# 8 The Ghost project

## 8.1 A PHP web site generator project

The ability to write short programs in C to automate tedious tasks or to do things that would otherwise take hours of fiddling about with cumbersome tools such as doing mail-merge, is one on the things you will be most pleased you have learned how to do. This project is such a time-saver. Ghost is a lightweight PHP generator for you to customise.

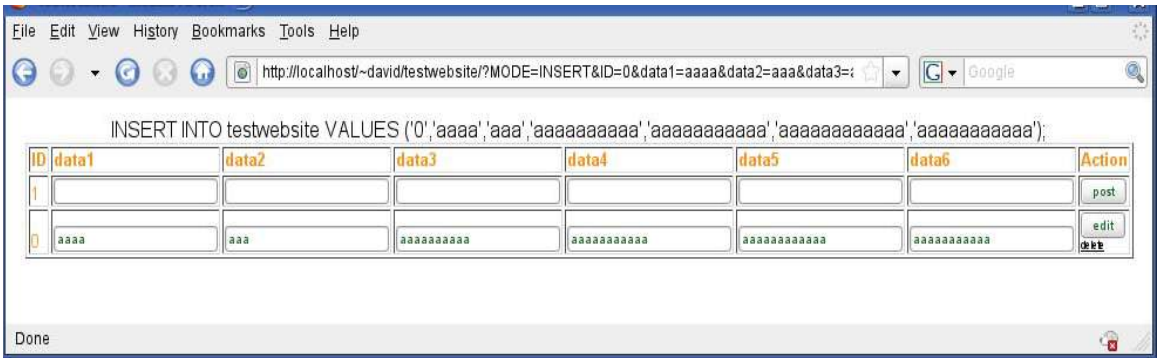
If you find yourself having to build PHP web sites all the time, a quick way to generate all the parameter-passing, decoding, forms building and database management code in one step would be useful. Tools like Ruby on Rails offer such functionality but are infinitely more complex to set up and run and you end up with needing to learn yet another language to go any further.

Probably the best way to start with this tool is to compile and run it. Unzip the ghost.zip source into your public_html folder which creates a folder called ghost. The Makefile contains a target **g1** that compiles and links ghost. So go to **public_html/ghost** and type: **make g1**.

To run the site generator type:

- **./ghost testwebsite data1 data2 data1 data3 data4 data6 data6**
- This will create:
- a folder **public_html/testwebsite**
- a mysql database table called **testwebsite** with text fields data1 data2 data1 data3 data4 data6 data6
- a **testwebsite.css** file
- empty **header.html** and **footer.html** pages
- **index.php** that demonstrates a form handling entry, edit & update, and delete to the database table for the data items specified.

In a browser what you see is this at http://localhost/~yourname/testwebsite




The idea behind this is that all the mechanical bits to create and manage the form content are done and can be customised. This screen shot shows the result of submitting one record. The top row is for entering new data, the lower row(s) allow editing or deleting of records. It is a framework that allows you to take and use parts in your own website design.

Let us examine this code in sections.

The first section declares the required data and creates the folder and CSS file.

```
int main(int argc, char *argv[])
{
        FILE *out = NULL;
        MYSQL *conn = NULL;
        MYSQL_RES *result = NULL;
        MYSQL_ROW row;
        MYSQL_FIELD *field;
        char SQL[STRINGSIZE]="";
        char BIT[STRINGSIZE]="";
        char SQLINSERT[STRINGSIZE]="";
        char SQLUPDATE[STRINGSIZE]="";
        char SQLDELETE[STRINGSIZE]="";
        int rc=0, i=0, num_fields=0;

        //CREATE DIRECTORY////////////////////////////////
        sprintf(BIT,"mkdir ~/public_html/%s",argv[1]);
        system(BIT);
        //BUILD CSS/////////////////////////////////////
        sprintf(BIT,"../%s/%s.css",argv[1],argv[1]);
        out = fopen(BIT,"w");
        fprintf(out,"table {width:800;vertical-align:top;}\n");
        fprintf(out,"th {font-size:80%;color:#fd9208;vertical-align:bottom;text-align:left;}\n");
        fprintf(out,"td {font-size:80%;color:#fd9208;vertical-align:bottom;text-align:left;}\n");
        fprintf(out,"input {font-size:80%;color:#196419;}\n");
        fprintf(out,"a  {font-size:65%;color:#000000;}\n");
        fclose(out);
```

