Ruby master - Feature #8544

OpenURI should open 'file://' URIs

06/19/2013 11:35 PM - silasdavis (Silas Davis)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

The following code prints the contents of '/tmp/file.txt':

```
require 'open-uri'
open('/tmp/file.txt').read {|f| puts f.read }
```

which although useful should probably fail since a unix file path is not a URI, and therefore might shield data problems in a system

However the following should produce the same output and is a URI, but fails:

```
open('file:///tmp/file.txt').read {|f| puts f.read }
```

I note that the documentation for open-uri does explain that it is a wrapper for http, https, and ftp, but to deserve its name it should open such URIs as specified in this RFC: http://tools.ietf.org/html/rfc1630. This coupled with the fact that it already does open files, but not by a URI specification.

History

#1 - 06/20/2013 12:35 AM - naruse (Yui NARUSE)

Your request sounds reasonable,
but as far as I understand RFC 1630 is informational and practically obsoleted by RFC 1738,
and RFC 1738 is also obsoleted.
Therefore there's no living RFC for file URI scheme.

#2 - 06/20/2013 09:10 AM - phluid61 (Matthew Kerwin)

By rights, RFC 1738 is only superseded by 3986 in terms of specifying a generic syntax; and RFCs 4248 (telnet:) and 4266 (gopher:) show that there is a precedent for perpetuating the scheme definitions from 1738.

In the absence of a living RFC, the file: scheme is a de-facto standard, supported by all major web browsers (for example), and documented all over the internet, so it makes sense for us to support it too.

Also note that RFC 3986 mentions it at least three times:

- Section 1.1 [p6] uses "file:///etc/hosts" as an example
- Section 1.2.3 [p10] mentions the "file" scheme regarding relative references
- Section 3.2.2 [p21] says that '...the "file" URI scheme is defined so that no authority, an empty host, and "localhost" all mean the end-user's machine...'

#3 - 06/21/2013 09:51 AM - phluid61 (Matthew Kerwin)

phluid61 (Matthew Kerwin) wrote:

> In the absence of a living RFC [...]

Perhaps this can get some traction: http://tools.ietf.org/html/draft-kerwin-file-scheme-01

#4 - 06/21/2013 07:04 PM - duerst (Martin Dürst)

phluid61 (Matthew Kerwin) wrote:

> Perhaps this can get some traction: http://tools.ietf.org/html/draft-kerwin-file-scheme-01

It would be great if it did. Just for your reference, others have tried before you.
Please check out http://tools.ietf.org/html/draft-yevstifeyev-ftp-uri-scheme-08, http://tools.ietf.org/html/draft-hoffman-ftp-uri-04, and http://tools.ietf.org/html/draft-hoffman-ftp-uri-04.

The file: scheme is simple in theory, but very complicated in practice!

Also, in your original post, you write:

> require 'open-uri'
> open('/tmp/file.txt').read {|f| puts f.read }
>
> which although useful should probably fail since a unix file path is not a URI, and therefore might shield data problems in a system

Of course this should NOT fail. 'open-uri' does not change open to open URIs and only URIs, it *adds* the capability to open URIs on top of the capability to open files.

#5 - 06/24/2013 01:33 AM - naruse (Yui NARUSE)

- Target version changed from 3.0 to 2.6

> > phluid61 (Matthew Kerwin) wrote:
> > In the absence of a living RFC [...]
>
> Perhaps this can get some traction: http://tools.ietf.org/html/draft-kerwin-file-scheme-01

Great!

duerst (Martin Dürst) wrote:

> phluid61 (Matthew Kerwin) wrote:
>
> > Perhaps this can get some traction: http://tools.ietf.org/html/draft-kerwin-file-scheme-01
>
> It would be great if it did. Just for your reference, others have tried before you.
> Please check out http://tools.ietf.org/html/draft-yevstifeyev-ftp-uri-scheme-08, http://tools.ietf.org/html/draft-hoffman-ftp-uri-04, and http://tools.ietf.org/html/draft-hoffman-ftp-uri-04.

They are ftp: scheme.
For file scheme,
http://tools.ietf.org/html/draft-hoffman-file-uri-03
http://url.spec.whatwg.org/

http://suika.fam.cx/~wakaba/wiki/sw/n/file has additional points but in Japanese

#6 - 06/24/2013 08:54 AM - phluid61 (Matthew Kerwin)

naruse (Yui NARUSE) wrote:

> For file scheme,
> http://tools.ietf.org/html/draft-hoffman-file-uri-03
> http://url.spec.whatwg.org/
>
> http://suika.fam.cx/~wakaba/wiki/sw/n/file has additional points but in Japanese

These are excellent resources, I wasn't sure about (and wasn't able to find) the Hoffman draft when I started. Thank you very much.

#7 - 06/25/2013 05:59 PM - akr (Akira Tanaka)

2013/6/19 silasdavis (Silas Davis) ruby-lang@silasdavis.net:

> Issue #8544 has been reported by silasdavis (Silas Davis).
>
> However the following should produce the same output and is a URI, but fails:
>
> open('file:///tmp/file.txt').read {|f| puts f.read }

open(uri) is impelemented as URI.parse(uri).open.

If URI class has a dedicated class for file URI, such as URI::File,
it is possible to implement URI::File#open.

