacji abstrakcji w dostępie do baz danych
  - Dostępne od wersji 5
    - wymaga nowego interfejsu obiektowego
  - Zwykle trzeba doinstalować odpowiednie moduły
  - Do każdej bazy danych jest używany odpowiedni sterownik
  - Dostępne sterowniki: PDO_*
    - DBLIB, FIREBIRD, IBM, INFORMIX, MYSQL, OCI, ODBC, PGSQL, SQLITE

PHP Data Objects (PDO)

- Połączenie z bazą danych
  - Następuje w momencie utworzenia obiektu
  - Składnia konstruktora
    - __construct($dsn, $user, $password [, $drvoptions])
  - Opcje
    - $dsn – określenie bazy danych
    - $user – użytkownik
    - $password – hasło
    - $drvoptions – dodatkowe opcje steronika
      - przekazywane poprzez tablicę z ustawionymi kluczami i wartościami, np. array(PDO::ATTR_PERSISTENT => true)

PHP Data Objects (PDO)

- Połączenie z bazą danych
  - Parametr DSN może przyjmować jedną z trzech postaci:
    - Jawne wskazanie bazy danych
      - np. 'mysql:host=localhost;dbname=test'
    - Wskazanie pliku z parametrami
      - np. 'uri:file:///usr/local/dbconnect'
    - Podanie nazwy aliasu zdefiniowanego w pliku php.ini
      - po prostu nazwa aliasu + definicja w pliku php.ini
- Zamknięcie połączenia
  - Przypisanie obiektowi wartości null

PHP Data Objects (PDO)

- Atrybuty połączenia
  - Ustawianie
    - PDO->setAttribute(int $attribute, mixed $value)
  - Pobieranie
    - PDO->getAttribute(int $attribute)
- Dostępne atrybuty
  - **SET: PDO::ATTR_CASE, PDO::ATTR_ERRMODE, PDO::ATTR_ORACLE_NULLS, PDO::ATTR_STRINGIFY_FETCHES, PDO::ATTR_STATEMENT_CLASS, PDO::ATTR_AUTOCOMMIT, PDO::MYSQL_ATTR_USE_BUFFERED_QUERY**
  - ***GET: PDO::ATTR_AUTOCOMMIT PDO::ATTR_CASE PDO::ATTR_CLIENT_VERSION PDO::ATTR_CONNECTION_STATUS PDO::ATTR_DRIVER_NAME PDO::ATTR_ERRMODE PDO::ATTR_ORACLE_NULLS PDO::ATTR_PERSISTENT PDO::ATTR_PREFETCH PDO::ATTR_SERVER_INFO PDO::ATTR_SERVER_VERSION PDO::ATTR_TIMEOUT***

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 1
  - Dla zapytań typu INSERT, UPDATE, DELETE
    - PDO->exec(string $zapytanie)
  - Dla zapytań typu SELECT
    - PDO->query(string $zapytanie)
    - PDO->query (string $statement,
      int $PDO::FETCH_CLASS, string $classname)
    - PDO->query (string $statement,
      int $PDO::FETCH_INTO, object $object)

PHP Data Objects (PDO)

- Pobranie ostatnio wstawionego id
  - string lastInsertId ([string $name])
  - zależy sporo od sterownika, np. w postgresie jako name podaje się nazwę sekwencji
- Transakcje
  - bool PDO->beginTransaction()
  - bool PDO->commit()
  - bool PDO->rollBack()

PHP Data Objects (PDO)

- Pobranie błędów
  - string PDO->errorCode()
    - Zwrócony zostanie SQLSTATE, czyli kod 5-znakowy zgodny ze standardem ANSI SQL
  - array PDO->errorInfo()
    - Zwrócona tablica zawiera trzy wartości
      - 0 => kod 5-znakowy zgodny ze standardem ANSI SQL
      - 1 => kod błędu właściwy dla sterownika
      - 2 => komunikat błędu właściwy dla sterownika

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2
  - Tworzymy kod zapytania
    - W miejsce parametrów dajemy
      - Znak ?
      - Nazwę parametru :param
  - Wykonujemy polecenie
    - PDOStatement PDO->prepare(string $statement [, array $driver_options])
    - Najczęstsze użyciem parametru $driver_options to ustawienie parametru PDO::ATTR_CURSOR na wartości
      - PDO::CURSOR_SCROLL lub
      - PDO::CURSOR_FWDONLY