Next the header.html and footer.html files are generated. These files is loaded by the PHP file and could be used as a generic common header and footers. The CSS file is referenced from the header.html file.

```
//BUILD HEADER HTML//////////////////////////////////
sprintf(BIT,"../%s/head.html",argv[1]);
out = fopen(BIT,"w");
fprintf(out,"<html>\n");
fprintf(out,"<head>\n");
fprintf(out,"<link rel=\"stylesheet\" type=\"text/css\"href=\"%s.css\" />\n",argv[1]);
fprintf(out,"<title>%s</title>\n",argv[1]);
fprintf(out,"</head>\n");
fprintf(out,"<body>\n");
fprintf(out,"<center>\n");
fprintf(out,"<table><tr><td>\n");
fprintf(out,"</td></tr></table>\n");
fclose(out);
//BUILD FOOT HTML//////////////////////////////////
sprintf(BIT,"../%s/foot.html",argv[1]);
out = fopen(BIT,"w");
fprintf(out,"</body></html>\n");
fclose(out);
```

Next we create the data base.

```
//OPEN DATABASE//////////////////////////////////////
conn = mysql_init((MYSQL *) 0);
mysql_options(conn,MYSQL_READ_DEFAULT_GROUP,"mysqlcapi");
mysql_real_connect(conn, "localhost","","","test",0, NULL, 0);
//CREATE TABLE//////////////////////////////////////
sprintf(SQL,"drop table if exists %s",argv[1]);
rc = mysql_query(conn,SQL);
sprintf(SQL,"create table %s (ID varchar(255)",argv[1]);
for(i=2; i < argc; i++)
{
        sprintf(BIT,",%s varchar(255)",argv[i]);
        strcat(SQL,BIT);
}
strcat(SQL,");");
rc = mysql_query(conn,SQL);
```

The complicated part starts now, of generating a php script. The best way to understand this is to examine the actual output of the program when we view the source of the page in the browser.

The top row is a form with a text box for each column defined in the table generated by running the ghost program.

```
<tr>
<form><input type='hidden' name='MODE' value='INSERT'>
<td><input type=hidden name='ID' value='1'>1 </td>
<td><input type=text name='data1'></td>
<td><input type=text name='data2'></td>
<td><input type=text name='data3'></td>
<td><input type=text name='data4'></td>
<td><input type=text name='data5'></td>
<td><input type=text name='data6'></td>
<td><input type='submit' value='post'></td>
</form>
</tr>
```

For each row in the table we now generate a form allowing editing of the data and an anchor link to do a delete operation.

```
<tr>
<form><input type='hidden' name='MODE' value='UPDATE'>
<td><input type=hidden name='ID' value='0'>0</td>
<td><input type=text name='data1' value='aaaaaaaaaaa'></td>
<td><input type=text name='data2' value='aaaaaaaaaaaaaaa'></td>
<td><input type=text name='data3' value='aaaaaaaaaaaaaaa'></td>
<td><input type=text name='data4' value='aaaaaaaaaaaaa'></td>
<td><input type=text name='data5' value='aaaaaaaaaaaaaaa'></td>
<td><input type=text name='data6' value='aaaaaaaaaaa'></td>
<td><input type='submit' value='edit'></form>
<a href='?ID=0&MODE=DELETE'>delete</a></td
</tr>
```

Close examination of the file index.php will allow you to see where all this happens, and to work backward to find where in the ghost.c source code this PHP code is generated. A good idea is to use a highlighter pen on a printout as we are embedding a language (HTML) inside another language (PHP) which is in turn inside another language so very very careful use is made of the escape characters '\ 'to express quotation marks both single and double where necessary to make it all work. This may seem complex – but the speedy prototyping that ghost permits makes it worthwhile to spend time customising the C code so the PHP that you want and the database you want come out the way you want it.

Here is the part of the PHP file index.php which generates the edit or delete rows. The static HTML is highlighted and the other parts are inserted by MySQL PHP function calls.

```
$result = mysql_query("select * from testwebsite");
while($row = mysql_fetch_array($result))
{
        echo "<tr>";
        echo "<form><input type='hidden' name='MODE' value='UPDATE'>";
        for($i=0;$i < mysql_num_fields($result); $i++)
        if($i==0)
                echo "<td><input type=hidden name='" .
                        mysql_field_name($result,$i) . "' value='" .
$row[mysql_field_name($result,$i)] .
                "'>".
                $row[mysql_field_name($result,$i)] .
                "</td>";
        else
                echo "<td><input type=text name='" .
                        mysql_field_name($result,$i) .
                        "' value='" .
                        $row[mysql_field_name($result,$i)] .
                                "'></td>";
        echo "<td><input type='submit' value='edit'></form>";
        echo "<a href='?ID=" .
                $row[mysql_field_name($result,0)] .
                "&MODE=DELETE'>delete</a>";
        echo "</td></tr>";
}
mysql_free_result($result);
mysql_close($con);
```

As you can see a great deal of tedious and repetitive work has been automated. You can move on by modifying the PHP code or go deeper to customise the C program which generates all of it.

I personally use ghost frequently to save time on site-building and this is why I wrote it. I got bored making mistakes writing virtually identical code to decode HTML forms and populate or update databases.