An Introduction to PHP

Barry Cornelius
Computing Services, University of Oxford
Date: 26th November 2003; first created: 16th November 2003
http://users.ox.ac.uk/~barry/papers/
mailto:barry.cornelius@oucs.ox.ac.uk

1 Introduction

PHP is a server-side scripting language. When a visitor to a WWW page visits a page that is a PHP page, the WWW server gets a PHP processor to examine the page. The PHP processor will produce some HTML which is then shipped by the WWW server to the visitor's computer.

As well as the usual constructs that most languages have, in PHP, you can read from files, write to files and execute system commands; you can send mail; you can interrogate an IMAP server; you can access LDAP servers; you can process XML documents; you can do sophisticated mathematical calculations; you can access most of the popular database servers including Oracle, Sybase, Generic ODBC, MSQL and MySQL; and you can do many other things.

In this document, we look at the language and at some of the above possibilities.

2 What is PHP?

PHP was conceived sometime in the Autumn of 1994. Its manual says that 'PHP's syntax is borrowed primarily from C. Java and Perl have also influenced the syntax'.

Back in mid-1998, PHP's manual boasted that using 'a conservative estimate ... PHP is in use on 150000 sites around the world'. Nowadays, their WWW site (http://www.php.net/) cites a Netcraft Survey as saying '14,528,748 Domains, 1,328,604 IP Addresses'.

You can download PHP from http://www.php.net/. It is available in source form. You can also get binary versions for Windows 98/Me and Windows NT/2000/XP. Although they do not distribute UNIX/Linux binaries, most Linux distributions come with PHP. Binaries are also available on other WWW sites for AmigaOS, Mac OS X, Novell NetWare, OS/2, RISC OS, SGI IRIX 6.5.x and AS/400.

Instead, you could obtain a software bundle containing Apache's WWW server (`httpd`), PHP and MySQL (a database server). For example, there are many such bundles for Windows, including EasyPHP, FoxServ, sokkit (which evolved from PHPTriad) and Uniform Server. More details about these products are available at: http://www.easyphp.org/, http://www.foxserv.net/, http://www.sokkit.net/ and http://sourceforge.net/projects/miniserver/.

3 Simple PHP scripts

3.1 A simple example

Here is a simple example of a PHP script:

```
<HTML><BODY>
<?php
   $numrows = 3;
   for ($rownum = 0; $rownum<$numrows; $rownum++)
   {
?>
      <P>hello world</P>
<?php
   }
?>
</BODY></HTML>
```

Suppose that the file `hellothree.php` contains this code. It can be executed by passing it to a WWW server that understands PHP. Click on the following link to see what happens: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/hellothree.php

If a WWW server is configured to handle PHP, it will pass any file with a `php` extension to a PHP processor. The PHP processor knows it has to interpret the bits between the `<?php` and `?>` symbols. When given the above script, the PHP processor will output the following HTML and this is what is sent to the visitor's browser:

```
<HTML><BODY>
      <P>hello world</P>
      <P>hello world</P>
      <P>hello world</P>
</BODY></HTML>
```

3.2 Outputting HTML from PHP

Instead of switching between HTML and PHP, it is possible to stay in PHP, using the PHP command `echo` (or `print` or `printf`) to output HTML.

So the PHP script given above can instead be written as:

```
<?php
   echo "<HTML><BODY>\n";
   $numrows = 3;
   for ($rownum = 0; $rownum<$numrows; $rownum++)
   {
      echo "<P>hello world</P>\n";
   }
   echo "</BODY></HTML>\n";
?>
```

Go to the script at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/hellothreeecho.php

3.3 Using a form to supply data

Often the author of a WWW page will want to get data for his/her PHP script from the visitor. This can be done using a WWW form:

```
<HTML><BODY>
<FORM METHOD="POST" ACTION="helloform.php">
   How many times?
   <INPUT TYPE="text" NAME="numrows">
   <P>
   <INPUT TYPE="submit" VALUE="submit">
</FORM>
</BODY></HTML>
```

This WWW form uses a text field to ask the visitor to the WWW page for the number of rows. When the visitor clicks on the *submit* button, the `ACTION` attribute of the WWW form indicates that the `helloform.php` script will be executed. Because the text field of the WWW form has the name `numrows`, the contents of this text field can be accessed in `helloform.php` by using this name as an index to PHP's `$_POST` array. Here is a PHP script that does this:

```
<HTML><BODY>
<?php
   $numrows = $_POST["numrows"];
   for ($rownum = 0; $rownum<$numrows; $rownum++)
   {
?>
      <P>hello world</P>
<?php
   }
?>
</BODY></HTML>
```

