

Relocation ?

Philippe Prados

pp@philippe.prados.name

Avant propos

Ce document explique les ambiguïtés dans la définition d'une URL. Il explique quand et pourquoi utiliser l'en-tête la relocation d'URL et l'en-tête [ETag](#).

Qu'est-ce qu'une URL ? Théoriquement, une URL est la localisation absolue d'une ressource, d'un document ou d'un objet informatique.

Une URL est une chaîne de caractères qui respecte une syntaxe permettant de retrouver un élément informatique. Une URL est un sous-ensemble d'une URI. Une URL permet de retrouver un élément sur un réseau. De localiser une ressource. Une URI peut également identifier un élément par son nom ou d'autres informations (numéro de sécurité sociale, ...). Une URL *localise* une ressource, une URI l'identifie. Par exemple, une URI peut référencer un livre par son code ISBN ([isbn:2-212-08917-1](#)), une URL indiquera une localisation pour le trouver (<http://perso.club-internet.fr/pprados/Livre/QCPP/index.html>).

Une URI est décomposée en deux parties : un protocole suivi de deux-points et une partie spécifique au protocole.

`<protocole>:<spécifique>`

Les protocoles les plus utilisés sont : [http](#), [https](#), [ftp](#), [iiop](#), [ldap](#), [nfs](#), [imap](#), [gopher](#), [mailto](#), [news](#), [telnet](#), etc. En général, les protocoles utilisent un format générique pour la partie spécifique.

`<protocole>://<autorité><chemin>?<requête>`

Pour les protocoles [http](#) et [https](#), le format est le suivant :

`http://[<user:pass>@]<serveur>[:<port>]/<chemin>[#<fragment>]?<requête>`

Les éléments entre crochet sont optionnels.

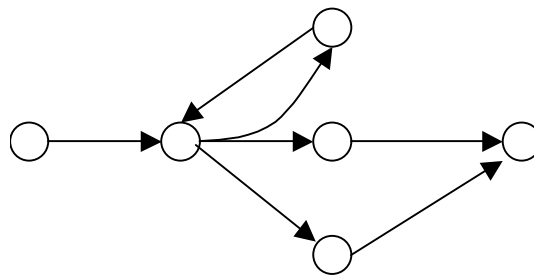
Les navigateurs utilisent une URL pour identifier une page. Le champ [protocole](#) indique le protocole à utiliser. Les champs [user](#) et [pass](#) permettent aux navigateurs d'indiquer l'identifiant de l'utilisateur suivant le protocole. Le [serveur](#) indique le nom ou l'adresse IP de la machine où se trouve la ressource. Le nom est éventuellement converti en adresse IP à l'aide d'un serveur de nom (DNS). Le [port](#) indique le port en écoute pour le protocole. Une valeur par défaut pour chaque protocole permet d'éviter de renseigner ce champ (80 pour le protocole HTTP, 443 pour le protocole HTTPS). [Chemin](#) indique l'emplacement de la ressource. [Fragment](#) indique la section de la ressource récupérée. [Requête](#) indique les informations à transmettre au serveur pour obtenir la ressource.

1. URL-lien ou URL-Document ?

Les navigateurs proposent une ergonomie de type « livre ». Ils supposent qu'il existe une URL par document. Ils proposent à l'internaute de mémoriser un signet sur un document afin de pouvoir le retrouver plus rapidement. Pour les pages statiques, c'est en effet le cas. Chaque URL représente un fichier sur le disque d'une machine du réseau.

Pour les pages calculées, le problème se complique. En effet, dans ce cas, une URL représente un lien entre deux pages. Dans un automate à état, les URL calculées représentent les arcs et non les états de l'automate.