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Wiążemy parametry w zapytaniu
    - bool PDOStatement->bindParam (mixed $parameter, mixed &$variable [, int $data_type [, int $length [, mixed $driver_options]]])
      - przekazywanie parametrów przez referencję
      - istotne przy dużych parametrach i parametrach typu INOUT
    - bool PDOStatement->bindValue (mixed $parameter, mixed $value [, int $data_type])
      - przekazywanie parametrów przez wartość

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Typy danych mogą być następujące
    - PDO::PARAM_BOOL
      - odpowiada typu boolean w SQL
    - PDO::PARAM_NULL
      - odpowiada typowi null w SQL
    - PDO::PARAM_INT
      - odpowiada typowi integer w SQL
    - PDO::PARAM_STR
      - odpowiada typom char i varchar w SQL
    - PDO::PARAM_LOB
      - odpowiada typowi lob i blob w SQL

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Typy danych c.d.
    - PDO::PARAM_STMT
      - odpowiada typowi reprezentującemu w SQL-u zbiór danych, np. kursory
      - aktualnie nie wspierany przez żadne sterowniki
    - PDO::PARAM_INPUT_OUTPUT
      - określa, że w procedurze parametr jest typu INOUT
      - w użyciu łączy się poprzez operator (|) w z innym typem danych PDO::PARAM_*

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Wykonanie zapytania
    - bool PDOStatement->execute ([array $input_parameters])
  - Przekazywanie parametrów
    - Albo przez wcześniejsze wiązanie
    - Albo przez parametr $input_parameters
      - execute(array($calories, $colour))
        - w przypadku parametrów pozycyjnych
      - execute(array(':calories' => $calories, ':colour' => $colour));
        - w przypadku parametrów nazwanych

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Przechwytywanie błędów
    - string PDOStatement->errorCode()
      - Analogiczne do PDO->errorCode()
    - array PDOStatement->errorInfo()
      - Analogiczne do PDO->errorInfo()

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Pobieranie wyników
    - mixed PDOStatement->fetch([int $fetch_style [, int $cursor_orientation [, int $cursor_offset]]])
  - Opcje
    - Parametr $fetch_style może przymować wartości: **PDO::FETCH_ASSOC, PDO::FETCH_BOTH, PDO::FETCH_NUM, PDO::FETCH_OBJ**
    - Parametr $cursor_orientation może przyjmować wartości: **PDO::FETCH_ORI_NEXT, PDO::FETCH_ORI_PRIOR, PDO::FETCH_ORI_FIRST, PDO::FETCH_ORI_LAST, PDO::FETCH_ORI_ABS, PDO::FETCH_ORI_REL**
    - Ostatni parametr ma znaczenie wraz niektórymi wartościami poprzedniego parametru

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Pobieranie wyników (całej tabeli)
    - array PDOStatement->fetchAll ()
  - Pobieranie wyników przez wiązanie parametrów
    - bool PDOStatement->bindColumn(mixed $column, mixed &$param [, int $type])
      - $column – określa kolumnę na liście SELECT
      - $param – określa zmienną, do której będą przekazywane wyniki
      - $type – określa typ wiązanego parametru (PDO::PARAM_*)
    - Używa się w połączeniu z funkcją PDOStatement->fetch()

PHP Data Objects (PDO)

- Wykonywanie zapytań ver. 2 c.d.
  - Ustawienie rodzaju pobierania (przed pobieraniem)
    - bool PDOStatement->setFetchMode (int $mode)
      - Parametr $mode przyjmuje wartości takie jak parametr $fetch_style z funkcji PDOStatement->fetch()
  - Zwalnianie zasobów związanych ze ,,statement"
    - bool PDOStatement->closeCursor()

PHP Data Objects (PDO)

- Przykłady
  - pdo_mysql1.php
  - pdo_mysql2.php
  - pdo_mysql3.php
  - pdo_mysql4.php
  - pdo_mysql5.php

PEAR DB

- Wprowadzenie
  - Jeden z komponentów obsługi baz danych w ramach PEAR Database
  - Inny interfejs obsługi baz danych
  - W ramach pakietu PEAR dostępne są też inne komponenty do obsługi baz danych
    - DB_DataObject, DB_DataObject_FormBuilder
    - DB_Pager, DB_QueryTool, DB_Table
    - MDB, DB2, MDB_QueryTool
  - Dokumentacja
    - http://pear.php.net/manual/en/package.database.php

