RFC 3986 : Uniform Resource Identifier (URI): Generic Syntax

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 janvier 2008

Date de publication du RFC : Janvier 2005

https://www.bortzmeyer.org/3986.html

Le Web repose sur trois piliers : le format HTML, le protocole HTTP et, surtout, le concept d'URI, une famille d'identificateurs à la fois compréhensibles par les humains et automatiquement analysables par les programmes. La syntaxe des URI est normalisée dans ce RFC.

Il y a des URI partout aujourd'hui. Sur le côté des autobus, sur les cartes de visite, dans les documentations techniques, dans les articles de la presse papier. Ils sont une des marques les plus spectaculaires du succès du Web. Peu de gens il y a quinze ans osaient dire qu'on verrait des URI à la devanture des boulangeries et des teintureries. Le Minitel n'avait pas d'équivalent (« Tapez 3615 JERAQUE puis choisissez le menu Locations puis allez dans Dernières locations ») et même les informaticiens connectés à Internet utilisaient souvent de telles descriptions (avant les URI, il n'y avait pas de syntaxe standard et analysable pour un nom de fichier accessible en FTP, on disait « ftp.sunet.se, dans /mirrors/macarchive/decoders/foobar.bin ». (Les URI ont été normalisés pour la première fois dans le RFC 1630[1] en 1994.) Même si certains utilisateurs n'ont pas encore compris les URI et donnent des instructions assez miniteliennes à leurs interlocuteurs comme « Tapez Locations en Bretagne dans Google <https://www.bortzmeyer.org/identificateur-vs-moteur-de-recherche.html> » ou bien « Allez sur www.exemple.fr et choisissez Locations puis Bretagne », ils sont largement diffusés, bien au-delà du cercle des quelques informaticiens et physiciens qui les utilisaient en 1990.

Grâce aux URI, on a donc, pour la première fois, des identificateurs standards, à la syntaxe rigoureuse (donc pouvant être traités par un programme) mais décodables et mémorisables par des humains. Ils

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1630.txt

permettent de désigner sans ambiguïté une ressource (contrairement aux moteurs de recherche et aux systèmes de mots-clés, qui sont une référence floue et instable).

Aujourd'hui, qu'est-ce qu'un URI, du point de vue des normes ? URI est le terme qui désigne toute la famille des identificateurs du Web. Les deux membres les plus connus de cette famille sont les URL (RFC 1738) et les URN mais notre RFC préfère ne pas trop utiliser ces termes et ne parler que d'URI (section 1.1.3, voir aussi RFC 3305), entre autres parce que le statut de « localisateur » ("*locator*") n'est pas clair (un URI `http` n'est pas forcément un URL, voir section 1.2.2). La syntaxe qui est décrite dans le RFC est **générique**, elle s'applique à tous les URI, quel que soit leur **plan** ("*scheme*"), la partie de l'URI avant le premier deux-points. Des syntaxes plus précises et plus restrictives peuvent être spécifiées par certains plans (le plus connu des plans est `http`). Notre RFC fournit, lui, ce qui est commun à tous les plans.

Le « cahier des charges » avait été écrit dans les RFC 1736 et RFC 1737. Le premier RFC qui l'avait mis en œuvre était le RFC 2396, que notre RFC 3986 remplace. Les trois lettres du sigle URI indiquent bien la démarche (section 1.1) :

— « "*Uniform*" » car un URI peut décrire une très grande variété de ressources, accessibles (ou pas) via de nombreux protocoles, avec une syntaxe commune,
— « "*Resource*" » car un URI identifie des ressources, sans poser de limites à ce que peut être une ressource (cela peut être un fichier mais aussi bien d'autres choses),
— « "*Identifier*" » car un URI nomme une ressource, il ne dit pas forcément où la trouver (tous les URI ne sont pas des URL). Les URI s'inscrivent donc dans la grande tribu des identificateurs formels.

La section 1.1.1 décrit la syntaxe générique, détaillée en section 3 : un URI commence par un plan, suivi d'un deux-points, d'une **autorité** optionnelle (par exemple le nom de domaine d'une machine), puis d'un **chemin** ("*path*"), dont la syntaxe dépend du plan, d'une requête optionnelle et d'un fragment optionnel. Voici quelques exemples d'URI, inspirés du RFC :

```
http://fr.wikipedia.org/wiki/Jacques_Brel

ldap://[2001:DB8::7:23A]/dc=example,dc=org?objectClass?one

tel:+1-816-555-1212

urn:oasis:names:specification:docbook:dtd:xml:4.1.2
```

Ainsi, `droite://example.org:8042/par/ici?label=ChezMoi#note` se décompose en :
— `droite` : le plan,
— `example.org:8042` : l'autorité,
— `/par/ici` : le chemin,
— `label=ChezMoi` : la requête,
— `note` : l'identificateur d'un fragment.

