PHP Programming

What is PHP?

- PHP stands for PHP Hypertext Processor — A recursive definition!!.
- PHP is a server-side scripting language that is embedded in a web page or can be run as script (much Like Perl) from the command line (Full support since version 4.3)

PHP Characteristics

The main characteristics of PHP are:

- PHP is web-specific and open source
- Scripts are embedded into static HTML files
- Fast execution of scripts
- Fast access to the database tier of applications
- Supported by most web servers and operating systems
- Supports many standard network protocols libraries available for IMAP, NNTP, SMTP, POP3,
- Supports many database management systems libraries available for UNIX DBM, MySQL, Oracle,
- Dynamic Output any text, HTML XHTML and any other XML file.
- Also Dynamic Output images, PDF files and even Flash movies
- Text processing features, from the POSIX Extended or Perl regular expressions to parsing XML documents.
- A fully featured programming language suitable for complex systems development

What can PHP do?

As we have just said PHP is a fully featured programming language so it can do just about anything.

However it is best suited and is mainly focused on server-side scripting, so you can do anything any other CGI program can do such as:

- Collect form data,
- Generate dynamic page content,
- Send and receive cookies.
- But PHP can do much more than we have not got time to address here
  - Please refer to recommended course books
  - Web sites: http://www.php.org, http://www.php-scripts.com, http://php.resourceindex.com, and others.

Three main uses of PHP

Server-side scripting — This is the most traditional and main target field for PHP. You need three things to make this work:

- The PHP parser (CGI or server module),
- A web server — needs a connected PHP installation
- A web browser — access PHP page through URL

Command line scripting — You can make a PHP script to run without any server or browser. You only need the PHP parser to use it this way. These scripts can also be used for simple text processing tasks similar to PERL

Writing client-side GUI applications — PHP is probably not the very best language to write windowing applications, but PHP-GTK (PHP Graphics Tool Kit) can be used to write such programs. **Not dealt with in this course.**

PHP support and installation

PHP can be used on all major operating systems:

- Linux,
- Many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows,
- Mac OS X,
- Windows
- RISC OS,
- and probably others.

PHP already installed on School's Web Server

Information on installing PHP on Windows may be found at:
http://www.php.net

PHP v Perl

Let us compare PERL and PHP first

Perl:

- Interpreted language optimized to process text files — Ideal for CGI use.
- Using a Perl script as a CGI means that when an online form calls the script to be processed, the web server will load the Perl script, and pipe the output of the script back to the user, on its web browser.
- Simple syntax, similar to C
- Includes advanced functionality through numerous packages (CPAN)

PHP v Perl (Cont.)

PHP :

- Server-side scripting language embedded in HTML pages.
- Processed by the web server when the HTML page is loaded.
- PHP code included in special HTML tags which are parsed by the web server.
- Syntax is different from Perl (but not that much different)
- PHP also has some very advanced functions.

Advantages of Perl/PHP

Perl has a lot of advantages:

- General purpose language — functions to do nearly everything.
- Perl has modules that can be downloaded and loaded in a Perl script.
- Scripts can also be made to be processed by the web server and by a user from the shell (NOT WEB BASED), if needed.

Advantages of Perl/PHP (Cont.)

PHP also has some advantages:

- It is included inside HTML pages.
  This means that:
  - All your work can be done in the same directory, and
  - A single file can contain HTML and PHP code
  - Much easier to edit/maintain web pages.
  - This greatly improves tasks like
    - Dynamic page processing.
    - Checking and doing simple HTML Form based tasks.
    - Database connectivity

Disadvantages of Perl/PHP

Perl:

- Management of **simple** scripts is more difficult than PHP:
  - Scripts need to be loaded separately from the HTML pages.
  - Typically scripts also be kept in a special directory — `E.g. cgi-lib.pl`,
- PHP integrates itself very well in HTML files —
  so scripts don't need to be loaded from a separate location.
- However, if the PHP code is complex it can lead to a complex mixture of code on the page.

Disadvantages of Perl/PHP (Cont.)

PHP:

- Great for simple Dynamic web page processing,
- Not so great for more complex tasks
  - Harder to parse/understand and maintain lots of code mixed in with web page (see above).
  - Not as **general purpose** as Perl.
- PHP requires that the server is configured to parse PHP files.
  - Perl similar but both now widely supported.
- Also, PHP files need to be parsed by the server at each load, so they should not be used in files that don't need to be processed.
  - Perl similar but both slightly less overhead.

