MySQL and PHP Basics

Data Manipulation with SQL, Working with PHP

greenwich.edu.vn

UNIVERSITY of GREENWICH
Alliance with FPT Education

Table of Contents

CONNECTING DATABASE FROM PHP

UNIVERSITY of GREENWICH
Alliance with FPT Education

PHP and Databases

- PHP supports about 20 RDBM servers
  - Including MySQL, Oracle, MS SQL, DB2, Firebird and Paradox
  - Supports connection over ODBC driver
  - Provided different sets of functions for accessing the different RDBMS
    - Each function starts with prefix – the DB server type
      Example: `mysql_connect`, `mssql_query`, etc

UNIVERSITY of GREENWICH
Alliance with FPT Education

Connecting MySQL

- `mysql_connect` – function to connect to MySQL server
  - Parameters: `$server, $username, $password, $new_link, $client_flags`
  - Returns resource result, identifying the new link (link identifier)
    - The result is used as parameter to other `mysql_` functions

```
mysql_connect("localhost", "user", "userpassword");
```

Connecting MySQL{2}

- Once connected a database must be selected to perform queries upon
  - In some cases it is not required – `show databases` query for instance
  - `mysql_select_db ($dbname, $link)` – selects database from the server
    - Returns true if successful

```
$dblink = mysql_connect("local host", "user", "userpassword");

mysql_select_db("mydb", $dblink);
```

SENDING QUERY

Executing Query

- `mysql_query ($query, $link)` – execute query on database
  - `$query` is string – the query to be executed
  - `$link` is database link identifier
  - The returned result depends on the query
    - If query is select, show, describe, explain – returns resource or false on error
    - Otherwise true if successful, false on error

```
mysql_query("select * from users", $dblink);
```

  - The link parameter can be omitted in all `mysql_` functions if working with only one database
    - Only one call to `msql_connect` in the script

UNIVERSITY of GREENWICH
Alliance with FPT Education

Select Query Results

- PHP provides several functions for working with MySQL select query results
  - `mysql_query` returns resource when performing select query that holds the data
  - The result is accessed row-per-row from first towards last with internal pointer
- Additional functions to get number of affected rows on update/delete or auto-generated id of inserted row

FETCHING DATA

UNIVERSITY of GREENWICH
Alliance with FPT Education

Fetch Row From Result

- `mysql_fetch_row` – returns numerical array, containing the current row from the result and moves the pointer to the next row
  - Returns false if there are no more rows

```
$res = mysql_query ("select id, name from people");
$row = mysql_fetch_row($res);
if ($row)
        print_r($row); // 0->id, 1->name
else
        echo "No results!";
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Fetching Row From Result (2)

- `mysql_fetch_assoc` – returns associative array containing the current row in result and moved the pointer to the next one
  - The field names are keys in the array
  - Returns false if no more rows

```
$res = mysql_query ("select id, name from people");
$row = mysql_fetch_assoc($res);
if ($row)
        echo "Name: ".$row['name'];
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Fetching Single Value

- `mysql_result ($result, $row, $field)` – return the value or single cell In MySQL query result
  - `$field` is either field index or name
  - Returns false on failure
  - Must NOT be mixed with other functions for reading query result
  - Much slower than fetching data row-per-row

```
$res = mysql_query ("select count(*) from people");
echo mysql_result($res, 0, 0);
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Number of Rows

- `mysql_num_rows ($result)` – returns the number of rows in the result set
  - Does not work with unbuffered queries (`mysql_unbuffered_query`)

```
$res = mysql_query ("select id, name from people");
$count = mysql_num_rows($res);
echo $count;
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Executed Query Result

- `mysql_insert_id($link)` – get the auto generated ID of previous insert/replace query
  - Returns 0 if no ID was generated, false on error
  - Works only for `AUTO_INCREMENT` columns
  - `$link` can be omitted if only one link established

```
mysql_query ("insert into people ("name", "age") values
("To6ko", "30");
echo mysql_insert_id();
```

Executed Query Result (2)

- `mysql_affected_rows($link)` – returns number of affected rows in most recent insert/update/delete/replace query
  - As with all `mysql_` functions `$link` can be omitted if only one link established
  - Returns -1 if last query failed

```
mysql_query ("update people set age+1 where age < 20");
echo mysql_insert_id();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Error Handling

- `mysql_errno ($link)` - returns the error code from the last query
  - Returns 0 if no error occurred
- `mysql_error ($link)` – returns the error text from the last query
  - Returns empty string if no error occurred

```
mysql_query ("insert into nosuchtable");
echo mysql_errno().": ".mysql_error();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Closing and Freeing

- `mysql_free_result($resource)` – clears the memory occupied by select query result
- `mysql_close($link)` – closes connection to mysql server
- When PHP script ends all resources are freed automatically and all connections – closed
  - Freeing is not necessary
  - Closing is needed only when using persistent connections

