

WebDAV – LA RECHERCHE, LES DROITS D'ACCÈS ET LE

VERSIONING



PREDRAQ.VICEIC@epfl.ch, DOMAINE IT

Dans l'article précédent (FI 6/2008), j'ai eu l'occasion de présenter le protocole WebDAV. J'ai montré à l'aide d'exemples comment utiliser cette extension du protocole **HTTP** pour implémenter un gestionnaire de documents en ligne.

WebDAV, étant basé sur le protocole **HTTP**, possède tous les avantages de ce dernier: il est facile à implémenter à travers de multiples plates-formes, il est transparent quant à son transport à travers le réseau et surtout il est bien respecté en tant que standard par les fabricants de logiciels ou de plates-formes hardware. Ce dernier point est important, car il rend immédiate l'intégration du partage des fichiers à travers WebDAV dans les environnements matériels existants de redondance, distribution de charge ou accélération de cryptage. Pour ces périphériques spécialisés, le protocole WebDAV est vu comme du **HTTP**.

Cet article **numéro deux** parlera des extensions du WebDAV. En effet, la gestion de fichiers *simple*, comprenant le dépôt, la copie, le déplacement, la suppression, le renommage, etc., d'un dossier ou d'un fichier contient un ensemble *sine qua non* de fonctionnalités indispensables à un système de gestion de fichiers. Si toutefois nous voulions

parler de la gestion *documentaire* ou mieux de partage de documents ou carrément de **travail collaboratif**, il y a évidemment besoin de plus que ce que WebDAV peut offrir. Je parlerai donc dans cet article des extensions permettant la recherche des documents, la manipulation des versions (*versioning*) et la gestion des droits d'accès. J'illustrerai, comme dans l'article précédent, toutes ces fonctionnalités par des exemples exécutables en utilisant le logiciel console Telnet.

WEBDAV SEARCH

Webdav SEARCH, le nouveau nom, depuis juillet 2008, pour DASL (*DAV Searching And Locating*) est une des premières (1999) extensions du protocole WebDAV. Cette extension, comme toutes les autres extensions, n'est pas obligatoire. Ce caractère *facultatif*, propre à la constellation des protocoles gravitant autour de WebDAV, permet d'offrir sur un serveur uniquement, par exemple, la partie gestion de fichiers sans implémenter la partie recherche de documents, consi-

SUITE EN PAGE 5

SUITE DE LA PREMIÈRE PAGE

déré typiquement comme plus complexe. Bien entendu la réciproque est valable pour les clients qui eux, à leur tour, peuvent *comprendre* la gestion des fichiers, la recherche, mais peuvent ne pas implémenter la gestion des droits d'accès ou de versions.

OPTIONS

Afin que le client puisse savoir quelles sont les fonctionnalités que le serveur implémente, il utilisera la méthode OPTIONS, dont voici un exemple d'utilisation:

Requête

```
telnet documents.epfl.ch 80
OPTIONS http://documents.epfl.ch/users/p/pv/
pviceic/www HTTP/1.1
```

Réponse

```
HTTP/1.1 200 OK
DAV: 1,2, access-control, ticket, version-
control
[..]
Allow: OPTIONS, PROPFIND, PROPPATCH, LOCK,
UNLOCK, DELETE, GET, HEAD, MOVE, COPY, ACL,
SEARCH
DASL: <DAV:basicsearch>
[..]
```

Ici par exemple on voit que le serveur *documents.epfl.ch* supporte les extensions **access-control**, **ticket** et **version-control**. Vous reconnaîtrez également, grâce à l'article précédent, la liste des méthodes comprises par le serveur. Enfin, constatons que le langage de requêtes de recherche, donné par l'entête **DASL**, est **basicsearch**.

Cet exemple introductif esquisse de la plus belle manière l'extraordinaire souplesse des protocoles de la constellation WebDAV. C'est avec la méthode **OPTIONS**, qui date de HTTP/1.1, que le client et le serveur se mettront d'accord sur le sous-ensemble du langage ainsi que sur des grammaires qu'ils utiliseront pour communiquer. En d'autres termes, c'est à ce moment-là qu'ils établissent l'ontologie de la communication.

