PHP-MYSQL AFFAIR

Last Lecture ...

1. PHP Variable, Array, Date and Time
2. Form Handling in PHP
3. HTML data Handling
4. Protecting from hackers
5. File IO in PHP
6. A Web Counter Example
7. The **all important** include statement.

Today – Useful PHP

- In the previous PHP lecture we moved from basics to advanced. Today, we continue:

1. PHP & MySQL
2. Starter MySQL
3. Connecting a database with PHP
4. Magic quotes Problem
5. MySQL connection function
6. Analysing that mySQL data
7. Terminating Execution

PHP & MySQL Chemistry

- Open source has brought a lot more than Linux to the computing world. It has also given us PHP and MySQL.
- *PHP and MySQL are viewed by many as the world's best combination for creating data-driven sites.*
- MySQL databases are ideal for storing that data we have collected about a user or for holding user preferences between visits. **It is free and it is easy**.

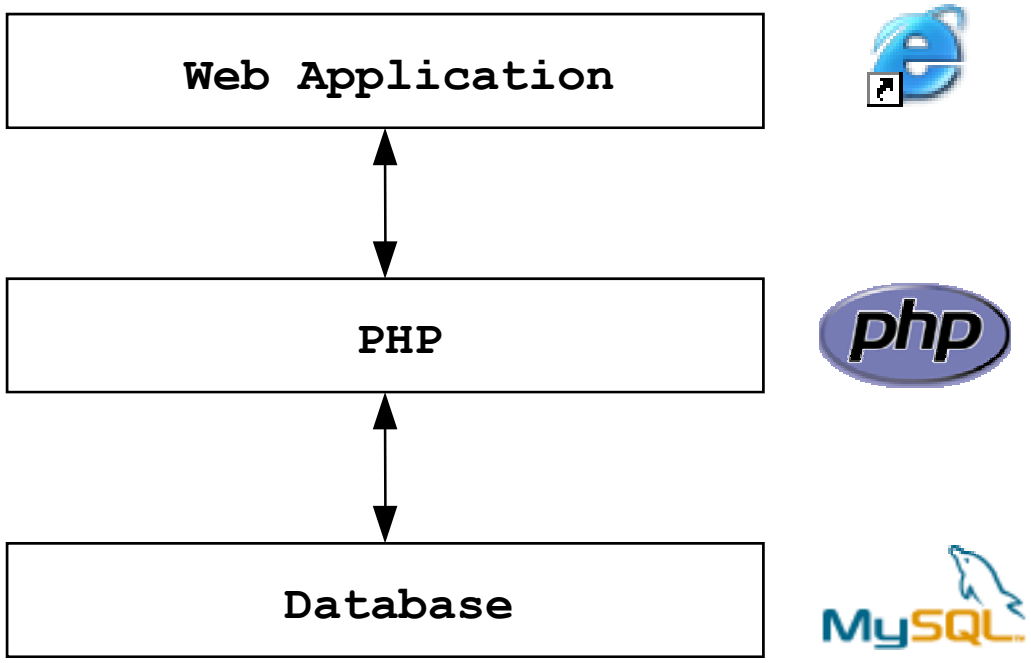
MySQL

- MySQL central is **http://www.mysql.com/.**

- We're not going to go through installation of mysql. WAMP has already installed MySQL for us.

A Web Application

- The diagram below shows the relationship between your user, the scripting language and the DB.



The Query

- The Query is the basic method by which data is entered or extracted from a database.
- It is common to all database systems.
- It is simply a command to the MySQL database in order to tell it to do something.

SQL Commands

- SQL is a reasonably powerful query language.
- However it is incredibly simple. You can learn it in a night.
- The fundamental SQL commands are:
  - **CREATE**
  - **SELECT**
  - **INSERT**
  - **DELETE**
  - **UPDATE**

Assignment 5: Playing with MySQL

- Develop relational database for Bank in MySQL having following tables
  - Branch (id INT, name VARCHAR(25), city VARCHAR(30), assets DECIMAL(14,4))
  - Transaction (id INT, timestamp DATETIME, account INT, amount DECIMAL(10,4))
  - Account (AccNumber INT, customerId INT, branchId INT, balance DECIMAL(10,4))
  - Customer (id INT, name VARCHAR(30), address VARCHAR(50), city VARCHAR(30), phone VARCHAR(15))

- Populate the database with some data.
- SELECT queries to retrieve:
  1. Insert a record in Branch table with values 18, Chaklala Scheme-III, Rawalpindi, `480925063.27 Rs.`
  2. Update a record in Branch table whose id is 18 and whose assets are increased by 50000Rs.
  3. Delete all records from Branch where assets are less than 20000Rs.
  4. Display all the branches whose name starts with s.

Assignment 5: Playing with MySQL

5. Display all the branches whose assets are greater than 5000000Rs
6. Retrieve cities and number of branches present in that city by using GROUP BY clause
7. Retrieve branch name with maximum assets in all the cities by using GROUP By clause.
8. List down sum of assets for a particular city having more than 5 branches. Render the results in ascending order of city name
9. Display all the customers name, account numbers, balance, branch names.

- Submit the queries along with the complete metadata of the database that you created in MySQL.
- (Use export command in the PHPMyAdmin to see and save complete metadata)

- Essential Reading
  - Web Database Application (2nd Ed.), Chapter 5 & 15

Creating a Table


```
CREATE TABLE people (
   first varchar(30),
   last varchar(30),
   address varchar(255)
);

INSERT INTO people VALUES
   ('George', 'Bush', 'Hell');

INSERT INTO people VALUES
   ('Adolf', 'Hitler', 'Unknown');

INSERT INTO people VALUES
   ('Pervez', 'Musharraf', 'Currently Homeless');
```

First MySQL/PHP Program


```
<?
   $db = mysql_connect("localhost", "root", "");
   mysql_select_db("mydb",$db);
   $result= mysql_query("SELECT * FROM people", $db);
   for($i = 0; $i < mysql_num_rows($result); $i++)
   {
     $first= mysql_result($result, $i, "first");
     $last = mysql_result($result, $i,  "last");
     $addr=mysql_result($result, $i, "address")."<br>";
   }
?>
```

You can also specify fields indices instead of field names.