PERSISTENT CONNECTIONS

UNIVERSITY of GREENWICH
Alliance with FPT Education

Persistent Connections

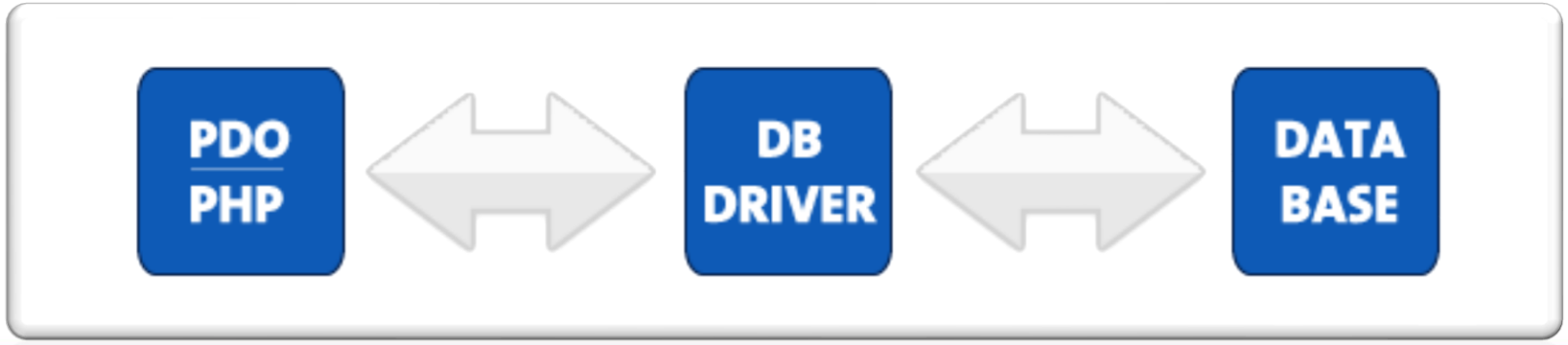
- Persistent connections are connections that are kept open after script ends
  - Allows reusing
  - Saves time for next script to connect
  - Very useful for slow-login databases (MS SQL, Firebird, etc)
  - When performing persistent connect PHP searches for already opened connection and reuses it
- `mysql_pconnect` – similar to `mysql_connect` but checks for previous persistent connection with same parameters and reuses it

PDO PREPARED STATEMENTS

UNIVERSITY of GREENWICH
Alliance with FPT Education

PDO: What?

- PDO - PHP Data Object
- An interface for accessing databases in PHP
- Provides a data-access abstraction layer
  - Same functions are used across DBMSs
- Object-oriented



UNIVERSITY of GREENWICH
Alliance with FPT Education

PDO: Why?

- Portability
- Speed
  - Pre-compiled SQL statements that accept zero or more parameters
- Security
  - Supports prepared statements
    - Prevents SQL injection by using placeholders for data

UNIVERSITY of GREENWICH
Alliance with FPT Education

Get connected

- This creates a database object called $dbh

```
try {
    //Instantiate a database object
    $dbh = new PDO("mysql:host=$hostname;
         dbname=myDB", $username, $password);
    echo 'Connected to database';
}
catch(PDOException $e) {
     echo $e->getMessage();
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Steps

- Define the query
  - `$sql = "…";`
- Execute the query
  - Queries that don't return a result set (e.g. INSERT)
    - `$dbh->exec($sql);`
  - Queries that do return a result set (SELECT)
    - `$dbh->query($sql);`
- Process the result

UNIVERSITY of GREENWICH
Alliance with FPT Education

A SELECT Query: Multiple Rows

- query() returns a result set

```
//Define the query
$sql = "SELECT * FROM animals";

//execute the query
$result = $dbh->query($sql);

//process the result
foreach($result as $row) {
        print $row['animal_type'] .
        ' - '. $row['animal_name'];
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

A SELECT Query: Multiple Rows

- `fetch()` returns a single row

```
//Define the query
$sql = "SELECT * FROM animals
        WHERE animal_id = 3";

//query() returns the result
$result = $dbh->query($sql);

//fetch() returns the first row
$row = $result->fetch();
print $row['animal_type'] .
        ' - '. $row['animal_name'];
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

A SELECT Query: Multiple Rows

- set `exec()` returns the number of rows affected
- `lastInsertId()` returns the ID of the last inserted row

```
$sql = "INSERT INTO animals(animal_type, animal_name)
        VALUES ('kangaroo', 'troy')";
$dbh->exec($sql);

//Get the ID of the last inserted row
$id = $dbh->lastInsertId();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Prepared Statements

- A *prepared statement* is a pre-compiled SQL statement
- Can be reused
- Executes more quickly
- Prevents SQL injection

UNIVERSITY of GREENWICH
Alliance with FPT Education

Prepared Statements: Steps

- Define the query
  - `$sql = "…";`
- Prepare the statement
  - `$statement = $dbh->prepare($sql);`
- Bind the parameters
  - `$statement->bindParam(param_name, value, type);`
- Execute
  - `$statement->execute();`
- Process the result

