RFC 7320 : URI Design and Ownership

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 juillet 2014

Date de publication du RFC : Juillet 2014

https://www.bortzmeyer.org/7320.html

Ah, les URI... Comme ces identificateurs sont très souvent vus et manipulés par un grand nombre d'utilisateurs, ils suscitent forcément des passions et des discussions sans fin. Ce RFC de bonne pratique s'attaque à un problème fréquent : les applications ou extensions qui imposent des contraintes sur l'URI, sans que cela soit justifié par le format de ces URI. Par exemple, on voit des CMS imposer, au moment de l'installation, que le CMS soit accessible par un URI commençant par le nom du logiciel. S'il s'appelle Foobar, on voit parfois des logiciels qui ne marchent que si l'URI commence par `http://www.example.org/Foobar/`. Pourquoi est-ce une mauvaise idée et que faudrait-il faire à la place ? (Ce RFC a depuis été remplacé par le RFC 8820 [1], moins normatif.)

D'abord, l'argument d'autorité qui tue : la norme des URI, le RFC 3986, dit clairement que la structure d'un URI est déterminé par son **plan** ("*scheme*" en anglais) et que donc l'application n'a pas le droit d'imposer des règles supplémentaires. Les deux seuls « propriétaires » sont la norme décrivant le plan (qui impose une syntaxe, et certains éléments, par exemple le nom du serveur avec le plan `http://`) et l'organisation qui contrôle cet URI particulier (par exemple, toujours pour `http://`, l'organisation qui contrôle le domaine dans l'URI). Imposer des règles, pour une application ou une extension, c'est violer ce principe de propriété. (Il est formalisé dans une recommandation du W3C <http://www.w3.org/TR/2004/REC-webarch-20041215>, section 2.2.2.1.)

Un exemple de structure dans les URI est l'interprétation du chemin ("*path*" en anglais) comme étant un endroit du système de fichiers. Ainsi, bien des serveurs HTTP, en voyant l'URI `http://www.example.org/foo/bar/` chercheront le fichier en `$DOCUMENT_ROOT/foo/bar/toto.html`. C'est une particularité de la mise en œuvre de ce serveur, pas une obligation du plan d'URI `http://`. Un autre exemple est l'utilisation de l'extension du nom de fichier comme moyen de trouver le type de média de la ressource. (« Ça se termine en `.png` ? On envoie le type `image/png`. ») Ces deux cas ne violent pas forcément le principe de propriété des URI car l'utilisateur, après tout, choisit son serveur.

Mais, dans certains cas, l'imposition d'une structure (autre que celle déjà imposée par la norme du plan d'URI) a des conséquences néfastes :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8820.txt`

— Risque de collisions entre deux conventions différentes.
— Risque d'instabilité si on met trop d'informations dans l'URI (voir la section 3.5.1 du document « "*Architecture of the World Wide Web, Volume One*" », cité plus haut). Pour reprendre l'exemple de l'extension du fichier, si on change le format de l'image de PNG en JPEG, l'URI changera, ce qui est néfaste `<http://www.w3.org/Provider/Style/URI.html>`.
— Rigidité accrue. Si un logiciel impose d'être installé en `/Foobar` comme dans l'exemple au début, il contraint les choix pour l'administrateur système (et si je veux des URI avec un chemin vide, genre `http://foobar.example.org/`, je fais quoi ?)
— Risque que le client fasse des suppositions injustifiées. Si une spécification décrit le paramètre `sig` comme étant forcément une signature cryptographique, il y a une possibilité qu'un logiciel client croit, dès qu'il voit un paramètre de ce nom, que c'est une signature.

Donc, pour toutes ces raisons, notre RFC **déconseille fortement** de rajouter des règles de structure dans les URI. Ces règles diminuent la liberté du propriétaire de l'URI.

Qui risque de violer ce principe ? Les auteurs d'applications, comme dans l'exemple Foobar plus haut, mais aussi des gens qui font des extensions aux URI, par le biais de nouvelles spécifications (par exemple pour mettre des signatures ou autres métadonnées dans l'URI). En revanche, ce principe ne s'applique pas au propriétaire lui-même, qui a évidemment le droit de définir ses règles pour la gestion de ses URI (exemple : le webmestre qui crée un schéma de nommage des URI de son site Web). Et cela ne s'applique pas non plus au cas où le propriétaire de l'URI reçoit lui-même une délégation pour gérer ce site (par exemple, un RFC qui crée un registre IANA et spécifie la structure des URI sous `https://www.iana.org/` est dans son droit, l'IANA agissant sur délégation de l'IETF). Le principe « bas les pattes » de ce RFC n'est pas technique : on n'interdit pas de mettre de la structure dans les URI, on dit juste **qui** a le droit de le faire.