En Python, on peut facilement analyser un URI avec le module urlparse <`http://docs.python.org/lib/module-urlparse.html`> :

```
>>> from urlparse import urlparse
>>> scheme,authority,path,params,query,fragment = urlparse("http://fr.wikipedia.org/wiki/Jacques_Brel#Discog
>>> scheme
'http'
>>> path
'/wiki/Jacques_Brel'
>>> fragment
'Discographie'
```

La section 1.2 revient sur les choix de conception. Elle part d'un scénario où deux personnes échangent des URI en les notant sur une nappe en papier dans un restaurant, ce qui nécessite des URI parlants, mais écrits dans un alphabet répandu (demande qui peut être contradictoire avec la précédente). D'une manière générale, il est préférable d'utiliser de beaux URI <https://www.bortzmeyer.org/beaux-urls.html>. C'est également dans cette section qu'est posé le principe que les URI n'utilisent qu'un jeu de caractères très limité, un sous-ensemble d'ASCII. Cette limite très contraignante est levée avec les IRI du RFC 3987.

La suite de la section parle du problème important de la **récupération** ("*retrieval*") d'une ressource. Une des utilisations les plus courantes d'un URI est pour accéder à une ressource. Cela nécessite que l'URI soit **résolvable** (ce qui n'est pas, en général, le cas des URN). Mais tous les URI ne sont pas forcément résolvables, ne serait-ce que parce qu'ils peuvent n'être utilisés que comme identificateurs (c'est le cas, par exemple, des URI identifiant un espace de noms XML). Le RFC insiste aussi sur le fait que le plan n'est pas, en général, un protocole réseau. Même quand cela semble être le cas (URI `http`), divers mécanismes font que la ressource peut être récupérée par d'autres moyens que le protocole indiqué (par exemple, pour les documents XML, on utilise typiquement un **catalogue** qui indique une correspondance entre un URI et un fichier local, de sorte que le processeur XML n'aie pas besoin d'aller sur le réseau).

Puis cette section 1.2 discute le caractère hiérarchique des URI, avec ses composants allant du plus significatif, le plan, tout à fait à gauche, au moins significatif, à droite (à noter que les séparateurs des composants varient, ce que n'auraient pas apprécié Rob Pike et Peter J. Weinberger <https://www.bortzmeyer.org/pike-hideous-name.html>).

La section 2 est consacrée aux caractères utilisés pour former un URI, uniquement ASCII, on l'a vu. La section 2.2 donne une liste des caractères qui, quoique présents dans ASCII, ne peuvent pas être utilisés pour un URI car ils ont une signification spéciale. C'est le cas par exemple du # qui introduit l'identificateur d'un **fragment** de la ressource. La section 2.1 normalise le codage « pour cent » où les caractères non-ASCII sont remplacés par le signe % suivi de leur valeur en hexadécimal. (C'est hélas la valeur de chaque octet, pas le point de code Unicode.) Malheureusement, la section 2 précise qu'aucun encodage particulier n'est choisi. Il est donc très difficile de faire des URI utilisant des caractères non-ASCII, même en utilisant le codage « pour cent » puisqu'il n'y a pas de moyen d'indiquer l'encodage utilisé. Ainsi, `http://example.org/caf%E9` est sans doute du Latin-1 mais sans qu'on puisse en être sûr. On limite les risques en supposant que l'encodage est UTF-8 comme dans l'URI `http://fr.wikipedia.org/wiki/caf%C3%A9` (`C3 A9` est l'encodage en UTF-8 du e accent aigu).

L'encodage d'une chaîne de caractères pour en faire un URI peut être effectuée en XSLT par la fonction EXSLT str :encode-uri <http://www.exslt.org/str/functions/encode-uri/index.html>, par exemple ainsi :

```
<xsl:value-of select="str:encode-uri($word, true())"/>
```

ou bien en Python avec la fonction `quote` de la bibliothèque urllib <http://docs.python.org/lib/module-urllib.html> :

```
>>> from urllib import quote
>>> quote("foo # bar ?")
'foo%20%23%20bar%20%3F'
```

Voyons plus en détail les différents composants d'un URI. Le **plan**, d'abord, décrit en section 3.1. Notre RFC ne contient pas de liste de plans. Ils doivent être enregistrés à l'IANA, comme indiqué dans le RFC 7595 (on peut aussi voir le RFC 2718). Les plans les plus connus sont `http`, `mailto` et `urn`. `tag` (RFC 4151) est moins connu mais est un de mes favoris.