SEARCH

Voici donc un exemple de la méthode **SEARCH** à l'aide de la grammaire **basicsearch**:

```
SEARCH / HTTP/1.1
Host: documents.epfl.ch
Connection: Close
Content-Length: 314

<?xml version="1.0"?>
<D:searchrequest xmlns:D="DAV:">
  <D:basicsearch>
  <D:select>
    <D:prop><D:displayname/></D:prop>
  </D:select>
```

```
<D:from>
  <D:scope>
    <D:href>
      http://documents.epfl.ch/
    </D:href>
    <D:depth>infinity</D:depth>
  </D:scope>
</D:from>
<D:where>
  <D:contains>my.epfl</D:contains>
</D:where>
</D:basicsearch>
</D:searchrequest>
```

```
HTTP/1.1 207 Multi-Status
Date: Thu, 28 Aug 2008 13:04:34 GMT
[..]

<?xml version="1.0" encoding="utf-8" ?>
[..]
<D:response>
<D:href>http://documents.epfl.ch/users/p/pv/
pviceic/www/test/test.txt</D:href>
  <D:propstat>
    <D:prop>
      <D:displayname>
        <![CDATA[test.txt]]>
      </D:displayname>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:response>
<D:response>
<D:href>
  http://documents.epfl.ch/users/b/ba/ballini/
  public/Ophtha/Welcome.doc
</D:href>
  <D:displayname>
    <![CDATA[Welcome.doc]]>
  </D:displayname>
  <D:status>HTTP/1.1 200 OK</D:status>
</D:response>
<D:response>
<D:href>
  http://documents.epfl.ch/users/b/ba/ballini/
  www/Download/Welcome2.doc
</D:href>
  <D:displayname>
    <![CDATA[Welcome2.doc]]>
  </D:displayname>
  <D:status>HTTP/1.1 200 OK</D:status>
</D:response>
<D:response>
<D:href>
  http://documents.epfl.ch/groups/m/my/myepfl/
  www/homepage/homepageNM.html
</D:href>
  <D:displayname>
    <![CDATA[myepflNM]]>
  </D:displayname>
  <D:status>HTTP/1.1 200 OK</D:status>
</D:response>
<D:response>
<D:href>
  http://documents.epfl.ch/groups/m/my/myepfl/
  www/homepage/homepage.html
</D:href>
  <D:displayname>
    <![CDATA[myepfl]]>
  </D:displayname>
  <D:status>HTTP/1.1 200 OK</D:status>
</D:response>
<D:response>
<D:href>
  http://documents.epfl.ch/users/d/de/dennaoui/
  public/coursjahia/%2B%20Read%20Me%20First.
  txt
</D:href>
  <D:displayname>
    <![CDATA[Read Me First]]>
  </D:displayname>
  <D:status>HTTP/1.1 200 OK</D:status>
</D:response>
<D:response>
<D:href>
  http://documents.epfl.ch/groups/k/ki/kis-
  unit/public/homepage/maintenance.html
</D:href>
  <D:displayname>
    <![CDATA[maintenance]]>
  </D:displayname>
  <D:status>HTTP/1.1 200 OK</D:status>
</D:response>
</D:multistatus>
```

Ici, nous avons demandé au serveur de nous retourner les noms des documents contenant le mot-clé **my.epfl**. Nous avons spécifié les termes et l'étendue de la recherche à l'aide des mots-clés tels que **select**, **from**, **where**, **orderby** ou **limit**. Ces mots-clés sont couramment utilisés également dans d'autres langages de requêtes.

