Glob Patterns for File Matching in PHP

30 Apr 2010

This article has been on the cards for a while now with recent articles elsewhere[1,2] prompting me to get this finished and up on the blog. The focus here is on the wildcard patterns that can be used with`glob`.

Intro to the `glob` function

To quote the succinct description on the PHP manual page for `glob`:

> The `glob` function searches for all the pathnames matching *pattern* according to the rules used by the libc `glob` function, which is similar to the rules used by common shells.
>
> – *http://php.net/glob*

What's in a pattern?

Most people who have already encountered `glob` know to make use of the `*` metacharacter to match*some characters*, and those digging a little deeper often discover that discrete alternatives can be globbed with braces (e.g. `image.{gif,jpg,png}`). However, there are more special characters and sequences that can be used to be more (or less, if we want) specific about what to find.

Aside: please **do not** make the mistake of thinking that glob patterns are *regular expressions*, they're just not. If you do want to use regular expressions to find paths/files then you are invited to use SPL's RegexIterator, which allows filtering of an Iterator based on a PCRE regex, in conjunction with a DirectoryIterator or FilesystemIterator (there are recursive flavours of the Regex- and DirectoryIterator if you need to delve into folders). For those SPL-ly inclined, also note the [GlobIterator][globitertor] which combines the goodness of globbing with iteration. If that made entirely *no sense*, please read on! Globs are much less verbose.

So, here are the special doohickeys (technical term!) that we can use with `glob`:

`*` (an asterisk)

Matches zero of more characters.

`?`

Matches exactly any one character.

`[...]`

Matches one character from a group. A group can be a list of characters, e.g. `[afkp]`, or a range of characters, e.g. `[a-g]` which is the same as `[abcdefg]`.

`[!...]`

Matches any single character not in the group. `[!a-zA-Z0-9]` matches any character that is not alphanumeric.

`\`

Escapes the next character. For special characters, this causes them to not be treated as special. For example, `\[` matches a literal `[`.
If *flags* includes *GLOB_NOESCAPE*, this quoting is disabled and `\` is handled as a simple character.

good glob examples

Here are a few examples of what globs might look like alongside a brief description of the intended behaviour: if you have any suggestions please do make them in the comments as I'm running short on inspiration!

pattern	description
`*.txt`	Get directory contents which have the extension of `.txt` (Note: a file could be named simply `.txt`!).
`??`	Get directory contents with names _exactly_ two characters in length.
`??*`	Get directory contents with names _at least_ two characters in length.
`g?*`	Get directory contents with names at least two characters in length and starting with the letter `g`
`*.{jpg,gif,png}`	Get directory contents with an extension of `.jpg`, `.gif` or `.png`. Remember to use the *GLOB_BRACE* flag.
`DN?????.dat`	Get directory contents which start with the letters `DN`, followed by five characters, with an extension of `.dat`.
`DN[0-9][0-9][0-9][0-9][0-9].dat`	Get directory contents which start with the letters `DN`, followed by five _digits_, with an extension of `.dat`.
`[!aeiou]*`	Get directory contents which do not start with a vowel letter.
`[!a-d]*`	Get directory contents which do not start with `a`, `b`, `c` or `d`.
`*\[[0-9]\].*`	Get directory contents whose basename ends with a single digit enclosed in square braces. If *GLOB_NOESCAPE* is used, a single digit enclosed in `\[` and`\]` which would be a pretty weird name.
`subdir/img*/th_?*`	Get directory contents whose name starts with `th_` (with at least one character after that) within directories whose names start with `img` in the`subdir` directory.

Well there we go, I've said what I came here to say so all that remains to be done is give some link love to those two recent articles that prompted me to dust off this draft and click the "publish" button.

With thanks

1. php|architect's Putting glob() to the test focuses on execution times of `glob` and other techniques of getting a list of files.
2. Marcus Schumann's Loop Through Folders with PHP's Glob() article on Nettuts+ gives a brief introduction for those who might not have been introduced to `glob`.