Go to the WWW form at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/helloform.htm

3.4 Using your own functions

Before long, you will want to use your own functions. Here is an example of some code that declares and uses a function:

```
<?php
   function in_range($numrowspara)
   {
      if ($numrowspara>=1 && $numrowspara<=10)
         return true;
      else
         return false;
   }
   echo "<HTML><BODY>\n";
   $numrows = $_POST["numrows"];
   if (! in_range($numrows))
   {
      echo "<P>numrows is not within range: $numrows</P>\n";
      echo "</BODY></HTML>\n";
      return;
   }
   for ($rownum = 0; $rownum<$numrows; $rownum++)
   {
      echo "<P>hello world</P>\n";
   }
   echo "</BODY></HTML>\n";
?>
```

Go to: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/helloformcheck.htm to access a WWW form that actions the above PHP script.

3.5 Is it possible to execute an infinite loop?

Is it possible for a PHP script to execute an infinite loop and eat up lots of cpu-time on the WWW server? Here is a PHP script to try this:

```
<HTML><BODY>
<?php
   $count = 0;
   while (true) {
      $today = date("Y-m-d");
      $count++;
      print "count is: $count and today is: $today.<BR>";
   }
?>
</BODY></HTML>
```

Note: this PHP script uses a pre-defined function called `date`: this will be discussed later.

Go to the script at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/loop.php

The default is to allow a PHP script 30 seconds of cpu-time: it is possible to configure PHP so as to use a different value.

3.6 Using PHP from the command line

Although the code of a PHP file is normally triggered into action by a webserver, it is now possible to run standalone PHP programs, i.e., to run PHP programs from a Unix prompt or from a Windows command shell window.

So, if EasyPHP is being used on a Windows platform, you could type something like the following at a command prompt:

```
"C:\Program Files\EasyPHP1-7\php\php.exe" hellothree.php
```

and the PHP processor in the file `php.exe` would execute `hellothree.php`. It would output something like:

```
Content-type: text/html
X-Powered-By: PHP/4.3.3

<HTML><BODY>
   <P>hello world</P>
   <P>hello world</P>
   <P>hello world</P>
</BODY></HTML>
```

4 Calling pre-defined functions

4.1 Introduction

As has already been indicated, PHP allows you to access a lot of different things. Because of this, PHP has a large number of pre-defined functions. This is the main reason why the manual for PHP is enormous. Most of the sections of the manual you can ignore until you need the topic described by the section.

Here is a simple example of a call of a pre-defined function. The `date` function can return the current date (and time) in various formats:

```
<HTML><BODY>
<?php
   $today = date("Y-m-d");
   print "<CENTER>today is: $today.</CENTER>";
?>
</BODY></HTML>
```

Go to the script at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/today.php

4.2 Finding out what is available

Related functions are grouped together in the various sections of the PHP manual. Besides the core sections, there are sections which are optional. So what you get in your version of PHP depends on how it was built. There is a function called `phpinfo` that can be used to determine what is available in the version of PHP you are using:

```
<HTML><BODY>
<?php
   phpinfo();
?>
</BODY></HTML>
```

Go to the script at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/phpinfo.php

4.3 Including other files and executing OS commands

The next example contains two unrelated topics:

- A PHP script can include other files. This feature could be used for many purposes: it could be used to include a library of functions; it could be used to deploy a standard house style; and so on. This feature makes it easier to maintain code required by more than one PHP script. Note: it is possible to impose constraints on the directory in which an included file is located.

- A PHP script can get an OS command executed. It is possible to impose constraints on the directory in which these commands are located.

Here is a WWW form that asks a visitor for a command:

```
<HTML><BODY>
<FORM METHOD="POST" ACTION="system.php">
   <P>
   Enter the command to be executed:<INPUT TYPE="Text" NAME="command">
   <INPUT TYPE="Submit" VALUE="Execute command">
</FORM>
</BODY></HTML>
```

And here is a PHP script to execute the visitor's OS command:

```
<HTML><BODY>
<?php
   $command = $_POST["command"];
   require("head.htm");
   if ( $command == "DIR" )
      system("$command");
   else
      echo "<P>not executing: $command</P>";
   require("foot.htm");
?>
</BODY></HTML>
```

Go to the WWW form at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/system.htm

4.4 Accessing an LDAP server

LDAP (*Lightweight Directory Access Protocol*) provides a way of storing data such as Directory information (or *White Page* information).