UNIVERSITY of GREENWICH
Alliance with FPT Education

An Insert Query Example

```
//Define the query
$sql = "INSERT INTO animals(animal_type, animal_name)
         VALUES (:type, :name)";

//Prepare the statement
$statement = $dbh->prepare($sql);

//Bind the parameters
$type = 'kangaroo';
$name = 'Joey';
$statement->bindParam(':type', $type, PDO::PARAM_STR);
$statement->bindParam(':name', $name, PDO::PARAM_STR);

//Execute
$statement->execute();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

An Update Query Example

```
//Define the query
$sql = "UPDATE animals SET animal_name = :new
         WHERE animal_name = :old";

//Prepare the statement
$statement = $dbh->prepare($sql);

//Bind the parameters
$old = 'Joey';
$new = 'Troy';
$statement->bindParam(':old', $old, PDO::PARAM_STR);
$statement->bindParam(':new', $new, PDO::PARAM_STR);

//Execute
$statement->execute();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

A Delete Query Example

```
//Define the query
$sql = "DELETE FROM animals
        WHERE animal_type = :type";

//Prepare the statement
$statement = $dbh->prepare($sql);

//Bind the parameters
$type = 'kangaroo';
$statement->bindParam(':type', $type, PDO::PARAM_STR);

//Execute
$statement->execute();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

A SELECT Query: a Single Row

- `fetch()` returns a single row

```
//Define the query
$sql = "SELECT animal_name, animal_type FROM animals
          WHERE animal_id = :id";
//Prepare the statement
$statement = $dbh->prepare($sql);
//Bind the parameters
$id = 3;
$statement->bindParam(':id', $id, PDO::PARAM_INT);
//Execute the statement
$statement->execute();
//Process the result
$row = $statement->fetch(PDO::FETCH_ASSOC);
echo $row['animal_name']." - ".$row['animal_type'];
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

A SELECT Query: Multiple Rows Example

```
//Define the query
$sql = "SELECT animal_name, animal_type FROM animals";

//Prepare the statement
$statement = $dbh->prepare($sql);

//Execute the statement
$statement->execute();

//Process the result
$result = $statement->fetchAll(PDO::FETCH_ASSOC);
foreach ($result as $row) {
        echo $row['animal_type'] . ' - '   . $row['animal_name'];
}
```

MYSQLI_

UNIVERSITY of GREENWICH
Alliance with FPT Education

A SELECT Query: Multiple Rows

- The MySQLi Extension (MySQL Improved) is a relational database driver used in the PHP programming language to provide an interface with MySQL databases.
- The `mysqli` extension features a dual interface
  - Procedural
  - Object Oriented

UNIVERSITY of GREENWICH
Alliance with FPT Education

Connections Example

```
<?php
$mysqli = new mysqli("localhost", "user", "password",
"database");
if ($mysqli->connect_errno) {
    echo "Failed to connect to MySQL: (" .
       $mysqli->connect_errno . ") " .
       $mysqli->connect_error;
}
echo $mysqli->host_info . "\n";

?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Navigation through unbuffered results

```
<?php
$mysqli->real_query("SELECT id FROM test ORDER BY id
ASC");
$res = $mysqli->use_result();

echo "Result set order...\n";
while ($row = $res->fetch_assoc()) {
    echo " id = " . $row['id'] . "\n";
}
?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Select Example

```
$sql='SELECT col1, col2, col3 FROM table1 WHERE
condition';

$rs=$conn->query($sql);

if($rs === false) {
  trigger_error('Wrong SQL: ' . $sql . ' Error: ' .
$conn->error, E_USER_ERROR);
} else {
  $rows_returned = $rs->num_rows;
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Iterate Example

- Using column names – recommended

```
$rs->data_seek(0);
while($row = $rs->fetch_assoc()){
    echo $row['col1'] . '<br>';
}
```

- Using column index

```
$rs->data_seek(0);
while($row = $rs->fetch_row()){
    echo $row[0] . '<br>';
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Insert Example

```
$v1="'" . $conn->real_escape_string('col1_value') . "'";

$sql="INSERT INTO tbl (col1_varchar, col2_number) VALUES
($v1,10)";

if($conn->query($sql) === false) {
  trigger_error('Wrong SQL: ' . $sql . ' Error: ' .
$conn->error, E_USER_ERROR);
} else {
  $last_inserted_id = $conn->insert_id;
  $affected_rows = $conn->affected_rows;
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Update Example

```
$v1="'" . $conn->real_escape_string('col1_value') . "'";

$sql="UPDATE tbl SET col1_varchar=$v1, col2_number=1
WHERE id>10";

if($conn->query($sql) === false) {
  trigger_error('Wrong SQL: ' . $sql . ' Error: ' .
$conn->error, E_USER_ERROR);
} else {
  $affected_rows = $conn->affected_rows;
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Delete Example

- Use the following syntax:

```
$sql="DELETE FROM tbl WHERE id>10";

if($conn->query($sql) === false) {
  trigger_error('Wrong SQL: ' . $sql . ' Error: ' .
$conn->error, E_USER_ERROR);
} else {
  $affected_rows = $conn->affected_rows;
}
```