CommonRuby - Feature #8258

Dir#escape_glob

04/12/2013 10:05 AM - steveklabnik (Steve Klabnik)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

This is inspired by https://github.com/rails/rails/issues/6010.

Basically, if you do a Dir.glob in a directory whose name contains a glob character, things break. It would be nice to have a method which would escape the input so that we can Dir.glob inside of those directories.

History

#1 - 04/13/2013 12:01 AM - rkh (Konstantin Haase)

File.fnmatch_escape would make more sense, imo.

#2 - 04/13/2013 01:31 AM - headius (Charles Nutter)

rkh (Konstantin Haase) wrote:

> File.fnmatch_escape would make more sense, imo.

But it would be harder to remember when what you want is "glob" :-)

Why not just {Dir,File}.quote or .escape, to match Regexp.quote/escape? I would vote for File.escape, a method that escapes any file path to make it suitable for globbing.

#3 - 04/13/2013 08:07 AM - steveklabnik (Steve Klabnik)

I don't feel strongly about the name, specifically.

#4 - 04/14/2013 06:34 PM - Eregon (Benoit Daloze)

headius (Charles Nutter) wrote:

> rkh (Konstantin Haase) wrote:
>
> > File.fnmatch_escape would make more sense, imo.
>
> But it would be harder to remember when what you want is "glob" :-)
>
> Why not just {Dir,File}.quote or .escape, to match Regexp.quote/escape? I would vote for File.escape, a method that escapes any file path to make it suitable for globbing.

I agree, this would be strictly superior.

I guess the most common use case is globbing on a directory recursively, so only the base directory is to be escaped, but this is not worth a specific method I think and could be done easily: Dir.glob("#{Dir.escape dir}/**/*.rb") { |file| ... }
Pathname could likely avoid this problem nicely in this situation: dir = Pathname("some_dir"); dir.glob("**/*.rb") { |file| ... }

#5 - 04/14/2013 08:39 PM - Eregon (Benoit Daloze)

What is more worrying is implementations differ quite a bit in treating \ as an escape for these glob characters ({,},[,],*,?).

From my tests:

- MRI handle them fine
- Rubinius does not handle escaped [, { and }.
- JRuby does not handle escaped [and] (For details, see https://travis-ci.org/eregon/path/builds/6326360)

If I am not mistaken, escaping is as simple as: dir.gsub(/\[|\]|*|\?|\{|\}/, '\\\\' + '\0').

#6 - 04/14/2013 10:27 PM - rkh (Konstantin Haase)

- Rubinius does not handle escaped [, { and }.
- JRuby does not handle escaped [and]

These are implementation bugs, imo, and nothing to worry about here.

> If I am not mistaken, escaping is as simple as: dir.gsub(/\[|\]|*|\?|\{|\}/, '\\\\' + '\0').

Yes, but it shifts responsibility for keeping this up to date from the user code to the Ruby implementation, and should be flag dependent. I.e. Ruby 2.0 introduced the EXTGLOB flag.

#7 - 04/15/2013 07:29 AM - Eregon (Benoit Daloze)

rkh (Konstantin Haase) wrote:

> - Rubinius does not handle escaped [, { and }.
> - JRuby does not handle escaped [and]
>
> These are implementation bugs, imo, and nothing to worry about here.

But it means the problem will not be solved in the general case before a while.
It must also have been problematic for some time, so I guess we are not in a hurry either.

> > If I am not mistaken, escaping is as simple as: dir.gsub(/\[|\]|*|\?|\{|\}/, '\\\\' + '\0').
>
> Yes, but it shifts responsibility for keeping this up to date from the user code to the Ruby implementation,

I agree there should be Dir.escape or Dir.escape_glob.

> and should be flag dependent. I.e. Ruby 2.0 introduced the EXTGLOB flag.

Can you give examples? If it works for every case except FNM_NOESCAPE, I think it is better to have a single simple way.

#8 - 04/15/2013 09:23 AM - nobu (Nobuyoshi Nakada)

(13/04/14 18:34), Eregon (Benoit Daloze) wrote:

> I guess the most common use case is globbing on a directory recursively, so only the base directory is to be escaped, but this is not worth a specific method I think and could be done easily: Dir.glob("#{Dir.escape dir}/**/*.rb") { |file| ... }

It reminded me about old proposal, Dir#glob (not Dir.glob).

--
Nobu Nakada

#9 - 04/15/2013 07:38 PM - Eregon (Benoit Daloze)

nobu (Nobuyoshi Nakada) wrote:

> It reminded me about old proposal, Dir#glob (not Dir.glob).

Interesting, do you have a link?

#10 - 06/26/2014 04:54 PM - jacknagel (Jack Nagel)

An official API for escaping paths would be a hugely useful feature. In Homebrew, we use Dir[], Dir.glob and Pathname.glob a lot, but little attention has been paid to properly escaping paths, and over the years we have accumulated a great deal of potentially problematic code.

Benoit Daloze wrote:

> Pathname could likely avoid this problem nicely in this situation: dir = Pathname("some_dir"); dir.glob("**/*.rb") { |file| ... }

We also use Pathname quite heavily in Homebrew and would definitely take advantage of this.

#11 - 04/19/2018 08:13 AM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Feedback

Issue #13056 introduced base: option to Dir.glob method. Is this issue still needed?

#12 - 04/19/2018 11:28 AM - Eregon (Benoit Daloze)

Looks to me like this can be closed since we have Dir.glob(pattern, base: dir) and Pathname#glob uses it.

#13 - 04/19/2018 03:54 PM - mame (Yusuke Endoh)

Eregon (Benoit Daloze) wrote:

> Looks to me like this can be closed since we have Dir.glob(pattern, base: dir) and Pathname#glob uses it.

Consider that we want to enumerate all files that are under a specified directory and whose name is also specified. If the name in question is "foo.txt" for example, we can do it by:

```
basedir = "/path/to/base/dir/"
filename = "foo.txt"
Dir.glob(basedir + "**/" + filename) # or Dir.glob("**/" + filename, base: basedir)?
```

However, if filename is "foo[bar]baz.txt", this code does not work. In this case, this feature is still useful.

(I personally prefer File.fnmatch_escape to Dir.escape_glob.)

#14 - 11/24/2018 04:02 PM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote:

> Eregon (Benoit Daloze) wrote:
>
> > Looks to me like this can be closed since we have Dir.glob(pattern, base: dir) and Pathname#glob uses it.
>
> Consider that we want to enumerate all files that are under a specified directory and whose name is also specified. If the name in question is "foo.txt" for example, we can do it by:
>
> ```
> basedir = "/path/to/base/dir/"
> filename = "foo.txt"
> Dir.glob(basedir + "**/" + filename) # or Dir.glob("**/" + filename, base: basedir)?
> ```
>
> However, if filename is "foo[bar]baz.txt", this code does not work. In this case, this feature is still useful.

Because you'd want to list files whose name is actually "foo[bar]baz.txt"?
I see, makes sense.

My impression is everyone knows "glob'ing" and Dir.glob but very few know the cryptic "fnmatch", so Dir.escape_glob seems easier to find.