--
Tanaka Akira

#8 - 06/26/2013 10:24 AM - duerst (Martin Dürst)

naruse (Yui NARUSE) wrote:

> duerst (Martin Dürst) wrote:
>
> > phluid61 (Matthew Kerwin) wrote:
> >
> > > Perhaps this can get some traction: http://tools.ietf.org/html/draft-kerwin-file-scheme-01
> >
> > It would be great if it did. Just for your reference, others have tried before you.
> > Please check out http://tools.ietf.org/html/draft-yevstifeyev-ftp-uri-scheme-08, http://tools.ietf.org/html/draft-hoffman-ftp-uri-04, and http://tools.ietf.org/html/draft-hoffman-ftp-uri-04.
>
> They are ftp: scheme.

Really sorry about that! I knew Paul Hoffman had a draft about file:. I made a mistake when searching, but once I found a draft by Paul, that made it more difficult to realize my mistake.

#9 - 06/27/2013 01:47 PM - naruse (Yui NARUSE)

Experimental implementation is below, it needs error handling and rdoc and tests.

diff --git a/lib/open-uri.rb b/lib/open-uri.rb
index 32f0662..21af81b 100644
--- a/lib/open-uri.rb
+++ b/lib/open-uri.rb
@@ -781,4 +781,20 @@ module URI

```
 include OpenURI::OpenRead
```

end
+

- class File
- def buffer_open(buf, proxy, options) # :nodoc:
- if self.host && self.host.downcase != 'localhost'
- raise ArgumentError, "URI::File#open can't open remote file"
- end
- begin
- raise if options[:mode].nil?
- rescue
- options[:mode] = IO::RDONLY
- end
- buf.instance_variable_set :@io, ::File.open(self.path, options)
- end +
- include OpenURI::OpenRead
- end end diff --git a/lib/uri.rb b/lib/uri.rb index 2e136eb..19bd1d6 100644 --- a/lib/uri.rb +++ b/lib/uri.rb @@ -103,6 +103,7 @@ end

require 'uri/common'
require 'uri/generic'
+require 'uri/file'
require 'uri/ftp'
require 'uri/http'
require 'uri/https'
diff --git a/lib/uri/file.rb b/lib/uri/file.rb
new file mode 100644
index 0000000..b6806fb
--- /dev/null
+++ b/lib/uri/file.rb
@@ -0,0 +1,91 @@
+# = uri/file.rb
+#
+# Author:: Akira Yamada akira@ruby-lang.org
+# License:: You can redistribute it and/or modify it under the same term as Ruby.
+# Revision:: Id
+#
+# See URI for general documentation
+#

+
+require 'uri/generic'
+
+module URI
+

- #
- # File URI syntax is defined by RFCXXXX section X.X.
- #
- class File < Generic
- #
- # An Array of the available components for URI::File
- #
- COMPONENT = [
- :scheme,
- :userinfo, :host,
- :path
-].freeze +
- #
- # == Description
- #
- # Creates a new URI::File object from components, with syntax checking.
- #
- # The components accepted are +userinfo+, +host+, +path+ and
- # +typecode+.
- #
- # The components should be provided either as an Array, or as a Hash
- # with keys formed by preceding the component names with a colon.
- #
- # If an Array is used, the components must be passed in the order
- # [userinfo, host, port, path, typecode]
- #
- # If the path supplied is absolute, it will be escaped in order to
- # make it absolute in the URI. Examples:
- #
- #   require 'uri'
- #
- #   uri = URI::File.build(['user:password', 'ftp.example.com', nil,
- #   '/path/file.> zip', 'i'])
- #   puts uri.to_s -> ftp://user:password@ftp.example.com/%2Fpath/file.zip;type=a
- #
- #   uri2 = URI::File.build({:host => 'ftp.example.com',
- #   :path => 'ruby/src'})
- #   puts uri2.to_s -> ftp://ftp.example.com/ruby/src
- #
- def self.build(args) +
- tmp = Util::make_components_hash(self, args) +
- return super(tmp)
- end +
- #
- # == Description
- #
- # Creates a new URI::File object from generic URL components with no
- # syntax checking.
- #
- # Unlike build(), this method does not escape the path component as
- # required by RFC1738; instead it is treated as per RFC2396.
- #
- # Arguments are +scheme+, +host+, +path+,
- # +query+ and +fragment+, in that order.
- #
- def initialize(*arg)
- super(*arg)
- if @port
- raise InvalidURIError, "a file URI can't have port"
- end
- if @password
- raise InvalidURIError, "a file URI can't have password"
- end
- end +
- # Returns a String representation of the URI::File
- def to_s
- str = super +
- return str
- end

- end
- @@schemes['FILE'] = File +end

#10 - 02/25/2015 11:02 PM - woollyams (Mike Williams)

Here's a naive implementation of support for "file:"

```
require 'open-uri'
require 'uri'

module URI

  class File < Generic
    def open(*args, &block)
      ::File.open(self.path, &block)
    end
  end

  @@schemes['FILE'] = File

end
```

#11 - 12/12/2015 04:30 PM - razum2um (Vlad Bokov)

I think the difference in #relative? #absolute? behaviour also points that absolute URI with "file:///" should be "open"-able

```
> URI.parse('file:///root').absolute?
=> true
> URI.parse('/root').absolute?
=> false
```