Some things to watch out for when using PHP and Javascript when building websites

Les Hatton

10 Sep 2003

1 PHP

PHP is a C-like language which evolved from Perl scripts originally produced by Rasmus Lerdorf and intended to make the construction of Web sites with dynamic content in general and access to SQL databases in particular easier to accomplish. It accomplishes this task remarkably well. It has gone through three major revisions under the care of many other developers and represents today a formidably good example of just how effective Open Source development can be.

There remain a few gotchas in practice which have got me at least once so I thought I would document them as a service to other users.

1.1 Linguistic problems

1.1.1 $$ in a variable name

All PHP variables use $ to precede them. If you inadvertently put $$ in front of them, PHP says nothing and you do not access the object contents, (my guess is you access the object address). Its typographically hard to see and it fails silently. Beware - this has caught me at least three times.

1.1.2 General and local names

If you use session management in PHP (and you should as it makes handling variables with a wider scope than a single web page much easier), watch out for the following. Arrays in PHP are indexed by naming the element, for example. Consider the following statement:-

```
$hello = $HTTP_SESSION_VARS['hello'];
```

On the face of it, this initialises a local variable called $hello from the contents of the array element called 'hello'. However, PHP has an initialisation option, (they are usually set in /etc/php.ini or something similar). which will globalise these for convenience making $hello synonymous with the array element. In other words, if globalisation is on:-

```
$hello = 0;
```

is actually synonymous with:-

```
$HTTP_SESSION_VARS['hello'] = 0;
```

However, if globalisation is not on, it only initialises a local copy. The rub is that if you use PHP on your ISP, you have no control over this and in general don't know whether its on or off, so you may find yourself initialising variables you did not intend to. I have done this on several occasions.

Always choose your local variables to have a different name from the array index.

1.1.3 Correct way of using session_start()

I didn't read the documentation but got no warning. The correct way of using session_start() and session_id() is as follows. If you want to start a session for the first time but need the session ID:-

```
session_start();
$SESSID = session_id();
```

If you want to restart a session for which you already have the session id.

```
session_id($SESSID);
session_start();
```

But don't do what I did:-

```
session_start($SESSID);
```

(Yes, I know I didn't RTFM, but a warning would have been nice.)

1.2 File permissions

If you use PHP to open, read, write and close files using its fopen(), ... functions, be very careful about permissions. On the server at your ISP, you don't have any control over the username adopted by PHP when it runs and creates files on behalf of your web-site visitors. Watch out for the following:-

- Put any directories into which your PHP scripts create and use files above your web-site directory, (normally public_html and linked to www). If the server has been set up correctly, for example, Apache, it will stop anybody accessing these files directly although they will be able to access them through your scripts as you intend.
- Make these directories 777 access so everybody can get at them, (through your web scripts of course). The fact that they are 777 doesn't mean to say that people can access them directly - the server should prevent this as described above.
- Be careful not to inadvertently edit the files created and accessed through your PHP scripts directly when as webmaster you log onto your web-site. If you do, your PHP scripts will no longer be able to access them as the file permissions will have been changed to reflect your direct access as webmaster.

2 Javascript

Javascript is a bit of an abomination in comparison with some wondrous bugs. For example, you can embed in front of your forms in html a nice little Javascript checking function, something like:

```
//
//   Script to check for blanks.
//
     function  isblank(s)
     {
//        Check string has no blanks.

          for( var i = 0; i < s.length; ++i )
          {
               var  c = s.charAt(i);
               if ( (c != ' ') || (c != '\n') || (c != '\t') )
               {
                    return    false;
               }
          }

          return    true;
     }
//
//   Verifying function.
//
     function  verify_form(f)
     {
          var  msg;
          var  errors    = "";
//
//        Loop through each field in the form.
//
          for( var i = 0; i < f.length; ++i )
          {
               var  e = f.elements[i];
//
//             Check fields.
//
               if ( (e.value == "") || isblank(e.value) )
               {
                    errors    +=   "\n- The field " +
                                        e.description       +
                                        " must not be null and"  +
                                        " must not contain blanks, tabs or newlines.";
               }
          }

          if (errors)
```

```
          {
               msg  = "There are errors in this filename specification:-\n\n" + errors;

               alert(msg);
               return    false;
          }
          else
          {
               return    true;
          }
     }
</script>

     <FORM METHOD="POST" NAME=array_length onSubmit=
     "this.form_value.description = 'Array length';
      this_form_value.numeric = true;
      this_form_value.min = 0.0;
     return    verify_form(this);"
     ACTION="<?php echo "$GDFSPATH/gdfs_accessarray.php";?>">
     <TABLE WIDTH=100%>
...
```

The idea here is that Javascript checks for non-numeric values or values less than zero when the client fills submits the form. Unfortunately, it only checks the first character, so Javascript not only thinks that 6.0 is a numeric value, which is comforting, but it also thinks that "6.;rm *" is also numeric, which is not. Sigh.

Always make sure that you check all values on the server using PHP. Don't just rely on Javascript checking running in the client's browser. Its not very reliable unfortunately.

2.1 Debugging tip

You can use error_log() in PHP when testing your scripts on your own machine. However, when you upload them, you may not be able to use error_log() and you may have debugging problems for some of the reasons above even if everything appears fine on your test machine. Its a bit roundabout, but you can use mail() in PHP to e-mail you variable values to help with a tricky problem. Just make sure you restrict access to your shop while you are doing this. One nice way of doing this is to use a variable which indicates whether the site is open and a test variable, say $TEST_MODE which restricts access to your top-level page unless it is set to the correct value. If you pass this as a GET variable, i.e. embedded in the URL, then you can pass this test value to enter your site knowing that external users cannot reach it, (because they will normally access the site through a link which passes a non-test mode). In PHP something like the following works well:-

```
...
```

```
if ( ($SITE_STATE == "CLOSED") && ($TEST_MODE != "TEST") )
{
/*
 * Skip to "Site is closed, sorry." page
 */
}
...
```

I will add to this document as my experience grows. So far apart from the usual logic errors, the above is all I have found and PHP is an attractive way of developing web-sites for dynamic content such as shops.