Figure 1 : Automate à états d'un site Internet



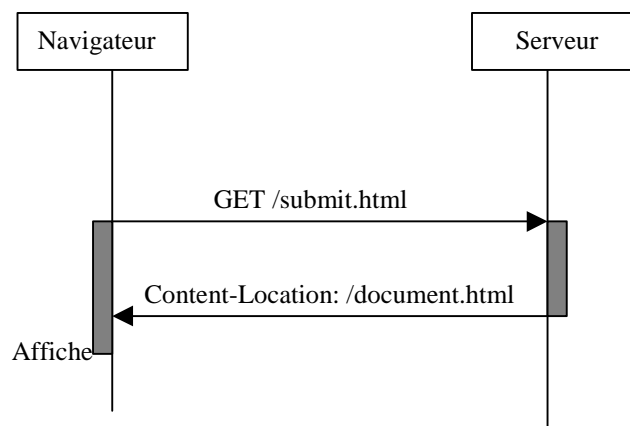
Plusieurs arcs peuvent arriver au même document. De même, une URL peut arriver vers plusieurs pages différentes, suivant les calculs effectués par le traitement. Dans ce cas, l'utilisateur ne peut plus mémoriser de signet, car cela ne correspond plus à un document, mais à un chemin. S'il parcourt à nouveau le signet quelques jours plus tard, il peut se retrouver sur un autre document. Il s'agit alors d'un URL-lien.

Comment concilier les deux approches ? Est-ce qu'une URL doit toujours représenter un document ou un lien ? Comment transformer une URL-lien en URL-document ? Il existe deux techniques pour faire cela.

2. Content-Location

Il existe un en-tête dans le protocole HTTP qui permet de transformer un URL-lien en un URL-document. Pour cela, tous les traitements dépendant d'un état antérieur ou pouvant arriver vers plusieurs pages doivent retourner un [Content-Location](#) pour indiquer la page cible sélectionnée par le traitement.

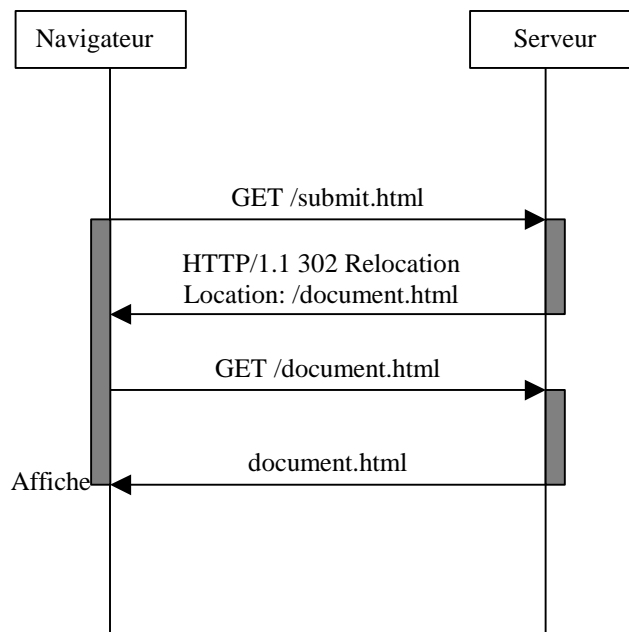
Figure 2 : Content-Location



Cette approche permet à l'utilisateur de mémoriser un signet valide. Il demande la page [submit.html](#), puis mémorise [document.html](#). Malheureusement, les navigateurs ne traitent pas correctement cet en-tête. Ils devraient normalement modifier l'URL référençant le document par la valeur de l'en-tête [Content-Location](#). Ils ne le font pas.

Il faut utiliser une autre approche moins efficace : le code d'erreur 302 (Relocation). Cette approche oblige le client à invoquer deux fois le serveur.