L'**autorité** (section 3.2) indique quelle entité va gérer le reste de l'URI. Pour un URN, c'est un simple nom (comme `isbn` pour les ISBN du RFC 8254).Pour le plan `http`, l'autorité est un nom de domaine ou une adresse IP. Dans ce cas, elle peut aussi inclure un nom d'utilisateur, pour l'authentification. Si l'adresse IP est IPv6, il faudra un échappement spécial (la mettre entre crochets), décrit dans le RFC. Enfin, il peut y avoir un numéro de port, bien que cela pose un problème avec les enregistrements SRV (RFC 2782) qui peuvent aussi spécifier un port.

Le **chemin** (section 3.3) indique une ressource particulière pour une autorité. Il est composé de plusieurs parties séparées par une barre oblique (pour des raisons historiques : c'est le séparateur de répertoires sur Unix). Si l'URI est de type `http` et que les pages Web sont des fichiers HTML statiques, le chemin est un répertoire sur le disque dur, mais ce n'est pas forcément le cas pour tous les URI, loin de là !

La **requête** (section 3.4), située après le ? est un composant optionnel de l'URI. Elle est utilisée typiquement pour les ressources générées dynamiquement, en réponse à la question que représente ce composant, mais rien dans la norme n'oblige à les utiliser pour cela.

Le **fragment** (section 3.5), qui se place après le # désigne une partie de la ressource. Si l'URI est résolvable, le fragment est interprété par le client uniquement, le serveur ne le voit pas. L'exemple le plus courant est un fragment composé d'un nom, par exemple, si la ressource est en HTML, un attribut `name` ou `id` d'un élément HTML.

La section 4 explique l'utilisation des URI par un programme. Elle couvre le cas d'URI **relatifs** (la section 5 développe ce point) ou bien, dans la section 4.5, d'URI dont le début, par exemple le plan, manque. De tels URI sont courants dans la presse ou la publicité <`http://thomas-fourdin.net/blog/index.php?post/2008/01/11/Usages-des-adresses-Internet-en-publicite-panorama-Presse-n`
par exemple `www.afnic.fr` au lieu de `http://www.afnic.fr/`. Le RFC note que cette pratique, quoique courante, est dangereuse car elle dépend du contexte pour une bonne résolution. Elle ne devrait pas être utilisé pour des URI à longue durée de vie.

La section 6 s'attaque à un problème difficile, la **normalisation** des URI. On a souvent besoin de comparer deux URI. Par exemple, un navigateur Web affiche les pages déjà visitées avec une couleur différente. Pour cela, il doit comparer les URI présents dans la page Web avec son historique des liens visités. Or, faire cette comparaison octet par octet peut rater beaucoup d'égalités. Par exemple, le plan est insensible à la casse donc `urn:bortzmeyer:exemple:1` est égal à `URN:bortzmeyer:exemple:1`. D'autres canonicalisations dépendent du plan. Par exemple, avec les URI `http`, l'autorité est un nom de domaine et les noms de domaine sont insensibles à la casse donc `http://www.demaziere.fr/eve/` est égal à `http://www.Demaziere.FR/eve/` (la section 6.2.2.1 détaille les comparaisons insensibles à la casse ; sauf indiqué explicitement, les composants d'un URI sont sensibles à la casse). Le navigateur doit donc normaliser les URI avant de les comparer.

Le RFC note qu'aucune normalisation ne sera parfaite. Après tout, deux URI différents peuvent parfaitement indiquer la même ressource, sans que le logiciel ne puisse le deviner.

Une section 7 détaille les questions de sécurité associées aux URI. Elle note que beaucoup de ces questions n'ont pas de solution technique et qu'elles dépendent d'une approche sociale. Par exemple, la persistence des URI, un sujet de recherche très actif <`http://2007.jres.org/planning/paper69c5.`

html?pid=163>, dépend plutôt de bonnes pratiques sociales (notamment d'organisations stables, et qui peuvent voir sur le long terme) que d'une technique particulière. Une autre question de sécurité est traitée dans la section 7.6, c'est celle du hameçonnage, pour les cas où l'auteur du site de hameçonnage tente de fabriquer des URI trompeurs (très peu d'utilisateurs vérifient les URI et la plupart des hameçonneurs ne se donnent même pas la peine d'en fabriquer de vraisemblables). La syntaxe des URI est suffisamment compliquée pour que certains URI soient difficiles à décoder par un humain.

Enfin, notons que l'appendice D expose toutes les différences entre ce RFC et son prédécesseur, le RFC 2396. La principale est la section sur la normalisation, qui a été complètement refaite, mais il y en a beaucoup d'autres, en général de faible importance.