The University of Durham's LDAP server enables anyone who has access to the WWW to find out the e-mail address and phone number of members of the University. Here is a WWW form that asks the visitor to the WWW page to type in part of a person's name:

```
<HTML><BODY>
<FORM METHOD="POST" ACTION="ldap.php">
   To search for someone from an LDAP server,
   type in the name of the server:
   <INPUT TYPE="text" NAME="server">
   <P>
   type in the search base:
   <INPUT TYPE="text" NAME="search_base">
   <P>
   type in part of their surname:
   <INPUT TYPE="text" NAME="part_name">
   <P>
   <INPUT TYPE="submit" VALUE="Run query">
</FORM>
</BODY></HTML>
```

And here is some PHP code to access the LDAP server. It uses six functions:

- `ldap_connect` and `ldap_bind` connects and binds to the LDAP server running on a given computer;
- `ldap_search` requests the LDAP server to search for entries containing a particular string;
- `ldap_get_entries` returns an array containing the entries that were found.
- `ldap_close` closes the connection to the LDAP server.

Here is the code of the `ldap.php` script:

```
<HTML><BODY>
<?php
   $server = $_POST["server"];
   $search_base = $_POST["search_base"];
   $part_name = $_POST["part_name"];
   $c_result = ldap_connect("$server");
   $b_result = ldap_bind($c_result);
   $s_result = ldap_search($c_result, "$search_base", "cn=*$part_name*");
   $info = ldap_get_entries($c_result, $s_result);
   $numrows = $info["count"];
   if ( $numrows == 0 ) {
      echo "<P>There is no entry with a name of $part_name</P>";
      echo "</BODY></HTML>";
      ldap_close($c_result);
      exit;
   }
?>
   <TABLE BORDER="1">
<?php
   for ($rownum = 0; $rownum<$numrows; $rownum++) {
?>
      <TR>
         <TD>
            <?php echo $info[$rownum]["cn"][0]; ?>
         </TD>
         <TD>
            <?php echo $info[$rownum]["ou"][0]; ?>
         </TD>
         <TD>
            <?php echo $info[$rownum]["telephonenumber"][0]; ?>
         </TD>
         <TD>
            <?php
               echo "<A HREF=mailto:";
               echo $info[$rownum]["mail"][0];
               echo ">";
               echo $info[$rownum]["mail"][0];
               echo "</A><BR>";
            ?>
         </TD>
      </TR>
<?php
   }
?>
   </TABLE>
<?php
   ldap_close($c_result);
?>
</BODY></HTML>
```

Go to the WWW form at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/ldap.htm

4.5 Sending electronic mail

Here is a WWW page containing a WWW form to obtain the details of an e-mail message from a visitor to this WWW page:

```
<HTML><BODY>
<FORM METHOD="POST" ACTION="mail.php">
   <P>
   Enter the name of the sender:    <INPUT TYPE="Text" NAME="from">
   <P>
   Enter the name of the recipient: <INPUT TYPE="Text" NAME="to">
   <P>
   Enter the subject of the message:<INPUT TYPE="Text" NAME="subject">
   <P>
   Enter the text of the message:   <INPUT TYPE="Text" NAME="message">
   <P>
   Enter the name of the server:    <INPUT TYPE="Text" NAME="server">
   <P>
   <INPUT TYPE="Submit" VALUE="Send message">
</FORM>
</BODY></HTML>
```

And here is a PHP script to send an e-mail message:

```
<HTML><BODY>
<?php
   $to      = $_POST["to"];
   $subject = $_POST["subject"];
   $message = $_POST["message"];
   $from    = $_POST["from"];
   $server  = $_POST["server"];
   ini_set("SMTP", "$server");
   $result = mail("$to", "$subject", "$message", "From: $from");
   print "<CENTER>result is: $result.</CENTER>";
?>
</BODY></HTML>
```

Go to the WWW form at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/mail.htm

Note: it is possible to configure PHP to disable any attempt to send e-mail.

4.6 Manipulating XML documents

XML is a markup language that allows data to be described using your own tags.