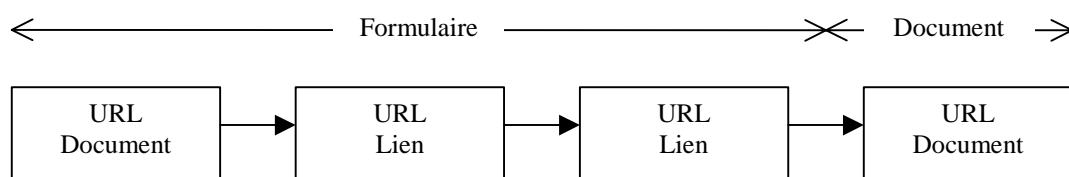
Figure 3 : Relocation



Deux requêtes sont nécessaires pour obtenir la page demandée, pour traverser le lien. Par contre, lorsque l'utilisateur demandera le signet mémorisé, il n'y aura qu'une seule requête, celle pointant vers l'URL-document. Ce n'est pas très efficace en termes de performance. C'est le prix à payer pour garantir une utilisation cohérente de la sémantique des URL pour tous les navigateurs. Les clients peuvent alors mémoriser toutes les pages qu'ils désirent. Les clients ne côtoient que des URL-documents.

Concrètement, cette technique ne doit être utilisée que dans certaines situations. Seuls les documents-cibles, ayant une réelle valeur, étant potentiellement mémorisable par l'utilisateur, doivent faire l'objet de cette technique. Par exemple, si un questionnaire se déroule sur plusieurs pages afin d'arriver vers un document sélectionné, seul ce document doit avoir une URL-document. Les pages du formulaire, sauf la première, n'ont pas besoin d'utiliser une URL-document. Elles peuvent se contenter d'URL-lien.

Figure 4 : Formulaires



Lors de l'utilisation d'un formulaire, pour que les URL-documents soient calculables lors de l'utilisation d'un signet, il est préférable d'utiliser le mode GET. Ce mode permet d'ajouter tous les paramètres du formulaire dans l'URL. L'URL contient les paramètres.

Seuls les navigateurs récents savent mémoriser les champs d'un formulaire utilisant le mode POST avec un signet. À défaut, il faut éventuellement générer un résumé de la requête de l'utilisateur pour l'ajouter à l'URL-Document.

Par exemple, un formulaire permet à l'utilisateur d'entrer son nom, son prénom, et de cocher une case. Ces trois informations seront résumées dans un tableau de byte avec : le nom ; un caractère de fin ; le prénom ; un caract-

tère de fin ; la valeur zéro ou un, suivant la valeur de la case à cocher. Le tout peut éventuellement être compressé puis traduit en ASCII. Le résultat de ce traitement est ajouté en paramètre à l'URL-document.

```
url+"?" + encode( compress( nom+prenom+flag ) )
```

Le navigateur utilisera une URL ressemblant à : `document.html?req=AZGHYYZUIDUUSZSGHYSDO`

Le traitement de l'URL-document consiste à extraire le paramètre, à l'analyser et à recalculer la page.

3. ETag

Une autre approche consiste à ajouter un en-tête **ETag** lors de la réponse de la page. Cet en-tête permet de différencier plusieurs versions de la même page, d'identifier le calcul. Les caches des proxies doivent tenir compte de celui-ci pour identifier les différentes versions des pages. Les données présentes dans cet en-tête sont opaques. Elles seront retournées au serveur lors du rafraîchissement de la page. Les en-têtes **If-Match** et **If-None-Match** permettent de demander la page si elle correspond à une valeur particulière pour le champ **ETag**.

Par exemple, une page est calculée pour afficher le cours d'une action. La page peut être rafraîchie si la valeur de l'action évolue de plus de 3 %. Il n'est pas possible de prévoir cela à l'avance. Lors du calcul de la page, l'en-tête **ETag** possèdera la dernière valeur de l'action. Un champ **Refresh** indiquera qu'il est nécessaire de redemander la page toutes les deux minutes.

```
HTTP/1.1 200 Ok
ETag: "USD=1000"
Refresh: 120
...
```

Lors du rafraîchissement de la page, le tag **If-None-Match** permettra de savoir s'il est nécessaire de recalculer une nouvelle page.

```
GET /Favorite.html HTTP/1.1
If-Match: "USD=1000"
```

Le navigateur envoie automatiquement un en-tête **If-Match** pour une page présente dans son cache.