First MySQL/PHP Program (Another way to access values)


```
<?
   $db = mysql_connect("localhost", "root", "");
   mysql_select_db("mydb",$db);
   $result= mysql_query("SELECT * FROM people", $db);
   while($row = mysql_fetch_array($result))
   {
        echo $row["first"];                 //echo $row[0];
        echo $row["last"];                  //echo $row[1];
        echo $row["address"] . "<br>";      //echo $row[2];
   }
?>
```

You can also specify fields indices instead of field names.

Step 1: mysql_connect()

- **mysql_connect()** establishes a connection to a MySQL server.

- It takes 3 parameters.
  - The address of the server
  - Your Username for that db account
  - Your password

```
$conn = mysql_connect("address", "user", "pass");
```

Step 2: mysql_select_db()

- In our code **mysql_select_db()** then tells PHP that any queries we make are against the mydb database.

```
mysql_select_db("dbname",$conn);
```

- We could create multiple connections to databases on different servers. But for now, you'll only need one database.

Step 3: mysql_query()

- Next, **mysql_query()** does all the hard work.
- Using the database connection identifier, it sends a line of SQL to the MySQL server to be processed.
- This is the key command for interacting with the database.
- In our example the results that are returned are stored in the variable **`$result`**.

Step 4 & 5: Iterating through each row and displaying each field

- Using mysql_num_rows($result), we iterate through each row, and return the value of the specified fields.

- Finally, **mysql_result()** is used to display the values of fields from our query:

```
mysql_result($result,$row,"first");
```

Its as easy as that

- So there we have it.
- We have successfully executed a simple PHP script to retrieve some information.
- Now we move on to the things that will be really helpful with the semester project…

1. Magic Quotes Problem

- For example say, we are putting a record into our database – the mysql query might be:

```
insert into people values ('Cameron Diaz');
```

- In you php this would be sent to the database using mysql_query as follows:

```
mysql_query('insert into people values ('Cameron
Diaz')');
```

- This would cause a parse error – so make sure you use the correct quotes to avoid this.

2. MySQL Connection Function

- Here is a function that automates connecting to a certain database. Save it in a separate file e.g. ConnectToDB.inc

```
function db_connect()
{
 $result = mysql_connect("localhost", "jog", "pass");
 if (!$result) return false;
 if (!mysql_select_db("people")) return false;
 return $result;
}
```

- It is a good practice to write these statements in a function and separate file.
- Include this file and call the function wherever you want to connect to database.

```
<?
    include_once("ConnectToDB.inc");
?>
```

Making a query (reminder)

- Now we have connected by calling this function, we can access the database to make a query to it.
- Remember to send a mySQL instruction to the database we use **mysql_query()**
- You can make absolutely any query that you would type into the database command line via PHP in this way.

Create Table Example

- For example, to create a table from our PHP code you might type:

```
mysql_query("CREATE TABLE actors (
                    name varchar(30),
                    age integer)");
```

- Remember that this is something that you would only want to do once – once the table is created we don't want to wipe it by accident.

MySQL Insert Example

- Equally we can populate our tables with INSERT statements via mysql_query()

```
mysql_query("INSERT INTO actors VALUES
            ('Drew Barrymore',34)");

mysql_query("INSERT INTO actors VALUES
            ('Cameron Diaz',36)");

mysql_query("INSERT INTO actors VALUES
            ('Tom Cruise',40)");
```

- These are hard coded examples – but we could be using variables in these statements

Mysql Select Example

- We use a SELECT statement to grab data from a certain table and then put the result into a variable ready to analyse…

```
$result = mysql_query("SELECT * FROM actors WHERE age<35);
```

- However now result has all the info we want inside it… you can use any of the following function to access the returned result.

- **mysql_fetch_array()** is an extended version of **mysql_fetch_row()**.
- In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

- Which you use is up to you. Both functions are pretty similar.

mysql_num_rows()

- **mysql_num_rows()** returns the number of rows in a result set. This command is only valid for SELECT statements.

```
mysql_query("SELECT * FROM actors WHERE age<35);
print mysql_num_rows()." actors are younger than 35";
```

- It's a great function for when you need to loop round all the results in your query, or just to know how many matches you got

mysql_affected_rows()

- **mysql_affected_rows()** returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with. For example:

```
mysql_query("DELETE FROM mytable WHERE id < 10");

print "Records deleted: ".mysql_affected_rows()."<BR>";
```

- **Note:** this function does not work with SELECT statements - only on statements which modify records.

4. Terminating Execution

- There are two ways to stop the execution of a script. The first is using the **exit()** statement which simply stops the script wihtout returning anything.

- More useful – especially for bug checking – is the **die()** command.

- This language construct can be used to output an error message or execute a function before terminating the script.

Why won't you just die mr.bond…

```
die("Could not execute query");
```

- This would simply exit the script and send the message to the browser. However you can add a die statement to another using the or command…for example:

```
mysql_query($query)
        or die("Could not execute query");
```

- Alternatively you can use die to fire off **a function** – maybe you want to email notification to yourself when a major error has occurred or add errors to a log file?

Summary

- We studied

1. PHP & MySQL
2. Starter MySQL
3. Connecting a database with PHP
4. Magic quotes problem
5. mySQL connection function
6. Analysing that mySQL data
7. Terminating Execution