PHP comes with an XML parser that allows you to write PHP code that walks through an XML document. It generates events for each part of the document, generating an event for each start tag, for each end tag, and for each occurrence of characters appearing between a start tag and an end tag. In a PHP program, you can nominate functions to be called when these events occur. Note that each part of the XML document is visited only once. If instead you want the ability to wander around an XML document, it is possible to put the representation of the XML document into an array. For more details about parsing XML documents from PHP, look at my document 'Processing XML using PHP': http://www.dur.ac.uk/barry.cornelius/papers/xml.processing.using.php/

PHP provides an extension that can do XSL transformations. In order to use this extension, the person installing PHP needs to build it with a library that implements XSLT. Suppose we have a file (`consumables.xml`) that contains the following XML document:

```
<?xml version="1.0"?>
<consumables>
   <product>
      <category>cassette tape</category>
      <item>DC6150 cartridge</item>
      <price>9.50</price>
   </product>
   <product>
      <category>cassette tape</category>
      <item>DC4-60</item>
      <price>6.00</price>
   </product>

   ...

   <product>
      <category>writeable media</category>
      <item>writeable CD</item>
      <price>1.75</price>
   </product>
</consumables>
```

Also, suppose the following XSL instructions have been stored in a file called `consumables.xsl`:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="consumables">
   <html>
   <body>
   <table>
   <tr bgcolor="yellow">
   <td>category</td>
   <td>item</td>
   <td>price</td>
   </tr>
      <xsl:apply-templates/>
   </table>
   </body>
   </html>
</xsl:template>
<xsl:template match="product">
   <tr>
   <td><xsl:value-of select="category"/></td>
   <td><xsl:value-of select="item"/></td>
   <td><xsl:value-of select="price"/></td>
   </tr>
</xsl:template>
</xsl:stylesheet>
```

Here is a PHP script that takes the `consumables.xml` file and transforms it using the rules in the `consumables.xsl` file:

```
<HTML><BODY>
<?php
   chdir('/users/dcl0bjc/public_html/papers/phpintro/code');
   $xsltproc = xslt_create();
   $html = xslt_process($xsltproc,'consumables.xml','consumables.xsl');
   if (!$html) {
      die('XSLT processing error:' . xslt_error($xsltproc));
   }
   xslt_free($xsltproc);
   echo $html;
?>
</BODY></HTML>
```

Go to the script at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/xslt.php

My document 'Processing XML using PHP' gives more details about performing XSL transformations in PHP. It is available at: http://www.dur.ac.uk/barry.cornelius/papers/xml.processing.using.php/

5 Session handling

We may want a WWW page that can retain information from one visit to the WWW page to the next. What we can do is to indicate in the PHP script that we want to use session variables. Then, when a visitor visits our PHP script, they will be assigned a unique id called the *session id*. Either this can be stored in a cookie in the visitor's browser or the author of the PHP script can arrange for it to be propagated in the URL.

Suppose we want to collect the visitor's name and then greet him/her with this name whenever they visit the WWW page. So we will use a session variable to store the visitor's name. The PHP script can start by checking whether the session variable is set. If it is not set, the PHP script can display a WWW form to collect this information from the visitor. That gets us to the following code. Suppose this code is in a file called `intro.php`:

```
<?php
   session_start();
   if (isset($_SESSION["visitors_name"])) {
      $visitors_name = $_SESSION["visitors_name"];
      echo "Hello $visitors_name<br>\n";
   }
   else {
?>
      <FORM METHOD="post"   ACTION=...">
      Type in your name:
      <INPUT  TYPE="text"     NAME="visitors_name"/>
      <INPUT  TYPE="submit"  VALUE="Register"/>
      </FORM>
<?php
   }
?>
```

The ACTION attribute in the above WWW form is incomplete. So what script do we want to execute when the visitor clicks on the *Register* button? What we could do is to invoke the same script provided we alter the script so that one of the first things it does is to check whether it was triggered into action by the WWW form. So, we can add some code to check whether the appropriate $_POST variable is set, and, if it is, the code can then set the session variable:

```
   if (isset($_POST["visitors_name"])) {
      $_SESSION["visitors_name"] = $_POST["visitors_name"];

   }
```

The resulting script is given below. It also includes some code that counts the number of times the visitor has visited this WWW page.

