THE PHP SCRIPT

This is one of the most important parts of the process. Without this script our interface would not run. Here we're going to be reading actual files in a folder at a website. This way, dropping new files into the folders, or creating new folders will automatically update our Flash file browser. We accomplish this using a PHP script, which we call, pass a few parameters to, and it returns to us a variable string of filenames, types and sizes.

The parameters that the script expects are:

- `matchDir` – this contains a relative directory from where we would like the file listings. If we specify images, for example, then all files returned by the script will be from the `/images` directory.

- `matchExt` – the file extensions that we would like to display. If we only wanted GIF files to be sent back, then we would send `matchExt` as `gif`.

When we make a call to the script (which is called `dirstuff.php`), we pass parameters with a `LoadVars` object, like so:

```
myLoadVars.load(base +
➥ "dirstuff.php?matchDir=Tutorials&matchExt=*");
```

In this instance, base is a variable that points to the root of the Web server. So, base could be http://www.mysite.com/.

If we want all files to be returned, then matchExt can be set to *, or can be completely left out. When the files are returned, they'll be sent back in a string of text, which is divided into name/variable pairs, like so:

```
&file0=.&fsize0=512&file1=..&fsize1=512&file2=High Score
➥ List.link&fsize2=48&file3=Hi Score
➥ Files.zip&fsize3=159583&file4=FlashintheCan Game
➥ Talk.pop&fsize4=56&file5=Motion and Collision slides.swf&fsize5=15499&
```

Each file is passed as a name, file0, file1, file2, and a size, fsize0, fsize1, fsize2. So, in this example, file 5 is called Motion and Collision slides.swf and it's 15499 bytes in size.

Our PHP script is as follows:

```
<?php

$sitebase = "mysite/";

if (!isset($matchDir)) $matchDir = "./";
if (!isset($matchExt)) $matchExt = "*";

if ($matchDir == "") $matchDir = "./";

$matchDir = $sitebase . $matchDir;

chdir ($matchDir);
$handle=opendir("./");

$n = 0;

/* Parse the directory. */
while (false !== ($file = readdir($handle)))
{

   $sz = filesize($file);

   $ext = strtolower(strstr($file, '.'));
   if ($file == "..") $ext = "";

   if ($ext == "" && (($sz % 512) != 0)) $ext = "dummyextension";

   if ($matchExt != "*")
   {
      if ($ext == $matchExt)
      {
        echo "&file$n=$file&fsize$n=$sz";
        $n++;
      }
    }
    else
    {
      echo "&file$n=$file&fsize$n=$sz";
      $n++;
    }

}

echo("&\n");

closedir($handle);

?>
```

Now, we have something interesting called `sitebase`, which is a variable set at the top of the script. This is where our directories and files will *actually* be read from. We don't necessarily want users to have complete directory read access to the root URL of our site, so just before any processing is done, we tack on `sitebase` to the folder, and thereby disguise the actual location that the user is looking at. They think they're looking at:

http://www.mysite.com/images/

but in fact they're looking at:

http://www.mysite.com/SITEBASE/images/

Where `SITEBASE` is actually replaced with the value of the variable, `sitebase`. In this script, I've set it to `mysite/`. This is never passed back to Flash, so the user can't be sure what they're really looking at. Now, admittedly, the security could be made stronger in this script. We could, for example, ensure that the script only accepts and answers requests from a specific file (like our SWF file), but for demonstrational purposes, we're keeping it simple.

What we're doing is simply setting `matchDir` and `matchExt`, and if they're not set, setting them to default values. The default value for `matchDir` is `./` which is the root directory (of `sitebase`). The default value of `matchExt` is `*` , which means all files.

Once these are set, then we modify `matchDir` to include `sitebase`:

```
   $matchDir = $sitebase . $matchDir;
```

Then, we move to that directory, and open a file handle to it:

```
   chdir ($matchDir);
   $handle=opendir("./");
```

Using the file handle we enter a loop, which uses the `readdir` command to loop through all the files in the directory:

```
   while (false !== ($file = readdir($handle)))
```

The first thing we do is to look at the file size, and the extension name of the file we're looping through:

```
      $sz = filesize($file);

      $ext = strtolower(strstr($file, '.'));
```

Strictly speaking this doesn't return the extension of a file but everything after the first full stop, but as long as we are careful with our naming conventions this shouldn't present a problem.

We also check to see if this is a folder (which has no extension) by looking at its size.

```
   (($sz % 512) != 0)
```

If it's a multiple of 512, then it's most likely a folder and not a file. If the file is called `".."`, then this is a folder, but it's the parent folder. We set its extension to `dummyextension` because technically a folder has no extension, but we need an extension to compare.

Next, if the `matchExt` is not `*` , then we check to make sure that the file's extension and the `matchExt` match, and if they do, we generate name/variable pairs returning filename and file size. On the other hand, if `matchExt` was set to `*` , then we unconditionally return the filename and size.

When we're all finished, we return a final `&` and then close the directory handle. That's it. Our directory contents, and the file sizes, have been successfully transmitted to Flash for use with our `FileInfoComponent`, and Flash file browser.