Examples of Uses of Perl/PHP

Neither Perl **Nor** PHP best for **all** web tasks:

- Horses for Courses

Perl : Heavy CGI Computations, **E.g.**

- Search Engines,
- Data processing,
- Heavy File access **etc.**
- Heavy Database Access/Usage

PHP : Better Suited for dynamic HTML, **E.g.**

- Simple Form processing
- Making a complex site integration
  E.g. with a database connected — simple database connectivity.

Creating, Writing and Executing PHP

- PHP files are simply text files — use any text editor, integrate with HTML
- File has extension `.php`
- Store file the same directory as HTML pages (not `cgi-bin`)
- Link from web pages via standard URL

Using PHP on COMSC Web Server

- COMSC Web server preconfigured to run PHP
- Use Unix web space (`public_html` or `project_html`)
- Create file on Macs (or PCs) ftp or upload via file sharing.
- Store files with `.php` extension
- Run files by accessing through Web browser
  - **Must be run via web server URL**
  - **Cannot run PHP** through local `file://` URL.
- Possible to run (test/debug) scripts from UNIX/Mac OS X **command line** via **Telnet/Terminal**.

Including PHP in a Web Page

There are 4 ways of including PHP in a web page

1. `<?php echo("Hello world"); ?>`
2. ```
   <script language = "php">
     echo("Hello world");
   </script>
   ```
3. `<? echo("Hello world"); ?>`
4. `<% echo("Hello world"); %>`

You can also use *print* instead of *echo*

- Method (1) is clear and unambiguous — **My preferred method**
- Method (2) is useful in environments supporting mixed scripting languages in the same HTML file (most do not)
- Methods (3) and (4) depend on the server configuration

What happens when the page is loaded?

When the script is run:

- The code is executed and
- The tag is replaced the by the output (`''Hello world''`) in examples above.
- Replacement is exactly where the PHP is relation to HTML.
  - When you view source of a PHP/HTML page **you do not see the PHP**.
- Can have more than one PHP tag in a single Web page.
   ```

A Simple PHP Script Example

Here is first complete PHP script which is embedded in HTML:

- We create a level one header with the PHP output text.
- This file is called hello.php:

```
<html>
 <head>
  <title>Hello world</title>
 </head>
 <body>
  <h1><?php echo("Hello world"); ?></h1>
  <h1><?php print("This prints the same thing!"); ?></h1>

 </body>
<html>
```


Basic PHP

Comments

PHP supports **three** types of comments:

1. Shell style comments - denoted `#THIS IS A COMMENT`
2. C++ style comments - denoted THIS IS A COMMENT—
3. C style comments - denoted `/* ALL THIS COMMENTED! */`

Variables

Variables start with the `$` symbol (like Perl).

E.g.:

```
$myInteger = 3;
$myString = "Hello world";
$myFloat = 3.145;
```


PHP Data Types

Data types are not explicitly defined:

- Variable type is determined by **assignment**.
- **Different to Perl**.
- Strings can be defined with single (') and double (") quotes.
  - Rules for Strings as in Perl.
- PHP has a boolean type:

  Defined as false
  - An integer or float value of `0` or
  - The keyword `false`
  - The empty string `''` or the string `'0'`
  - An empty array or object
  - The `NULL` value

  Defined as true
  - **Any non-zero** integer or float value
  - The keyword `true`
- Standard operators with standard syntax applied to variables

Arrays

```
$myArray[0] = "Apples";
$myArray[1] = "Bananas";
```

Arrays can be **constructed** using the `array()` keyword.

```
$person = array("Dave", "Adam", "Ralph");
```

The `foreach` command can be used to loop through an array:

```
foreach ($person as $name){
  echo "Hello, $name\n";
}
```

Two things to note in the above:

- The use of the `array()` constructor.
- The syntax of `foreach` — **different to Perl**.

Associative arrays

```
$myArray["Monday"] = "Apples";
$myArray["Tuesday"] = "Bananas";
```

Associative Arrays can also be **constructed** using the `array()` keyword.

```
$food = array("Monday" => "Apples",
              "Tuesday" => "Bananas");
```

The `foreach` command can be used to loop through an associative array:

```
foreach ($food as $day => $item){
  echo "Today is  $day, Eat $item\n";
}
```

A few things to note in the above:

- The use of the `array()` constructor
- The syntax of `foreach` — **different to Perl**.
- The => delimits the hash name from the hash value

Many array functions also exist, for example:
`array_keys, array_merge, array_pop, asort`

Scope of Variables

Once PHP variables have been defined they are known for the rest of the Web page:

- Obeying standard scoping rules of course.
- Variables can be **local to functions** etc, much like any languages.
- **Some examples with functions soon**.