```
<?php
   session_start();
   if (isset($_POST["visitors_name"])) {
      $_SESSION["visitors_name"] = $_POST["visitors_name"];
      $_SESSION["num_visits"] = 0;
   }
   if (isset($_SESSION["visitors_name"])) {
      $visitors_name = $_SESSION["visitors_name"];
      $_SESSION["num_visits"]++;
      $num_visits = $_SESSION["num_visits"];
      echo "Hello $visitors_name on visit number $num_visits<br>\n";
      echo "<a href=\"intro.php\">click here</a>\n";
   }
   else {
?>
      <FORM METHOD="post"   ACTION="intro.php">
      Type in your name:
      <INPUT  TYPE="text"     NAME="visitors_name"/>
      <INPUT  TYPE="submit"  VALUE="Register"/>
      </FORM>
<?php
   }
?>
```

Go to the script at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/intro.php

As mentioned above, there are two ways in which a session id can be propagated. The manual says that ‘Cookies are optimal’, but because the visitor can disable support in his/her browser for cookies, PHP provides the alternative way of embedding a session id directly into a URL. The manual points out that ‘URL based session management has additional security risks compared to cookie based session management. Users may send an URL that contains an active session ID to their friends by email or users may save an URL that contains a session ID to their bookmarks and access your site with the same session id always, for example.’

6 Object oriented programming

PHP has some support for object-oriented programming. As shown below, it is possible to declare a class that has variables, constructors and functions.

Here is a file (called `person.php`) that declares a class called `Person`:

```
<?php
   class Person {
      var $iName, $iAge;
      function Person($pName, $pAge) {
         $this->iName = $pName;
         $this->iAge = $pAge;
      }
      function getName() {
         return $this->iName;
      }
      function setName($pName) {
         $this->iName = $pName;
      }
      function getAge() {
         return $this->iAge;
      }
      function setAge($pAge) {
         $this->iAge = $pAge;
      }
      function hasSameAge($pPerson) {
         return $this->iAge==$pPerson->getAge();
      }
      function display() {
         echo "<P>$this->iName’s age is $this->iAge</P>\n";
      }
   }
%>
```

And here is a file (called `persontest.php`) that declares and uses objects that are of this class:

```
<?php
   require("person.php");
   // tom and dick
   $tFirstPerson = new Person("tom", 25);
   $tFirstPerson->display();
   $tSecondPerson = new Person("dick", 24);
   $tSecondPerson->display();
   $tSameAge = $tFirstPerson->hasSameAge($tSecondPerson);
   if ($tSameAge)
      echo "<P>they have the same age</P>\n";
   else
      echo "<P>they have different ages</P>\n";
   // fred and barney
   $tFirstPerson = new Person("fred", 55);
   $tFirstPerson->display();
   $tSecondPerson = new Person("barney", 55);
   $tSecondPerson->display();
   $tSameAge = $tFirstPerson->hasSameAge($tSecondPerson);
   if ($tSameAge)
      echo "<P>they have the same age</P>\n";
   else
      echo "<P>they have different ages</P>\n";
%>
```

Go to the script at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/persontest.php

Other possibilities include:

- using the `::` symbol to access standalone functions of a class (called *class methods* in Java);
- using the `extends` symbol to derive a class from an existing class (single inheritance);
- using the `parent::` symbols to access members of a superclass.

In a paper entitled ‘The Object Oriented Evolution of PHP’, Zeev Suraski explains some of the limitations of PHP 4 and outlines what is likely to be provided in PHP 5.

Some of the limitations of PHP 4 are:

- Objects are passed to functions by value and not by reference.
- There is no *data encapsulation*, i.e., a variable of a class cannot be made private to the class. So any client of the `Person` class can alter the `$iName` and `$iAge` variables themselves (and can ignore the `setName` and `setAge` functions):

