URI fragments, and the future

Norman Gray, University of Glasgow, UK

IVOA TCG meeting, Pune

2011 October 16

history

- Started with an RFC comment on the VOEvent spec
- ...but I think it's a bigger issue
- ...and a practical one, rather than a theoretical nit-picking one.

history

- Started with an RFC comment on the VOEvent spec
- ...but I think it's a bigger issue
- ...and a practical one, rather than a theoretical nit-picking one.

I'm not trying to warn of an apocalypse, here, but there's a smart thing to do, here (and by implication...)

Punctu–ation,isn#t
ju‘st!dec$ora/tion

http://www.ietf.org/rfc/rfc3986.txt

http://www.ivoa.net/Documents/REC/Identifiers/Identifiers-20070302.html#defs

ivo://authorityID/resourceKey#local_ID

VOEvent identifiers are defined in VOEvent 2.0, §2.2

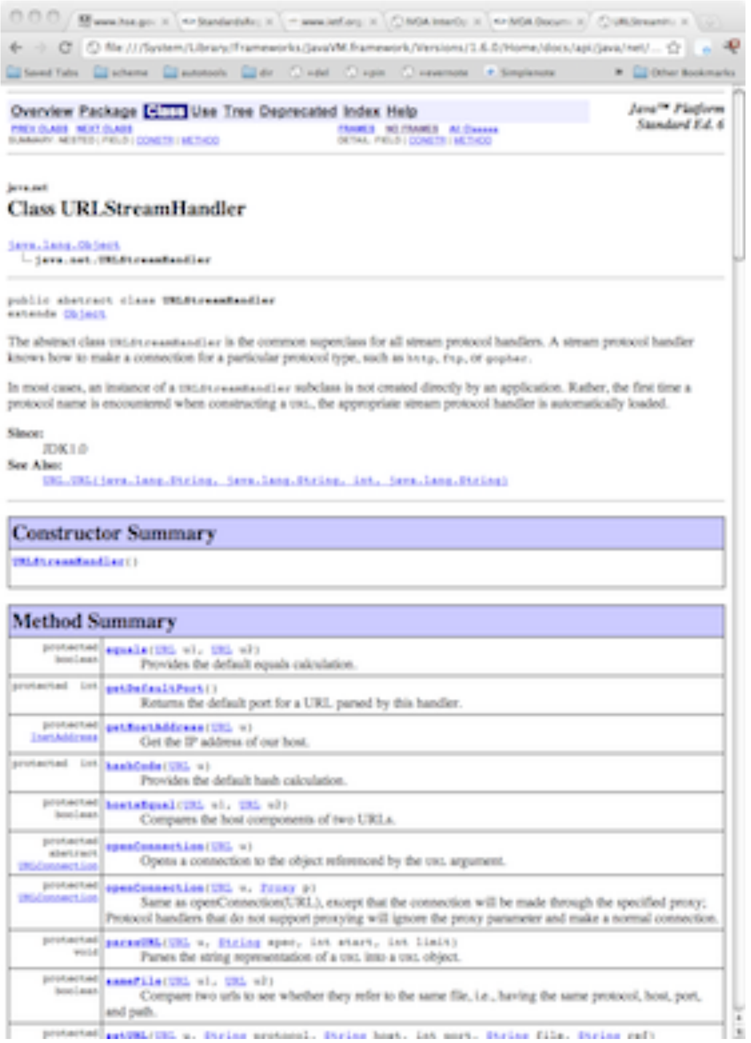
rfc 3986 is unambiguous

RFC 3986, §3.5:

- “The fragment identifier component of a URI allows indirect identification of a secondary resource by reference to a primary resource and additional identifying information.”
- “Fragment identifier semantics are independent of the URI scheme and thus cannot be redefined by scheme specifications.”
- “the fragment identifier is not used in the scheme-specific processing of a URI”

potential problem 1: apis

- A URI-handler API could be constructed in such a way that the implementation couldn't get access to the fragment.
- This *would not be a bug* in the API


www.hse.go
Standards
www.ietf.org
IVOA InterOp
IVOA Docum
URLStream
file:///System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/docs/api/java/net/...
Saved Tabs
scheme
autotools
dir
+del
+pin
+evernote
Simplenote
Other Bookmarks
Overview Package Class Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
Java™ Platform
Standard Ed. 6

java.net

Class URLStreamHandler

```
java.lang.Object
  └ java.net.URLStreamHandler
```

```
public abstract class URLStreamHandler
extends Object
```

The abstract class `URLStreamHandler` is the common superclass for all stream protocol handlers. A stream protocol handler knows how to make a connection for a particular protocol type, such as `http`, `ftp`, or `gopher`.

In most cases, an instance of a `URLStreamHandler` subclass is not created directly by an application. Rather, the first time a protocol name is encountered when constructing a `URL`, the appropriate stream protocol handler is automatically loaded.

Since:
JDK1.0

See Also:
`URL.URL(java.lang.String, java.lang.String, int, java.lang.String)`

Constructor Summary
`URLStreamHandler()`

Method Summary	
`protected boolean`	`equals(URL u1, URL u2)` Provides the default equals calculation.
`protected int`	`getDefaultPort()` Returns the default port for a URL parsed by this handler.
`protected InetAddress`	`getHostAddress(URL u)` Get the IP address of our host.
`protected int`	`hashCode(URL u)` Provides the default hash calculation.
`protected boolean`	`hostsEqual(URL u1, URL u2)` Compares the host components of two URLs.
`protected abstract URLConnection`	`openConnection(URL u)` Opens a connection to the object referenced by the `URL` argument.
`protected URLConnection`	`openConnection(URL u, Proxy p)` Same as openConnection(URL), except that the connection will be made through the specified proxy; Protocol handlers that do not support proxying will ignore the proxy parameter and make a normal connection.
`protected void`	`parseURL(URL u, String spec, int start, int limit)` Parses the string representation of a `URL` into a `URL` object.
`protected boolean`	`sameFile(URL u1, URL u2)` Compare two urls to see whether they refer to the same file, i.e., having the same protocol, host, port, and path.
`protected`	`setURL(URL u, String protocol, String host, int port, String file, String ref)`

potential problem 2: caches

- A proxy or cache must ignore the fragment
- RFC §6.1: “When URIs are compared to select (or avoid) a network action, such as retrieval of a representation, fragment components (if any) should be excluded from the comparison.”
- That is, you ask the proxy/cache for `ivo://auth/obj#frag`, you get `ivo://auth/obj`
- This also *is not a bug* in the cache

potential problem 3, uri++

URIs won’t last forever

potential problem 3, uri++

- URIs won’t last forever
- At some point – a decade? half a century? – there will be a replacement.
- URIs are important: there will be a mapping + gateways + proxies
- …which may not be optional

potential problem 3, uri++

- URIs won't last forever
- At some point – a decade? half a century? – there will be a replacement.
- URIs are important: there will be a mapping + gateways + proxies
- ...which may not be optional

Those gateways cannot be guaranteed to be friendly to URI schemes which depend on behaviour which the URI specification declares must not happen.

non-problem: uris as names

- TAPRegExt uses URIs as *names*: `ivo://ivoa.net/TAPRegExt#upload-http`
- Here, there's no suggestion that the `#upload-http` 'thing' is a differently-retrieved resource
- This goes *with* the grain of the URI definition

suggestions

Future-proof IVOA Recommendations by not going against the grain of the underlying specifications.

suggestions

Future-proof IVOA Recommendations by not going against the grain of the underlying specifications.

Norman Gray, `http://nxg.me.uk`