RFC 7240 : Prefer Header for HTTP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 18 juin 2014

Date de publication du RFC : Juin 2014

https://www.bortzmeyer.org/7240.html

Dans certains cas, la norme du protocole HTTP laisse un serveur choisir entre plusieurs comportements, tous légaux. Un client peut avoir une préférence personnelle et ce nouvel en-tête `Prefer:`, dans la requête HTTP, permet à ce client de la spécifier. Ce n'est pas un ordre, le serveur peut ignorer cette préférence.

Un exemple de cette latitude laissée aux serveurs ? Lorsqu'un client modifie une ressource Web avec la commande `PUT` (cf. RFC 5023 [1]), le serveur peut retourner la nouvelle ressource, complète, ou bien seulement la modification faite. Le client pourrait dire « je ne vais même pas lire ce texte, envoie le minimum » ou bien « je vais tout vérifier soigneusement, envoie la ressource après modification, pour m'éviter un `GET` ultérieur ». Le peut-il aujourd'hui avec l'en-tête `Expect:` (section 14.20 du RFC 2616) ? Non, car `Expect:` est impératif : tout serveur, ou même tout intermédiaire (par exemple un relais) qui ne le comprend pas, ou ne peut lui obéir, doit rejeter la requête. `Expect:` ne convient donc pas pour le cas où on veut dire « si tu arrives à le faire comme cela, ça m'arrangerait mais ne te casse pas trop la tête ». Comme le dit le RFC, dans un style très Jane Austen, « "*preferences cannot be used as expectations*" ». C'est d'autant plus important que certaines préférences peuvent avoir un coût de traitement élevé pour le serveur et que, si celui-ci les acceptait aveuglément, il pourrait se retrouver victime d'un déni de service (cf. section 6).

La section 2 spécifie la syntaxe du nouvel en-tête `Prefer:`. Sa sémantique, on l'a vu, est qu'il n'est pas impératif. Le serveur qui ne comprend pas un `Prefer:` ou ne peut pas ou ne veut pas le respecter continue le traitement de la requête en ignorant cet en-tête. Il peut y avoir plusieurs en-têtes `Prefer:` ou bien un seul avec une liste de préférences séparées par des virgules, comme ici :

```
GET /coffee HTTP/1.1
Prefer: no-sugar, no-milk, hot
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5023.txt`

Certaines préférences prennent une valeur, après un signe = :

```
GET /coffee HTTP/1.1
Prefer: sugar=2
```

Il peut aussi y avoir des paramètres à certaines préférences, après un ; comme :

```
GET /coffee HTTP/1.1
Prefer: milk; organic="true"
```

Vu le caractère optionnel de ce système de préférences, on ne s'étonnera pas que le RFC demande aux serveurs HTTP d'ignorer les préférences inconnues (ce qui est le cas de `milk` ou `sugar`, qui ne sont pas dans le registre officiel <https://www.iana.org/assignments/http-parameters/http-parameters.xhtml#preferences>...)

Que peut-on mettre dans un champ `Prefer:` ? Uniquement des préférences enregistrées <https://www.iana.org/assignments/http-parameters/http-parameters.xhtml#preferences>. La section 4 décrit les préférences existant actuellement :

- `respond-async` indique que le client n'aime pas attendre et préfère une réponse asynchrone plus tard. Le serveur qui acceptera enverra alors un code de retour 202, accompagné d'un champ `Location:` pour indiquer où récupérer la réponse lorsqu'elle sera prête. Cette préférence ne prend pas de valeur.
- Par contre, la préférence `wait` prend une valeur, le nombre de secondes qu'on est prêt à attendre, ce qui est utile en combinaison avec `respond-async`. Ainsi, `Prefer: respond-async, wait=30` signifiera qu'on est prêt à attendre une demi-minute, puis qu'on préférerait une réponse asynchrone ensuite.
- `return` a aussi une valeur, indiquant si le client préfère recevoir la ressource entière ou bien une version allégée. `Prefer: return=representation` exprime le premier choix et `Prefer: return=minimal` le second (c'est sans doute ce que demanderont les clients connectés à un réseau très lent).
- Enfin, `handling` indique où le client met le curseur entre le libéralisme et la rigidité. `handling=lenient` signifie que le client préférerait que le serveur soit libéral, où cas où la requête contienne quelques erreurs mineures, `handling=strict`, que le client demande au serveur de n'accepter que des requêtes parfaites.

Comme le respect de `Prefer:` est facultatif, attention à ne pas l'utiliser comme substitut de la négociation de contenu. En effet, il peut être ignoré par les caches, qui pourraient mémoriser un contenu correspondant à certaines préférences et le servir à d'autres clients qui auraient d'autres préférences. Si le serveur sert des contenus différents selon les préférences, il doit penser à mettre un `Vary:` dans la réponse.

Si le serveur HTTP est sympa, il peut inclure dans la réponse un en-tête `Preference-Applied:` qui indique quelles préférences ont été effectivement appliquées. Un exemple avec la commande `PATCH` du RFC 5789 et le format du RFC 6902 :

```
PATCH /my-document HTTP/1.1
Host: example.org
Content-Type: application/json-patch
Prefer: return=representation

[{"op": "add", "path": "/a", "value": 1}]
```

qui entraîne la réponse :

```
HTTP/1.1 200 OK
Content-Type: application/json
Preference-Applied: return=representation
Content-Location: /my-document

{"a": 1}
```

Les détails bureaucratiques sont en section 5, consacrée aux registres IANA. Les en-têtes `Prefer:` et `Preference-Applied:` sont ajoutés au registre des en-têtes <https://www.iana.org/assignments/message-headers/perm-headers.html>. Un nouveau registre, pour stocker les préférences, est créé <https://www.iana.org/assignments/http-parameters/http-parameters.xhtml#preferences>. Les enregistrements se feront selon la politique « Norme nécessaire » du RFC 5226 : chaque préférence devra être décrite dans un document stable, décrivant ses caractéristiques syntaxiques et sémantiques. Le tout sera envoyé à la liste `ietf-http-wg@w3.org` pour un premier examen, puis approuvé ou rejeté par l'expert nommé par l'IANA.

Début 2013, il semble que cet en-tête `Prefer:` soit déjà géré par Abdera. Le protocole OData cite aussi cet en-tête. Mais on ne s'attend pas à ce qu'il soit massivement utilisé par les navigateurs : il est plutôt prévu pour des clients automatiques, appelant une API REST.