  ```
  0560: $tFirstPerson->iAge = 24;
  ```

- Although you can access a *class method*, it is not possible to access *class variables*.
- There is no support for multiple inheritance.
- There are no constructs for exception handling (`try`/`throw`/`catch`).

Many of the above limitations are being removed in PHP 5. Beta 2 of PHP 5.0.0 has been released. Probably, the most up-to-date statement about the OO features of PHP 5 can be found at: http://www.php.net/zend-engine-2.php. Here is a summary of some of the new features in PHP 5:

- public, private and protected members (both variables and functions);
- interfaces, abstract classes and abstract methods;
- a `final` keyword for members (variables and functions) and for classes;
- PHP 5 also introduces:
  - a `__clone` function for cloning,
  - a `__construct` function for a constructor,
  - a `__destruct` function for a destructor,
  - a `__toString` function that allows overloading of object to string conversions;
- exception handling (that is similar to that of other programming languages).

Suraski’s paper is at: http://www.zend.com/images/press/Feb_2003-4_Zeev_PHP5.pdf. Another useful paper is at: http://www.zend.com/engine2/ZendEngine-2.0.pdf

7 Using MySQL

7.1 What is MySQL?

MySQL is software that is a database server. For more details, go to: http://www.mysql.com/.

7.2 Accessing a MySQL server from PHP

We now move on to produce WWW pages that can manipulate databases. To do this, all we have to do is call the appropriate functions of PHP that enable us to pass SQL statements to a MySQL database.

The PHP scripts that follow use these functions:

- `mysql_connect` which connects to a MySQL server running on some computer supplying a username and a password to the MySQL server. For example:

  ```
  $c_result = mysql_connect("dbserver.site.com", "dxy3fab", "ind1g0");
  ```

  Typically, the MySQL server will be configured so that the username and password are required when the developer wishes to change a database but they can be empty strings when he/she only wishes to inspect the database:

  ```
  $c_result = mysql_connect("dbserver.site.com", "", "");
  ```

- `mysql_select_db` which chooses a database:

  ```
  $s_result = mysql_select_db("Pdcl0bjc_prices", $c_result);
  ```

- `mysql_query` which executes a given SQL query:

  ```
  $q_result = mysql_query("SELECT * FROM consum WHERE price<1.0", $c_result);
  ```

- `mysql_num_rows` which finds out how many rows there are in the result of a query:

  ```
  $numrows = mysql_num_rows($q_result);
  ```

- `mysql_result` which returns the value of a given column of a given row of the result of a query:

  ```
  print mysql_result($q_result, $rownum, "price");
  ```

7.3 Providing a WWW page to query a table

Suppose we have created a database called `Pdcl0bjc_prices`; that a table of this database is called `consum`; and that this table has columns called `ID`, `goods` and `price`. The idea is that each row of this table contains the details about an item of consumables that is for sale; the columns headed `goods` and `price` contain the description and the price of the item, and the column headed `ID` contains a unique number for this item.

If we want to supply a WWW page that can be used to query this table, we need first to obtain from the person visiting the page the name of the item of consumables in which he/she is interested. We can do this by providing a WWW page that contains:

```
<HTML><BODY>
<FORM METHOD="POST" ACTION="pricesquery.php">
   To search for an item of consumables,
   type in the name of the MySQL server:
   <INPUT TYPE="Text" NAME="server">
   <P>
   and type in part of the name of the item (e.g., Series IV):
   <INPUT TYPE="Text" NAME="goods">
   <P>
   <INPUT TYPE="Submit" VALUE="Run query">
</FORM>
</BODY></HTML>
```

Go to the WWW form at: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/pricesquery.htm

When the visitor to the page clicks on the *Run query* button, the `pricesquery.php` script will be executed. The code of this PHP script appears below.

This script first uses `mysql_connect` to attempt to connect to the MySQL server. It then uses `mysql_select_db` to choose a database called `Pdcl0bjc_prices`. It then assigns to the `SQLQuery` variable a string containing the characters:

```
SELECT * FROM consum WHERE goods LIKE ’%$goods%’
```

i.e., a string containing the query in which the visitor is interested. It then calls the function mysql_query to send this SELECT statement to the database Pdcl0bjc_prices. If this is successful, the variable q_result now contains a pointer to a temporary table that has just been created; it contains only those rows satisfying the query.

The call of mysql_num_rows finds out how many rows are in this table. And if this is not zero, the script generates an HTML table containing the rows of data that are in this table. It does this by using a for command to wander through each row of the table. For each row, it generates HTML containing the contents of the ID, goods and price columns.