L'exemple suivant obtient comme réponse du serveur les URIs des documents contenant le mot-clé **my.epfl** et qui sont en format PDF.

```
SEARCH / HTTP/1.1
Host: documents.epfl.ch
Connection: Close
Content-Length: 416

<?xml version="1.0"?>
<D:searchrequest xmlns:D="DAV:">
<D:basicsearch>
  <D:select>
    <D:prop><D:displayname/></D:prop>
  </D:select>
  <D:from>
    <D:scope>
      <D:href>
        http://documents.epfl.ch/
      </D:href>
      <D:depth>infinity</D:depth>
    </D:scope>
  </D:from>
  <D:where>
    <D:and>
      <D:contains>my.epfl</D:contains>
      <D:eq>
        <D:prop>
          <D:getcontenttype/>
        </D:prop>
        <D:literal>
          application/pdf
        </D:literal>
      </D:eq>
    </D:and>
  </D:where>
</D:basicsearch>
</D:searchrequest>
```

```
[..]
<D:response>
<D:href>
  http://documents.epfl.ch/users/s/se/setz/www/
  Seconds%20pas%20my-epfl.pdf
</D:href>
  [..]
<D:href>
  http://documents.epfl.ch/groups/k/ki/kis-
  unit/www/myepfl/Cours_myepfl.pdf
</D:href>
  [..]
<D:href>
  http://documents.epfl.ch/users/p/pa/padespon/
  public/Cours_myepfl.pdf
</D:href>
  [..]
<D:href>
  http://documents.epfl.ch/users/s/se/setz/www/
  Elaine%20McMurray.pdf
</D:href>
  [..]
<D:href>
  http://documents.epfl.ch/groups/a/ag/agepoly_
  representation/www/StrategieInformatiquePour
  LaFormation.pdf
</D:href>
```

```
[..]
<D:href>
  http://documents.epfl.ch/groups/k/ki/kis-
  unit/www/seminaire/usability-week-nielsen-
  -2006.pdf
</D:href>
  [..]
</D:response>
[..]
```

Facile, non? À l'aide des opérateurs tels que **lt** (<), **gt** (>), **and**, **or**, **contains** ou **like** nous pouvons réaliser facilement des requêtes complexes. En ajoutant les métadonnées sur les documents, à l'aide de la méthode **PROPPATCH** comme vu dans l'article précédent, et en faisant la recherche basée sur ces métadonnées, il n'y a pratiquement pas de limites à ce que l'on peut faire.

PETIT BÉMOL

Le plus souvent nous serons intéressés à effectuer les requêtes basées sur les mots-clés se trouvant dans les documents. Pour faire cela, nous utilisons l'opérateur **contains**. Il est important de signaler que cet opérateur est facultatif, c'est-à-dire que le serveur n'est pas obligé de l'implémenter. Ceci est dû principalement au fait que pour répondre à ce type de requêtes, le serveur de fichiers doit avoir un mécanisme d'indexage des documents, ce qui n'est pas toujours le cas. Pour connaître l'état du support de l'opérateur **contains**, le client peut interroger le serveur WebDAV sur ses spécificités grammaticales à l'aide des méthodes du type *Query Schema Discovery* ou QSD. Hélas, le QSD est une fonctionnalité également facultative, ce qui rend très difficile l'implémentation d'un client générique qui supporterait toutes les fonctionnalités DASL. Par ailleurs et pour l'instant, les serveurs WebDAV implémentant de façon plus ou moins complète la norme Webdav SEARCH se comptent sur les doigts d'une main...

Évidemment, ici à l'EPFL, nous avons un tel serveur ☺.

VERSIONING EXTENSIONS TO WEBDAV (DELTA V)

Il existe des outils informatiques qui ne nous sont pas indispensables tant qu'on ne les a pas utilisés une première fois. Dès qu'on y a goûté, on en devient dépendant. Le *versioning* est un tel outil. Qu'on travaille seul, à deux ou à plusieurs, il est merveilleux de pouvoir revoir et reprendre l'historique d'un document, annoter les différentes versions, prendre un chemin différent et revenir ensuite sur la bonne route. Même si les systèmes de *versioning* élaboré (CVS, SVN,...) sont principalement utilisés pour manipuler le code source des programmes informatiques, des versions plus simples de tels systèmes sont particulièrement utiles pour manipuler les *documents de tous les jours*. **DeltaV** est l'extension rajoutant la capacité de *versioning* au protocole WebDAV. Mais avant d'en parler, je m'autorise une petite parenthèse.

