1 PHP

PHP is used to create dynamic websites. It is very easy to create a website that uses a database using PHP. It is similar to ASP, or JSP—you have mostly HTML like tags, with some code between them to do things.

Now, in this document, we won't be overly concerned with PHP itself, rather the MySQL interfaces from PHP. If you're interested in learning PHP, you are encouraged to visit: `http://www.php.net/` and read the documentation.

2 PHP MySQL Exampmle

There is a nice example on the `http://www.php.net/` website:

```
<?php
/* Connecting, selecting database */
$link = mysql_connect("mysql_host", "mysql_user", "mysql_password")
   or die("Could not connect : " . mysql_error());
echo "Connected successfully";
mysql_select_db("my_database") or die("Could not select database");

/* Performing SQL query */
$query = "SELECT * FROM my_table";
$result = mysql_query($query) or die("Query failed : " . mysql_error());

/* Printing results in HTML */
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
   echo "\t<tr>\n";
   foreach ($line as $col_value) {
       echo "\t\t<td>$col_value</td>\n";
   }
   echo "\t</tr>\n";
}
echo "</table>\n";

/* Free resultset */
mysql_free_result($result);

/* Closing connection */
mysql_close($link);
?>
```

What this code does is: connect to database, select appropriate database, execute a query, display results, and disconnect. That's all. You really don't need much beyond this to write database driven websites.

3 Basic Design Strategies

If starting from the code above, you can get yourself into a big mess without some design strategy. The first thing to consider is separating the database functionality from the 'mysql' specific calls. You do this by defining wrapper methods, like this:

```
<?php
function db_open(){
    $link = @mysql_connect("localhost","username","password");
    @mysql_select_db("database_name");
    return $link;
}

function db_close($link){
    @mysql_close($link);
}

function db_query($q){
    return @mysql_query($q);
}

function db_duplicate(){
    return mysql_errno() == 1062;
}

function db_fetch($r){
    return @mysql_fetch_array($r,MYSQL_ASSOC);
}

function db_freeres($r){
    @mysql_free_result($r);
}

function db_insertid(){
    return mysql_insert_id();
}
?>
```

This code would be placed in some outside location, like `db.php`, and included in all documents that need database access. If the need ever arises, you can replace this file with calls to other databases, etc.

The `db_open` call opens a connection to the specified database. The caller of this method shoudln't have to worry about where the database is, etc. The `db_close` accomplishes the opposite effect.

The `db_query` accepts a string representing any SQL statement, and runs it. If it's a 'select' statement, the query returns a result set; which you can fetch via `db_fetch` method.

Once done reading results, it is a good idea to free the result set by calling `db_freeres`.

The `db_duplicate` predicate returns true/false if the last insertion failed because of a duplicate key. This is useful for ensuring usernames are unique in the system.

The `db_insertid` returns the last auto-generated id from the last insert statement.

Together, all these functions enable us to do pretty much anything with the database.

4 Sample Application: Guest Book

As an example application, we'll implement a Guest Book. We'll proceed in steps, of first building the database, then guestbook viewing code, and then guest book signing code.

4.1 Schema

A schema is a file that creates the database (and all the tables). For this application, we'll need to create 'gbook' database:

```
-- first setup database (as 'root') via
create database gbook;

grant all on gbook.* to gbook identified by 'gbook';

-- (don't forget to mysqladmin -uroot -p reload)
```

We've just created the 'gbook' database and gave it to user 'gbook' whose password is also 'gbook'. Normally we'd pick more interesting names, but this is just an example. Next, we need to create the actual 'gbook' table:

```
use gbook;

create table gbook (
    id int auto_increment,
    name varchar(64),
    email varchar(64),
    message text,
    postdate datetime,
    primary key(id)
);
```

We first ensure we are using an appropriate database, we then use the 'create table' command to create the database—which is hopefully appropriate for an average guest book.

4.2 Guest Book Page

The guestbook main page (`gbook.php`) does this: Connect to database, retrieve all guest book entries, display them, display an entry form, and that's it. Here's the code:

```
<?php

include_once("db.php");

// connect to db
$link = db_open();

// if no db
if(!$link){
    Header("Location: nodb.php");
    exit;
}
?><html>
<head>
<title>Guest Book!</title>
</head>
<body>
<center>
<h3>Welcome to the Guest Book Page!</h3>

<p>[<a href="gbook.tar.gz">donwload source code</a>]</p>

<?php
    $sql = "select * from gbook order by postdate";
    if($result = db_query($sql)){
        ?><table border="1" width="60%" bgcolor="#DDDDDD"
            cellspacing="1" cellpadding="5">
        <?php
        while ($row = db_fetch($result)){
            ?><tr>
            <td width="20%" bgcolor="#EEEEEE"
                valign="top">
              <nobr><?=$row['name']?></nobr><br>
              <small><?=$row['email']?></small><br>
              <small><?=$row['postdate']?></small>
            </td>
            <td width="80%" bgcolor="#EEEEEE"
                valign="top">
              <?=$row['message']?>
            </td>
           </tr><?php
        }
        ?></table>
        <?php
        db_freeres($result);
```

```
    }else{
        ?><h3 style="color:red">Sorry,
            database Error.</h3><?php
    }
?>

<p> </p>

<h3>Sign this Guest Book!</h3>
<form action="gbook_action.php" method="POST">
<table border="0" width="40%" bgcolor="#DDDDDD">
 <tr>
  <td style="width:10%" align="right">Name:</td>
  <td style="width:90%">
   <input type="text" name="name" style="width:100%">
  </td>
 </tr>
 <tr>
  <td style="width:10%" align="right">
   <nobr>E-Mail:</nobr>
  </td>
  <td style="width:90%">
   <input type="text" name="email" style="width:100%">
  </td>
 </tr>
 <tr>
  <td style="width:100%" colspan="2">
   <textarea cols="60" rows="6"
   style="width:100%" name="message"></textarea>
  </td>
 </tr>
 <tr>
  <td style="width:100%" colspan="2" align="right">
   <input type="submit" value="Sign!">
  </td>
 </tr>
</table>
</form>

<p> </p>

<p>&copy; 2004, CIS45</p>

</center>
</body>
```

```
</html><?php
// close connection
db_close($link);
exit;
?>
```

Notice that the form action is `gbook_action.php`, which is another page that processes the 'sign' action.

4.3 Action Processing

When the user clicks on the "Sign" button, we need to add their signature to the database. We do this in `gbook_action.php`, which follows:

```
<?php

include_once("db.php");

// connect to db
$link = db_open();

// if no db
if(!$link){
    Header("Location: nodb.php");
    exit;
}

$name = trim($_POST['name']);
$email = trim($_POST['email']);
$message = trim($_POST['message']);

$sql = "insert into gbook(name,email,message,postdate)".
       "values('$name','$email','$message',NOW())";
db_query($sql);

// close connection
db_close($link);

// go back to main page.
Header("Location: gbook.php");
exit;
?>
```

Notice that all we're doing is getting user parameters, constructing a SQL statement, and then calling database with that statement. In the end, we go back to the `gbook.php` page.

4.4 Running Example

You can see the example of this running at: `http://particle.homeip.net/gbook/`