So another version of the "Hello World" program is `hello2.php` which uses a variable string:

```
<?php $outputString = "Hello world"; ?>
<html>
 <head>
  <title><?php echo($outputString); ?>
  </title>
 </head>
 <body>
  <h1><?php echo($outputString); ?></h1>
 </body>
</html>
```


Functions

Built in functions

PHP has many built in functions, **E.g.**:

Mathematical functions :

`abs, ceil, cos, log, min, rand, sqrt`

File handling :

`fopen, flock, feof, fgets, fputs, fclose`

We will meet some more when we study a few practical examples later.

See references (books, web pages etc.) for more details

User-defined functions

Functions defined much like most programming languages:

```
function show($myArray) {
  echo("<h1>Fruit of the day</h1>\n<ul>"
  foreach($myArray as $key => $val) {
    echo ("<li>$key -- $val</li>\n");
   }
  echo("</ul>\n";
}

$food = array("Monday" => "Apples",
              "Tuesday" => "Bananas");

show($food);
```


Local Variables

Variables defined in a function are simply **local** to that function.

E.G.:

```
function dummy($val) {
   $local_val = $val;
   return $local_val;
}

$a = 3;
$b = dummy($a); # Wasted op?
$c = $local_val; # Illegal op.
```


Variable Scope Example: What happens here?

```
$a = 3;

function what() {
    ++$a;
    echo "a = $a\n";
}

what();
echo "a = $a\n";
```

Is variable `$a` **global/local?**

What is the output of both `echo(..)`s — **Is it 4?**
Answer: **NO it is 1 and then 3!**

We actually have two variables called `$a`!!

Scope Example Explained

```
$a = 3;

function what() {
    ++$a;
    echo "a = $a\n";
}

what();
echo "a = $a\n";
```

In PHP:

- The variable `$a` in the function `what()` is **different** to outside variable `$a`
- Not recommended programming practice to have two variables of the same name.
- So the local function prints the value 1 whilst
- The outer prints 3.

Global Variables

So how do we share `global` values with a function?

- You **must** declare a variable inside a function as `global`

So our previous example could be phrased as:

```
$a = 3;

function what() {
    global $a;

    ++$a;
    echo "a = $a\n";
}

what();
echo "a = $a\n";
```

Now the output is 4 at both prints.

Flow-Control Statements

PHP provides the standard Flow-controls statements

`if, if/else`

`switch`

`while`

`for`

Basic syntax is same as Perl, Java, C, C++.

We therefore jump over examples here for now

Regular Expressions in PHP

- Syntax is identical to those found in Perl (and JavaScript)
- Use PHP function calls however to action a Regular Expression.
- Having created a pattern, we use one of the following functions:

  `preg_match`,
  `preg_match_all`,
  `preg_replace`,
  `preg_split`

Regular Expressions in PHP Example

For example, regexp.php:

```
<?php
  $test = "This is a Test";

  if (preg_match("/(\ws)\s(\ws)/", $test, $matches)) {
     echo "<h3>Matched</h3>\n";
     echo "$matches[0], <br>
     echo "$matches[1], <br>
     echo "$matches[2], <br>
 } else {
     echo "<h3>Did Not Match</h3>";
?>
```

The output is:

Matched

is is,
is,
is,

PHP Regular Expression Example Explained

The function `preg_match()`:

- Can take just two parameters as well: **E.g.**

```
if (preg_match($pattern,$test) {
    echo("Match Found\n");
}
```

- However in this more complex example we use the **parenthesis as memory** feature of regular expressions.
- The `$matches` parameter is used to store the results of matches in the brackets
- The `(\ws)` matches any word character followed by a letter `s`.
- The two matches are separated by a single space `\s`
- This first element of `$matches` stores to complete pattern match: `is is`
- The next two elements are the sub matches (delimited by parenthesis): two times `is`

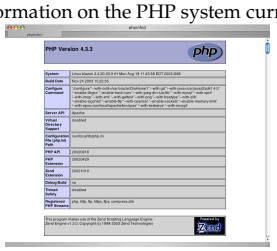
Some Practical PHP Examples

A 1 line PHP Script

Here a simple 1 line PHP script, `phpinfo.php`:

```
<?php phpinfo() ?>
```

It prints out information on the PHP system currently running:



