Database Access with PHP and MySQL

By,

Jestin James M
Assistant Professor, Dept of Computer Science
Little Flower College, Guruvayoor

PHP for Database Access

- Connect to the MySQL server
  - $connection = mysql_connect('localhost', $username, $password)
- Access the database
  - mysql_select_db('winestore', $connection)
- Perform SQL operations
- Disconnect from the server
  - mysql_close($connection)

Error Handling

- All mysql_ functions return NULL (or false) if they fail.
- Several functions are helpful in graceful failure
  - die(*string*) - halts and displays the string
  - mysql_errno() - returns number of error
  - mysql_error() - returns text of error

Error Handling examples

```
if (! ($connection = mysql_connect(' localhost' ,$name,$passwd)))
        die(' Could not connect' )

function showerror()
    {
        die(' Error ' . mysql_errno() . ' : ' . mysql_error())
    }

if (! (mysql_select_db(' winestor' , $connection)))
        showerror()
```

Building a Query

- Directly
  - $query = 'select * from wines'
- Using input information
  - $winery = $_POST['winery']
  - $query = “select * from wines where winery=$winery”

Running a Query

- mysql_query returns a result handle

```
$result = mysql_query($query, $connection)
```

- mysql_num_rows indicates the number of rows returned

```
$num_rows = mysql_num_rows($result)
```

- mysql_fetch_array creates array/hash of result

```
For ($n=0; $n<$num_rows; $n++)
        $row = mysql_fetch_array($result)
```

Result of fetch_array

- Contains both numeric and index tags
- Values are duplicated
- Example:
  - Query: select surname, city from customers;
  - Row: (0=>'Walker', ‘surname’=>'Walker', 1=>'Kent', 'city'=>'Kent');

Printing the Complete Row

- By number

```
for ($i=0; $i<mysql_num_fields($result); $i++)
    echo $row[$i] . "\n";
```

- By field

```
echo $row['surname'] . ' ' . $row['city']
```

Avoiding Special Characters

- When building HTML, characters such as '&' in the data can cause problems
- Function htmlspecialchars() replaces all such characters with HTML escapes such as &

```
print(   htmlspecialchars($row['surname']   .   '   '   .   $row['city'])
```

Special Characters in Input

- The same problem exists with special characters in input (e.g. ')
- PHP switch magic_quotes_gpc (default on) inserts backslashes before single & double quotes, backslashes and NULL characters in input data (from GET, PUT and cookie data)
- Use stripslashes() to remove the slashes
- Use addslashes() to add the slashes if magic_quotes_gpc is off

Avoid Dangerous User Input

- Passing user input to other programs opens the door to exploits
  - Eg. exec("/usr/bin/cal $input")
  - Generates a calendar (/usr/bin/cal 2004)
  - But a malevolent user might send '2004 ; cat /etc/passwd' or '2004 ; rm *'
- Overlong inputs can also cause problems
- Always clean input
  - $input = escapeshellcmd($input);
  - $input = substr($input,$maxlength);

PHP / Form in one Document

- Combine the original form with the PHP document that processes data

```
if empty($regionName)) { //parameter provided?
    //produce the <form>
}
else {
    //run the query using data from $_GET or
    $_POST
}
```

Inserting Into a Database

- Collect data from a form
- Validate data (JavaScript, PHP or both)
- Create a query

```
$query = "insert into customer set cust_id =
   NULL, " . "surname  =\"" . $surname ."\"" …
```

- Run the query

```
mysql_query($query, $db);
```

Updating a Database

- Query to find item to update
- Present old information
- Collect new information
- Validate
- Construct and run the update query