Quand on s'y connaît plus ou moins, il est somme toute aisé de concevoir un système qui permet de gérer les documents, les rechercher, manipuler les versions ou les droits d'accès. Une application Web d'importance moyenne à grande peut le faire. Les applications de GED (*Gestion*

Électronique des Documents) le font. Pourquoi alors utiliser plusieurs articles du FI pour parler de WebDAV qui ne fait rien de plus qu'une GED ? Mais tout simplement parce que WebDAV n'est pas une application. WebDAV est un PROTOCOLE. À aucun moment je n'ai parlé de comment on fait les *choses* mais plutôt comment on les demande et sous quelle forme on obtient la réponse. La beauté de WebDAV est qu'il ne fait pas les *opérations*, mais il décrit les processus. De plus, même si les nombreuses extensions rajoutent une grande complexité à WebDAV, n'importe quel développeur peut programmer un client simple qui permettra d'utiliser un petit sous-ensemble des fonctionnalités du serveur. C'est comme un *brunch*, on prend ce qu'on veut, on pourra toujours goûter le reste plus tard.

Rassasiés, revenons au DeltaV.

VERSION-CONTROL

```
VERSION-CONTROL /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Connection: Close
Content-Length: 0
```

```
HTTP/1.1 200 OK
```

Placer sous le régime de *versioning* un document sur un serveur WebDAV supportant l'extension DeltaV (et *documents.epfl.ch* est un tel serveur) est extrêmement aisé à l'aide de la méthode **VERSION-CONTROL**. Ce qui se passe ensuite dépend principalement de la configuration des différentes méthodes de *versioning* que le serveur supporte. Ceux qui ont eu l'occasion d'utiliser les systèmes de gestion de versions tels que CVS ou Subversion connaissent le principe. Tout d'abord, il s'agit d'effectuer un *check out* du document, ce qui se fait à l'aide de la méthode **CHECKOUT**:

```
CHECKOUT /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Connection: close
Content-Length: 0
```

```
HTTP/1.1 204 No Content
Cache-Control: no-cache
[...]
```

À partir de ce moment là, nous pouvons, à l'aide de la méthode **GET**, récupérer le document, le serveur ayant gardé un pointeur sur *notre* version de travail. Ensuite, nous avons tout le loisir de déposer les nouvelles versions du document, à l'aide de la méthode **PUT**, mais aussi de modifier les propriétés (métadonnées) du document, à l'aide de la méthode **PROPPATCH**. S'il y a d'autres personnes qui désirent travailler sur le document, elles effectueront le *check out* à leur tour et travailleront sur une version qui leur est propre.

Quand la nouvelle version du document est terminée et déposée sur le serveur, nous signalons la fin de notre travail à l'aide de la méthode **CHECKIN**:

```
CHECKIN /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Connection: close
Content-Length: 0
```

```
HTTP/1.1 201 Created
Location: http://documents.epfl.ch/_xy-3-892002_1-516993
Cache-Control: no-cache
```

Ici, le serveur nous confirme la création de la nouvelle version du document et nous donne l'URL de celle-ci. Si plusieurs personnes avaient travaillé en même temps sur le document, les différentes versions auraient été conciliées à l'aide de la méthode **MERGE**.

Ceux qui connaissent les systèmes tels que CVS ou SVN comprendront immédiatement que je simplifie intentionnellement le discours. D'autant plus si on sait que DeltaV est également le protocole utilisé pour la communication entre le client SVN et le serveur Subversion, donc il est très complexe. Je note toutefois que Subversion n'est pas le serveur générique DeltaV, il en implémente uniquement le sous-ensemble minimum nécessaire à son fonctionnement.

Le serveur WebDAV *documents.epfl.ch* implémente également un sous-ensemble de la norme DeltaV. Ainsi par exemple, il ne supporte pas la notion de *Workspace*, nécessaire à l'édition *en parallèle* d'un document. Ceci fait que lors de deux *check-in* distincts, il n'y aura pas de conciliation de versions, mais deux versions distinctes seront créées une après l'autre. De même, il n'y aura pas la possibilité de créer les *branches* (**checkout-fork**, **checkin-fork**), action très commune lors d'utilisation du SVN (ou CVS).