A 1 line PHP Script (Cont.)

```
<?php phpinfo() ?>
```

It is not much use except for:

- Testing that PHP is actually installed on the Web server
- Finding Version of PHP that is installed on the Web server
- Getting other PHP/Server stats

Handling HTML Forms

Handling forms could not be much easier than in PHP:

- You only need call a single line to make all form name variables available.
- Variables are given the same name as the `name` of the form item
- As in Perl ,a forms action references a script, .php in this case.
- The .php script then *extracts* the information, making it global.

A Simple PHP/HTML Form Processing Example

Lets Consider the Following HTML Form Source, form1.html:

```
<html> <head>
<title>Form 1</title>
</head>
<body>
<form name="whoAreYou" method="GET"
action="form1.php">
<h2>Who are you?</h2>
<h4>Name:<input type="text" name="username"></h4>
<h4>Address:<input type="text"  name="address"></h4>
<input type="submit" value="Send">
</form>
</body> </html>
```

Not much different from before (Perl/CGI) **EXCEPT**

- `action` refers to PHP script URL
  `http://www.cs.cf.ac.uk/user/Dave.Marshall/form1.php`
  in this case
- URL need not link to `cgi-bin`

A Simple PHP/HTML Form Processing Example (Cont.)

The form clearly looks like this:

Who are you?

Name:

Address:

Send

A Simple PHP/HTML Form Processing Example (Cont.)

The PHP which processes this is very simple, form1.php:

```
<html>
 <head>
  <title>form1.php</title>
 </head>
 <?php extract($_GET); ?>
 <body>
  <h2>Your name is: <?php echo($username); ?></h2>
  <h2>Your address is: <?php echo($address); ?></h2>
 </body>
<html>
```

The output of the PHP script looks something like this:

Your name is: David

Your address is: COMSC

Alternatively, you could have

```
<html>
 <head><title>form_do.php</title> </head>
 <body>
  <?php
  extract( $_GET );
  print("<h2>HELLO</h2>");
  print $username;
  print("<h1>address</h1>");
  print $address;
  ?>
 </body>
</html>
```

- Note that HTML tags may be enclosed in the print statements
- `extract($_GET);` is used to access form variables.
- If using POST then use `extract($_POST);` instead of GET

Creating and Processing Forms in One PHP file

You can have the `action` of form `self-reference` the page that created the form, This:

- Keeps all form processing in one file
- Is excellent for Web page develop and upkeep if:
  - Small PHP Scripts
  - Not **too many** fragments of PHP in single document.
- Example: Highlights many good features of PHP

Forms and Processing Forms in one PHP/HTML File Example

Consider the following example, a very basic/limited calculator,

```
<html>
<head>
<title>Very Simple Calculator Indeed...</title>
</head>

<body>
<h1>CALCULATOR!</h1>

<form action = "<?php echo $_SERVER['PHP_SELF']; ?>"
      method = "GET">
Enter a number:
<input  type = "text" name = "thenumber" /> <br/>
<input type = "submit" name = "Mul Num"/>
</form>

<?php
extract($_GET);
$theanswer = $thenumber* 321;
print("The number x 321 is:". $theanswer);
?>
</body>
</html>
```


Forms and Processing Forms in one PHP/HTML File Example Explained

One new PHP point to note:

- PHP has a variety of special variables — a bit like Perl.
- `$_SERVER['PHP_SELF']` is an associative array that holds information about the server that runs the script.
- `PHP_SELF` refers to the current PHP page that is running and set the `ACTION` attribute:
  - Makes for portable code to use this
  - Can change file name and it **still self-references** the page
- Much better than and explicit URL to the page
- Need to print the elements to form. Can do this in php script + format with HTML

Sending Email from PHP

Again this is a relatively simple task in PHP:

- The `mail()` function allows your scripts to send email.

For example:

```
<html>
 <head>
  <title>Email</title>
 </head>
 <body>
 <?php
$to = "A.student@cs.cardiff.ac.uk";
$subject = "PHP is Great";
$body = "Hello how are you goodbye";
$headers = "From: A.N.Other@cs.cardiff.ac.uk\r\n";
$sent = mail($to, $subject, $body, $headers);

if($sent)
    echo "Mail sent to $to";
else
echo "Error sending mail";
 ?>
 </body>
</html>
```

Sending Email from PHP Example Explained

- Hopefully everything is pretty much self explanatory
- Note that `mail()`
  - Has obvious parameters:

    `$to, $subject, $body, $headers`.
  - Returns a boolean value (stored in `$sent`) which checked by `if` statement

Dates in PHP

PHP has many functions for manipulating dates and times.