Si le cours de l'action n'a pas évolué de plus de 3 % depuis la dernière visite, aucune page ni aucun graphique ne sera calculé. Un status 304 sera retourné. Sinon, une nouvelle page est retournée, et une nouvelle valeur pour le tag **ETag** est indiquée.

```
HTTP/1.1 200 Ok
ETag: "USD=1031"
Refresh: 120
...
```


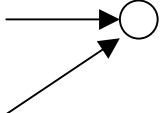
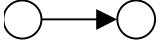
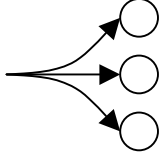
Si plusieurs utilisateurs demandent la même page à une minute d'intervalle, ils n'auront pas la même page de résultat. En effet, l'action aura évolué entre la consultation du premier utilisateur et la consultation du deuxième. Pourtant, l'URL est identique. Comme la valeur de l'en-tête **ETag** est différente pour les deux utilisateurs, les proxies seront capables de différencier les deux versions.

Lors de la mémorisation d'un signet, l'en-tête **ETag** est également sauvegardé. Si l'utilisateur demande à revoir une page favorite, le navigateur va forger une requête en indiquant **If-None-Match** afin d'obtenir éventuellement une nouvelle version. Le traitement sur le serveur pourra : soit considérer que la page présente dans le cache du navigateur est suffisamment fraîche, soit recalculer une nouvelle page et une nouvelle valeur pour l'en-tête **ETag**.

Attention, les navigateurs retournent l'**ETag** dans les requêtes, si et seulement si, le protocole HTTP 1.1 ou supérieur est indiqué dans le status. La version 4.72 de Netscape Communicator ne traite pas de cet attribut.

4. Synthèse

Pour maîtriser correctement les liens et les documents, il y a plusieurs situations à identifier :

- *Une URL référence une page calculée sans dépendance avec l'historique.* Par exemple, une page affiche le cours de la veille de l'action de l'entreprise. Cette situation ne pose pas de problème. Le calcul effectué pour construire la page est indépendant du contexte. Cette situation est strictement similaire à une page statique, non calculée. Eventuellement, un champ **ETag** permet d'identifier différentes versions de la page. Cela évite de recalculer la page si l'action est restée stable. 
- *Plusieurs URL différentes référencent le même document.* Par plusieurs chemins différents, il est possible d'arriver au même document. Pour l'utilisateur, s'il mémorise un signet, il va identifier un chemin particulier et non un document. Dans cette situation, il est préférable d'utiliser l'en-tête **Content-Location** ou le code d'erreur **302** du protocole HTTP. Ainsi, l'URL résultante représente bien le document et non le chemin pour l'atteindre. 
- *Une URL référence une page calculée dépendant du contexte.* C'est généralement le cas des pages calculées à partir des informations d'un formulaire. Cette situation est la plus difficile à régler. Le calcul va utiliser des informations supplémentaires à l'URL (identification de l'utilisateur, champs de formulaires, session de l'utilisateur, informations extérieures, ...) pour calculer la page résultat. Pour transformer ce traitement en URL-document, il faut trouver une astuce pour ajouter à l'URL les informations pertinentes permettant de fixer le résultat du calcul. L'en-tête **Content-Location** ou le code d'erreur **302** devront être utilisés. Une autre approche consiste à indiquer les informations ayant permis de calculer la page dans un en-tête **ETag**. 
- *Une URL référence plusieurs documents différents suivant le résultat d'un calcul.* Dans cette situation, il faut également utiliser les en-têtes **Content-Location** ou le code d'erreur **302** du protocole HTTP. Ou bien, pour différencier les différentes versions du document, il faut ajouter un en-tête **Etag**. L'utilisateur peut alors mémoriser le document et non le traitement. 

Nous avons vu qu'il n'est pas facile d'offrir une utilisation uniforme pour l'utilisateur. Les navigateurs proposent des utilisations qui sont parfois incompatibles avec les désires des serveurs. Ce document explique comment rapprocher les deux points de vue. Il est rare d'avoir une vision uniforme entre le client et le serveur. Les techniques proposées dans ce document améliorent cela, pour le bénéfice du client.

Références :

- RFC 2396 : Uniform Resource Identifiers
- RFC 2616 : HyperText Transfer Protocol