PEAR DB

- Wprowadzenie
  - Obsługiwane rodzaje baz danych:
    - dBase (dbase), FrontBase (fbsql), InterBase (ibase), Informix (ifx), Mini SQL (msql), Microsoft SQL Server (mssql), MySQL <= 4.0 (mysql), MySQL >= 4.1 (mysqli), Oracle 7/8/9 (oci8), ODBC (odbc), PostgreSQL (pgsql), SQLite (sqlite), Sybase (sybase)
- Instalacja
  - Pobranie i rozpakowanie pakietów PEAR i DB
  - Wstawienie w pliku php.ini wpisu
    - include_path="ścieżka do DB.php"

PEAR DB

- Połączenie z bazą danych
  - Odbywa się poprzez konstrukcję
    - $conn = DB::connect($dsn)
  - Parametr $dsn może być napisem lub tablicą
  - Format DSN-a
    - Napis:
      [typ bazy]://[użytkownik]:[hasło]@[serwer]/[nazwa bazy]
    - Tablica z elementami o kluczach: phptype, dbsyntax, username, password, hostspec, database, port

PEAR DB

- Połączenie z bazą danych
  - Przykładowe DSN-y
    - mysql://root@localhost/test
    - pgsql://postgres:pgsql812@localhost/postgres
    - array('phptype' => 'pgsql', 'username' => 'postgres', 'password' => 'pgsql812', 'hostspec' => 'localhost', 'database' => 'postgres',)
    - array('phptype' => 'sqlite', 'database' => 'thedb', 'mode' => '0644',);
- Rozłączenie z bazą danych
  - Odbywa się poprzez następujące wywołanie funkcji
    - $conn->disconnect()

PEAR DB

- Przechwytywanie błędów
  - Kontrukcja DB::isError($conn) określa, czy wynik (zmienna) jest błędem
  - Jeśli jest, możemy pobrać
    - Komunikat błędu
      - $conn->getMessage()
    - Kod błędu
      - $conn->getCode()
    - Raporty bezpośrednio z systemu DBMS
      - db->getUserInfo()
      - db->getDebugInfo()

PEAR DB

- Wykonywanie zapytań ver. 1
  - Służy do tego funkcja query(), która zwraca
    - obiekt klasy DB_result w przypadku zapytań SELECT
    - stałą DB_OK w przypadku zapytań modyfikujących dane
    - DB_ERROR w przypadku błędu
  - Składnia
    - mixed &query (string $query [, mixed $params = array()])
  - Parametry
    - $query – zapytanie, w którym można wstawiać parametry
    - $params – tablica parametrów, które z mapują się na kolejne znaki "?" w $query

PEAR DB

- Wykonywanie zapytań ver. 2
  - Najpierw zapytanie jest przygotowywane
    - resource prepare (string $query)
      - Zmienna $query może mieć parametry (znak ?)
  - Następnie wykonywane
    - mixed &execute (resource $stmt [, mixed $data=array()])
      - Drugi parametr to tablica wartości do zapytania
      - Zwraca DB_result, DB_OK lub DB_Error
    - integer executeMultiple (resource $stmt, array $data)
      - Drugi parametr to tablica dwuwymiarowa wartości do zapytania
      - Wykonuje zapytanie wielokrotnie dla kolejnych elementów $data
      - Zwraca DB_OK lub DB_Error

PEAR DB

- Ustawienie domyślnego trybu pobierania danych
  - Odnosi się do metod fetch*() i get*()
  - Polecenie
    - void setFetchMode (integer $fetchmode
      [, string $object_class = stdClass])
  - Parametry
    - Parametr $fetchmode może przyjmować wartości:
      - DB_FETCHMODE_ORDERED
      - DB_FETCHMODE_ASSOC
      - DB_FETCHMODE_OBJECT
    - Parametr $object_class ma sens przy ustawieniu poprzedniego parametru na DB_FETCH_OBJECT

PEAR DB

- Pobieranie wierszy z wyniku typu DB_result
  - Służą do tego dwie metody
    - mixed &fetchRow ([integer $fetchmode = DB_DEFAULT_MODE [, integer $rownum = NULL]])
      - Zwraca wiersz danych
    - integer fetchInto (array &$arr [, integer $fetchMode = DB_FETCHMODE_DEFAULT [, integer $rowNum = NULL]])
      - Wiersz danych jest wprowadzany do pierwszego parametru
  - Tryb pobierania można ustawić na wartości
    - DB_FETCHMODE_(ORDERED|ASSOC|OBJECT)
  - Ostatni parametr określa numer wiersza do pobrania (numerowanie od 0)