- The `date(date_string)` function can be used
  - To extract date information out in a variety of formats depending on parameters set in the `date_string)`.
- The date string, `date_string)`, may have the following formats

Date String Formats

Character	Meaning
a	Display "am" or "pm"
A	Display "AM" or "PM"
B	Swatch Internet time
d	day of the month, 2 digits with leading zeros; i.e. "01" to "31"
D	day of the week, textual, 3 letters; e.g. "Fri"
F	month, textual, long; e.g. "January"
g	hour, 12-hour format without leading zeros; i.e. "1" to "12"
G	hour, 24-hour format without leading zeros; i.e. "0" to "23"
h	hour, 12-hour format; i.e. "01" to "12"
H	hour, 24-hour format; i.e. "00" to "23"
i	minutes; i.e. "00" to "59"
I (capital i)	"1" if Daylight Savings Time, "0" otherwise.
j	day of the month without leading zeros; i.e. "1" to "31"
l (lowercase 'L')	day of the week, textual, long; e.g. "Friday"
L	boolean for whether it is a leap year; i.e. "0" or "1"
m	month; i.e. "01" to "12"
M	month, textual, 3 letters; e.g. "Jan"
n	month without leading zeros; i.e. "1" to "12"
O	Difference to Greenwich time in hours; e.g. "+0200"
r	RFC 822 formatted date; e.g. "Thu, 21 Dec 2000 16:01:07 +0200" (added in PHP 4.0.4)
s	seconds; i.e. "00" to "59"
S	English ordinal suffix for the day of the month, 2 characters; i.e. "st", "nd", "rd" or "th"
t	number of days in the given month; i.e. "28" to "31"
T	Timezone setting of this machine; e.g. "EST" or "MDT"
U	seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)
w	day of the week, numeric, i.e. "0" (Sunday) to "6" (Saturday)
W	ISO 8601 week number of year, weeks starting on Monday (added in PHP 4.1.0)
Y	year, 4 digits; e.g. "1999"
y	year, 2 digits; e.g. "99"
z	day of the year; i.e. "0" to "365"
Z	timezone offset in seconds (i.e. "-43200" to "43200"). The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.

Simple PHP Date Example

The following PHP sets the background colour to blue on Tuesdays, tuesday.php

```
<?php
  if(date("D") == "Tue") $colour = "blue";
  else $colour = "red";
?>
<html>
 <head>
  <title>Welcome</title>
 </head>
 <body bgcolor = <?php echo($colour) ?>>
   <h1>Welcome</h1>
 </body>
</html>
```

- Here we simply get day by setting the date string to "`D`"
- `if` sets up the `$colour` variable accordingly
- `echo $colour` to set `body` background colour `bgcolor`.

Another Date Example

xmas.php prints out the current date and works out how long it is to Christmas:

```
<html>
 <head>
  <title>Dates: Days to Christmas</title>
 </head>
 <body>
 <h1>Dates: Days to Christmas</h1>
 <p>
 Today is:<br>
 <?php
   echo(date("l dS F, Y"));
  ?>
 </p>
 <p> There are:
 <?php
  $daysgone = date("z");
  $daystoxmas = 358 - $daysgone;
  echo($daystoxmas);
 ?> days to Christmas</p>

<p><b> Better get shopping now!!!!</b></p>
</body>
</html>
```


Xmas Date Example Sample Output

When run the following output is produced in a Web browser:

Dates:Days to Christmas

Today is:
Tuesday 02nd December, 2003

There are: 23 days to Christmas

Better get shopping now!!!!