```
<HTML><BODY>
<?php
   $server = $_POST["server"];
   $goods  = $_POST["goods"];
   $c_result = mysql_connect($server, "", "");
   $s_result = mysql_select_db("Pdcl0bjc_prices", $c_result);
   $SQLQuery = "SELECT * FROM consum WHERE goods LIKE '%$goods%'";
   $q_result = mysql_query($SQLQuery, $c_result);
   $numrows = mysql_num_rows($q_result);
   if ( $numrows == 0) {
      echo "<p>There are no consumables with a name like $goods</p>";
   }
   else {
?>
      <TABLE BORDER="1">
      <?php
      for ($rownum = 0; $rownum<$numrows; $rownum++) {
      ?>
         <TR>
            <TD ALIGN="right">
               <?php echo mysql_result($q_result, $rownum, "ID"); ?>
            </TD>
            <TD>
               <?php echo mysql_result($q_result, $rownum, "goods"); ?>
            </TD>
            <TD BGCOLOR="yellow" ALIGN="right">
               <?php printf("%01.2f", mysql_result($q_result, $rownum, "price")); ?>
            </TD>
         </TR>
      <?php
      }
      ?>
      </TABLE>
<?php
   }
?>
</BODY></HTML>
```

7.4 Providing a WWW page to insert a new row

We can proceed in a similar way if we want to insert a new row in this table. Besides a box asking for the name of the item, the HTML form also has a box asking for details of its price. And, as we wish to change the database, we will need to supply the MySQL username and password that is appropriate for this database. So the form also has boxes asking for a username and a password:

```
<HTML><BODY>
<FORM METHOD="POST" ACTION="pricesinsert.php">
   To insert a new item of consumables,
   type in the following information:
   <P>
   name of MySQL server: <INPUT TYPE="Text"     NAME="server">
   <P>
   your username:        <INPUT TYPE="Test"     NAME="username">
   <P>
   your password:        <INPUT TYPE="Password" NAME="password">
   <P>
   name of the item:     <INPUT TYPE="Text"     NAME="goods">
   <P>
   price, e.g. 18.45:    <INPUT TYPE="Text"     NAME="price">
   <P>
   <INPUT TYPE="Submit" VALUE="Update consumables">
</FORM>
</BODY></HTML>
```

Go to the WWW form: http://www.dur.ac.uk/barry.cornelius/papers/phpintro/code/pricesinsert.htm

When the person using this WWW form clicks on the *Update consumables* button, the PHP script in the file `pricesinsert.php` gets executed. This time, because it wants to update the database, the script's call of `mysql_connect` needs to provide a username and a password. And, this time, because the *query* is actually an `INSERT`, the call of `mysql_db_query` returns either true or false depending on whether the `INSERT` was successful or not. The script outputs an appropiate message:

```
<HTML><BODY>
<?php
   $server   = $_POST["server"];
   $username = $_POST["username"];
   $password = $_POST["password"];
   $goods    = $_POST["goods"];
   $price    = $_POST["price"];
   $c_result = mysql_connect("$server", "$username", "$password");
   $s_result = mysql_select_db("Pdcl0bjc_prices", $c_result);
   $SQLQuery = "INSERT INTO consum SET goods='$goods', price=$price";
   $q_result = mysql_query($SQLQuery, $c_result);
   if ( $q_result == 0) {
      echo "<P>There seems to be a problem with updating the table</P>";
   }
   else {
      echo "<P>The table has been updated.</P>";
   }
?>
</BODY></HTML>
```

8 PHP Applications

8.1 Introduction

Once you have a PHP processor installed, there are plenty of pieces of software written in PHP that you can download from the WWW. There are a number of different lists of such software. The URLs of some of these lists are given at: http://www.php.net/links.php

Here we will look at two PHP applications.

8.2 IMP

At the University of Durham, one of the e-mail products we support is called `IMP`. This product is accessed using a WWW browser. It provides the visitor with a login screen and, once he/she has authenticated, IMP uses IMAP to contact the machine containing their e-mail. So the product enables a member of the University to handle their e-mail, e.g., to look at their messages, store messages, reply to messages, and compose messages. And they can do this on any machine anywhere in the world provided it has a WWW browser. All the code of IMP is written in PHP.

For more details about IMP, go to: http://www.horde.org/imp/

8.3 JpGraph

JpGraph is a collection of PHP code for producing graphs, pie-charts, bar-charts, etc. The WWW site for JpGraph is at: http://www.aditus.nu/jpgraph/. The developers say that JpGraph 'makes it easy to draw both "quick and dirty" graphs with a minimum of code and complex professional graphs which requires a very fine grain control'. The product comes with lots of examples and useful documentation.

9 Conclusions

- PHP is a powerful scripting language for performing server-side activities.
- It has a very large number of pre-defined functions providing access to many different features: some of these have been demonstrated in this document.
- In particular, it enables easy access to data stored in a database server (such as a MySQL server).
- Not only can code be created in-house, but there is also a large amount of third party code written in PHP.