Notre RFC reconnait que certaines normes IETF (non citées...) violent ce principe, et appelle à profiter de la prochaine révision pour les corriger.

La partie normative de notre RFC est la section 2. Elle explicite le principe « bas les pattes » (« "*get off my lawn*" » dans le nom original du document en anglais...) D'abord, éviter de contraindre l'usage d'un plan particulier. Par exemple, imposer `http://` peut être trop contraignant, certaines applications ou extensions pourraient marcher avec d'autres plans d'URI comme par exemple `file://` (RFC 8089).

D'autre part, si on veut une structure dans les URI d'un plan particulier, cela doit être spécifié dans le document qui définit le plan (RFC 7230 pour `http://`, RFC 6920 pour `ni:`, etc) pas dans une extension faite par d'autres (« touche pas à mon plan »).

Les URI comprennent un champ « autorité » juste après le plan. Les extensions ou applications ne doivent **pas** imposer de contraintes particulières sur ce champ. Ainsi, pour `http://`, l'autorité est un nom de domaine et notre RFC ne permet pas qu'on lui mette des contraintes (du genre « le premier composant du nom de domaine doit commencer par `foobar-`, comme dans `foobar-www.example.org` »).

Même chose pour le champ « chemin », qui vient après l'autorité. Pas question de lui ajouter des contraintes (comme dans l'exemple du CMS qui imposerait `/Foobar` comme préfixe d'installation). La seule exception est la définition des URI bien connus du RFC 8615. Le RFC 6415 donne un exemple d'URI bien connus, avec une structure imposée.

Autre conséquence du principe « bas les pattes » et qui est, il me semble, plus souvent violée en pratique, le champ « requête » ("*query*"). Optionnel, il se trouve après le point d'interrogation dans l'URI. Notre RFC interdit aux applications d'imposer l'usage des requêtes, car cela empêcherait le déploiement de l'application dans d'autres contextes où, par exemple, on veut utiliser des URI sans requête (je dois

dire que mes propres applications Web violent souvent ce principe). Quant aux extensions, elles ne doivent pas contraindre le format des requêtes. L'exemple cité plus haut, d'une extension hypothétique, qui fonctionnerait par l'ajout d'un paramètre `sig` aux requêtes pour indiquer une signature est donc une mauvaise idée. Une telle extension causerait des collisions (applications ou autres extensions qui voudraient un paramètre de requête nommé `sig`) et des risques de suppositions injustifié (un logiciel qui se dirait « tiens, un paramètre `sig`, je vais vérifier la signature, ah, elle est invalide, cet URI est erroné »). Au passage, les préfixes n'aident pas. Supposons qu'une extension, voulant limiter le risque de collisions, décide que tous les paramètres qu'elle définit commencent par `myapp_` (donc, la signature serait `myapp_sig`). Cela ne supprime pas le risque de collisions puisque le préfixe lui-même ne serait pas enregistré.

Sauf erreur, Dotclear gère bien cela, en permettant, via les « méthodes de lecture » `PATH_INFO` ou `QUERY_STRING` d'avoir les deux types d'URI (sans requête ou bien avec).

Pourtant, HTML lui-même fait cela, dans la norme 4.01 `<http://www.w3.org/TR/1999/REC-html401-19991224` en restreignant la syntaxe lors de la soumission d'un formulaire. Mais c'était une mauvaise idée et les futures normes ne devraient pas l'imiter.

Comme précédemment, les URI bien connus ont, eux, droit à changer la syntaxe ou contraindre les paramètres puisque, d'une certaine façon, l'espace sous `.well-known` est délégué à l'IETF et n'est plus « propriété » de l'autorité.

Dernier champ de l'URI à étudier, l'identificateur de fragment (ce qui est après le croisillon). Les définitions d'un type de média (RFC 6838) ont le droit de spécifier la syntaxe d'un identificateur de fragment spécifique à ce type de média (comme ceux du texte brut, dans le RFC 5147). Les autres extensions doivent s'en abstenir.

Bon, assez de négativité et d'interdiction. Après les « faites pas ci » et les « faites pas ça », la section 3 de notre RFC expose les alternatives, les bonnes pratiques pour remplacer celles qui sont interdites ici. D'abord, si le but est de faire des liens, il faut se rappeler qu'il existe un cadre complet pour cela, décrit dans le RFC 8288. Une application peut utiliser ce cadre pour donner la sémantique qu'elle veut à des liens. Autre technique rigolote et peu connue, les gabarits du RFC 6570, qui permettent de gérer facilement le cas de données spécifiques à l'application dans un URI.

Et, comme cité plusieurs fois, les URI bien connus du RFC 8615 sont un moyen de déclarer sa propre structure sur des URI. Par contre, ils sont censés avoir un usage limité (accéder à des métadonnées avant de récupérer une ressource) et ne sont pas un moyen générique d'échapper aux règles qui nous dérangent !