Xmas Date Example Sample Explained

Things to note in the above xmas.php code:

- We use the date() function twice
- Firstly we print out the date in a fancy format as can be seen in the screenshot of the display above.
- The format string "l dS F, Y" cause the long day of the week, day of the month with ordinal Suffix followed by a (verbatim) comma, followed a 4-digit Year.
- Secondly date() call loads a variable with the number of days gone in the year so far.
- Simple to compute days to go to xmas and format HTML output

Other date functions include:
getdate(), localtime(), mktime(), time()

Cookies and PHP

- A cookie is a text string stored on the client machine by your script (to track users and manage transactions)
- Cookies are automatically returned (by the client), and can be accessed using a variable of the same name
- The following script reads and displays a cookie, and sets it with a new value (string) that was passed to the script as a parameter.
- The cookie will expire after 20 minutes (1200 seconds)

```
<?php setCookie("CookieTest", $val, time()+1200); ?>
<html>
 <head><title>Welcome</title></head>
 <body>
   <?php echo("<h2>The cookie is: $CookieTest</h1>
 </body>
</html>
```

Checking for Cookies

- The Cookie can now be used to monitor users on following pages.
- For example, a proceeding page may only be accessible if a cookie has been sent.
- You could set a cookie after e.g. a successful log-in to a secure site

```
<html>
<body>
<?php
 if (isset($_COOKIE["uname"]))
   print "Welcome " . $_COOKIE["uname"] . "!<br/>";
 else
   print "You are not logged in!<br />";
?>
</body>
</html>
```


Database Connections in PHP

To complete our brief introduction to PHP lets see how easy it is to connect to a database.

- PHP can connect to a variety of databases
- We will use **MySQL** which is available in dept.
- For info on MYSQL see hrefhttp://www.mysql.com/documentation/index.html**http://www.mysq**
- You will need to see Mr Robert Evans (Dept. System Adim) Room C 2.14a to get a MySQL account set up
- This is **Not a lecture on Databases or SQL**
- Simple example and study of PHP connection here.

MySQL

- MySQL is a database management system (DBMS) for relational databases, based on the Standard Query Language (SQL)
- MySQL is open source
- For further information on MySQL see **http://www.cs.cf.ac.uk/systems/applications/mysql/**
- Use **phpMyAdmin** to interact with MySQL scripts providing an administrative interface to MySQL
  - See **http://www.cs.cf.ac.uk/phpMyAdmin.php** (**Need MySQL account Set Up First**)
  - Also can command line `mysql -h sentinel -u username -p databasename` for MYSQL Access

MySQL (Cont.)

- MySQL manages a system of relational databases
- A **username** and **password** are required to access the database system
- Once access is permitted, a user is able to query and/or update various databases within the system
- Each database contains tables
- Each table contains records (rows)
  - Records are made up of fields
- Warning don't use a database unless you need one!
- PHP is good at simple connections and when SQL and database to a lot of data processing.
- Perl is better if after extraction from a database you need to some heavy processing within the script.

MySQL Example: Setting up a MySQL Table

Lets develop a very simple database application.

- Lets look at the SQL side first.
- First we create a table.
  - We do this first from the MySQL command line `mysql>`
  - MySQL output shown.
  - Later you could do this from PHP (or Perl).

```
mysql> create table toys (
        -> toy_id int(5) NOT NULL auto_increment,
        -> toy_name varchar(50) NOT NULL,
        -> PRIMARY KEY (toy_id),
        -> KEY name (toy_name));
Query OK, 0 rows affected (0.01 sec)

mysql> describe toys;
+----------+--------------+------+-----+---------+
| Field    | Type         | Null | Key | Default |
+----------+--------------+------+-----+---------+
| toy_id   | int(5)       |      | PRI | NULL    |
| toy_name | varchar(100) |      | MUL |         |
+----------+--------------+------+-----+---------+
2 rows in set (0.00 sec)
```


MySQL Example: Populating the Database

Next we populate the database with entries:

```
mysql> INSERT INTO toys VALUES (NULL, "Woody");
mysql> INSERT INTO toys VALUES (NULL, "Buzz Lightyear");
mysql> INSERT INTO toys VALUES (NULL, "Emperor Zurg");
mysql> INSERT INTO toys VALUES (NULL, "Bullseye");

mysql> SELECT * FROM toys;

+--------+----------------+
| toy_id | toy_name       |
+--------+----------------+
|      1 | Woody          |
|      2 | Buzz Lightyear |
|      3 | Emperor Zurg   |
|      4 | Bullseye       |
+--------+----------------+
4 rows in set (0.00 sec)
```