PEAR DB

- Pobieranie danych bezpośrednio z DB_common
  - Mamy do tego następujące metody
    - array &getAll (string $query [, array $params = array() [, integer $fetchmode = DB_FETCHMODE_DEFAULT]])
      - Znaczenie parametrów takie samo jak tych w metodzie fetch()
    - array &getAssoc (string $query [, boolean $force_array = FALSE [, mixed $params = array() [, integer $fetchmode = DB_FETCHMODE_DEFAULT]]])
    - mixed &limitQuery (string $query, integer $from, integer $count [, mixed $params = array()])
      - Podobne do getAll(), ale
        - pierwszym rekordem jest $from (numeracja od 0)
        - liczba zwróconych rekordów jest określona przez $count

PEAR DB

- Pobieranie danych bezpośrednio z DB_common
  - Mamy do tego następujące metody c.d.
    - mixed &getOne (string $query [, mixed $params = array()])
      - Zwrócona zostaje pierwsza kolumna pierwszego wiersza wyniku, a następnie pamięć dla wyniku zostaje zwolniona
    - array &getRow (string $query [, array $params = array() [, integer $fetchmode = DB_FETCHMODE_DEFAULT]])
      - Można przekazywać do $query parametry

PEAR DB

- Automatyczne generowanie zapytań
  - Stosuje się do zapytań typu INSERT i UPDATE
  - Najpierw jest przygotowanie zapytania:
    - resource autoPrepare (string $table, array $table_fields, integer $mode = DB_AUTOQUERY_INSERT [, string $where = FALSE])
    - Znaczenie parametrów
      - $table, $table_fields – nazwa tabelki, lista pól objętych operacją
      - $mode – dwie akceptowane wartości:
        - DB_AUTOQUERY_INSERT or DB_AUTOQUERY_UPDATE
      - $where – określa warunek dla zapytania typu UPDATE
  - Potem można wykonać poprzez funkcje execute() lub executeMultiple()

PEAR DB

- Automatyczne generowanie i wykonanie zapytań
  - Stosuje się do zapytań typu INSERT i UPDATE
  - Mamy do tego funkcję
    - integer autoExecute (string $table, array $fields_values [, integer $mode = DB_AUTOQUERY_INSERT [, string $where = FALSE]])
    - Znaczenie parametrów takie samo jak tych w funkcji autoPrepare()
  - Działanie funkcji podobne do działania funkcji autoPrepare(), przy czym zapytanie jest od razu wykonywane

PEAR DB

- Sekwencje
  - Przydatne przy wstawianiu wierszy do tabel
  - Ułatwia panowanie nad identyfikatorami wstawianych rekordów
  - Tworzenie sekwencji
    - integer createSequence (string $seq_name)
    - Zwracane wartości to DB_OK lub DB_Error
  - Usuwanie sekwencji
    - integer dropSequence (string $seqName)
    - Zwracane wartości to DB_OK lub DB_Error

PEAR DB

- Sekwencje
  - Pobieranie wartości z sekwencji
    - integer nextId (string $seq_name,
      boolean $onDemand = TRUE)
    - Drugi parametr ustawiony na TRUE powoduje automatyczne utworzenie sekwencji w przypadku, gdy sekwencja nie istnieje
- Transakcje
  - mixed autoCommit ([boolean $onoff = FALSE])
  - mixed commit ()
  - mixed rollback ()

PEAR DB

- Inne przydatne funkcje
  - integer DB_result::numCols ()
    - Zwraca liczbę kolumn w wyniku
  - integer DB_result::numRows ()
    - Zwraca liczbę wierszy wyniku
  - mixed DB_common::quoteSmart (mixed $in)
    - Podobne do np. pg_quote_string() z PostgreSQL-a
  - integer DB_common::affectedRows ()
    - Zwraca liczbę ,,zmodyfikowanych" wierszy przez ostatnio wykonaną funkcję query()
    - Zwraca 0 dla zapytań typu SELECT

PEAR DB

- Przykłady
  - pear1.php
  - pear2.php
  - pear3.php