MySQL Example: PHP Database Querying

Now to the PHP side to query the database and display results on screen, toys1.php:

```
<h2>Toys for sale</h2>
 <pre>
 <?php
  // open database connection
  $connection
     = mysql_connect("localhost", "scmxx", "password");
  // select the database called "toyshop"
  mysql_select_db("toyshop", $connection)
     or die("Failed!");
  // run the query on the database
  $result
     = mysql_query("SELECT * FROM toys", $connection);

  // loop over each row in the result set
  while($row = mysql_fetch_row($result)) {

    // print the values of the attributes
    for($i=0; $i<mysql_num_fields($result); $i++)
      echo $row[$i]." ";
    echo "\n";
  }
  // close the database connection
  mysql_close($connection);
 ?>
 </pre>
```


MySQL Example: PHP Database Query Example Output

The output of this query is as follows:

Toys for sale

```
1 Woody
2 Buzz Lightyear
3 Emperor Zurg
4 Bullseye
```

MySQL Example: PHP Database Query Example Explained

Querying the database

In the PHP code there are **6 steps** involved in querying a database:

1. Connect to the DBMS (MySQL)
2. Select the database required
3. Run the query
4. Retrieve a row of results
5. Process the attribute values
6. Close the DBMS connection

1. Connect to the DBMS (MySQL)

- Connect to the DBMS using `mysql_connect()`
- parameters:
  - The **hostname** of the DBMS server to use
  - The **username** of a user having access to the database
  - The **password** of the user
- Return value
  - A connection handle to the DBMS
- If MySQL is installed on the same server as the scripting engine, we can use localhost as the hostname

```
$connection
= mysql_connect("localhost", "scmxx", "password");
```

For most of our work `localhost` will be `sentinel.cs.cf.ac.uk` or `blazon.cs.cf.ac.uk`

2. Select a database

- Select a database using `mysql_select_db()`
- Parameters:
  - The name of the database required
  - The connection handle to the DBMS (obtained in step 1)
- Return value
  - Nothing
  - Note the `die()` function to get out if we cant select the database — similar to Perl.

```
mysql_select_db("toyshop", $connection)
or die("Failed!");
```

Here we assume that `toyshop` is the name of database created.

3. Run a query

- Run a query on the database using `mysql_query()`
- Parameters:
  - An SQL query string
  - The connection handle to the DBMS (obtained in step 1)
- Return value
  - A result set handle - to retrieve the output (result set) of the query

```
$result
= mysql_query("SELECT * FROM toys",
              $connection);
```

or

```
$query = "SELECT * FROM toys";
$result = mysql_query($query, $connection);
```

4. Retrieve a row of results

- Retrieve a row of results using `mysql_fetch_row()`
- Parameters
  - The result set handle (obtained in step 3)
- Return value
  - The next row of the result set, or false when no more data is available.
- Use a while loop to retrieve the rows

```
while($row = mysql_fetch_row($result)) {
   // extract attribute values
   // (step 5) and process
}
```


5. Extract attribute values

- Extract attribute values using `mysql_num_fields()`
- Parameters:
  - The result set handle (obtained in step 3)
- Return value
  - The number of fields of the current row
- Use a for loop to retrieve the fields

```
for($i=0; $i<mysql_num_fields($result); $i++) {
    // process data here
    echo $row[$i]." ";
}
```


6. Close the DBMS connection

- Close the DBMS connection using `mysql_close()`
- parameters:
  - The connection handle to the DBMS (obtained in step 1)
- Return value
  - nothing

```
mysql_close($connection);
```


Command Line PHP

Command Line PHP on COMSC Computers

You can use Command Line PHP on:

- UNIX web server hosts (`sentinel` and `blazon`) **only**.
- Mac OS X machines

Uses of Command Line PHP

- Write standalone (non-web) PHP applications
- Debug/Test PHP scripts prior to Web Delivery — by-pass server

Running PHP Scripts from the Command Line

Two basic ways to do this:

- 1. Use the `php` command from the command line, **e.g.** `php my_script`

  There are many options that can applied with `php`.

  Type `man php` from the command line to find out more.

Running PHP Scripts from the Command Line (Cont.)

- 2. Make the php file an executable script
  - (Similar to Perl) Make the first line of the script
    `#!/usr/local/php/bin/php -q`

    (The `-q` option forces `php` to inhibit HTTP headers)

    The exact path may be different on various operating systems (possibly `/usr/bin/php`) — check with your systems administrator.

    The above path works on **Mac OS X**.
  - Use `#!/usr/local/bin/php` for `sentinel` and `blazon` UNIX/Linux Servers.
  - Make the script executable from the command line by using `chmod +x my_script`
  - Run the script by entering its name from the command line