Cette implémentation partielle du protocole DeltaV ne pose aucun problème, car, comme pour les autres extensions WebDAV, les concepteurs tiennent compte du caractère facultatif des fonctionnalités. Ainsi, si nous ne voulons pas que d'autres personnes travaillent sur le document en même temps que nous, nous pouvons précéder la méthode **CHECKOUT** par la méthode **LOCK**, comme vu dans l'article précédent à ce sujet. À la fin de notre travail et après avoir effectué le **CHECKIN**, nous devons bien entendu exécuter la méthode **UNLOCK**. Microsoft Word par exemple, utilise cette méthode lorsqu'il accède à un document déposé sur un serveur WebDAV.

Si nous voulons examiner l'historique des versions d'un document, nous le ferons à l'aide de la méthode **REPORT**:

```
REPORT /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Content-Type: text/xml; charset="utf-8"
Content-Length: 165
<?xml version="1.0" encoding="utf-8" ?>
<D:version-tree xmlns:D="DAV:">
  <D:prop>
    <D:version-name/>
    <D:creator-displayname/>
    <D:successor-set/>
  </D:prop>
</D:version-tree>
```

```

HTTP/1.1 207 Multi-Status
Date: Thu, 28 Aug 2008 13:28:40 GMT
Content-Type: text/xml; charset=UTF-8
Content-Length: 1517
Server: Apache-Coyote/1.1

[.]
<D:response>
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516818
</D:href>
  [.]
<D:version-name>
  test.txt (V3)
</D:version-name>
<D:creator-displayname>
  <![CDATA[Predrag Viceic]]>
</D:creator-displayname>
<D:successor-set>
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516820
</D:href>
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516819
</D:href>
</D:successor-set>
[.]
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516819
</D:href>
  [.]
<D:version-name>test.txt (V4)</D:version-name>
<D:creator-displayname>
  <![CDATA[Predrag Viceic]]>
</D:creator-displayname>
<D:successor-set>
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516820
</D:href>
</D:successor-set>
  [.]
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516820
</D:href>
  [.]
<D:version-name>
  test.txt (V5)
</D:version-name>
<D:creator-displayname>
  <![CDATA[Predrag Viceic]]>
</D:creator-displayname>
<D:successor-set></D:successor-set>
  [.]
</D:response>
</D:multistatus>

```

Le serveur nous retourne ainsi la liste des versions avec leur URL, le nom du créateur de la version ainsi que des autres métadonnées telles que la date de la modification de la version ou le pointeur sur la/les versions suivantes du document.

AUTOVERSION

J'avais insisté à de nombreuses reprises sur la structure *en couches* du protocole WebDAV. J'avais ainsi affirmé que le serveur WebDAV n'était vu par les clients HTTP que comme un serveur HTTP. Ceci est plus que jamais vrai lors de l'ajout des fonctionnalités complexes telles que le *versioning*. Ainsi, si un client Web essaie de récupérer un document versionné, il doit exister une sémantique qui lui permettra de le faire sans parler DeltaV. Dans cet exemple précis, c'est facile, le serveur retournera la version la plus récente du document. Mais que

se passe-t-il lors du dépôt des documents versionnés par un client Web(DAV) qui ne parle pas DeltaV? Et bien cela, ça dépendra principalement de la méthode d'*autoversioning* que le serveur utilise:

```

REPORT /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Content-Type: text/xml; charset="utf-8"
Connection: Close
Content-Length: 123

<?xml version="1.0" encoding="utf-8" ?>
<D:version-tree xmlns:D="DAV:">
  <D:prop><D:auto-version/></D:prop>
</D:version-tree>

```

```

[.]
<D:response>
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516818
</D:href>
  [.]
<D:auto-version>
  <D:checkout-unlocked-checkin/>
</D:auto-version>
  [.]
<D:href>
  http://documents.epfl.ch/_xy-3-892002_1-516819
</D:href>
  [.]
<D:auto-version>
  <D:checkout-unlocked-checkin/>
</D:auto-version>
  [.]
</D:response>
[.]

```

Ici, nous voyons que la ressource est configurée avec la méthode d'*autoversioning* **checkout-unlocked-checkin**. Cette méthode va précéder tout dépôt de documents (à l'aide de la méthode **PUT**) par la méthode **CHECKOUT**. Si la ressource est déverrouillée (*unlocked*), le dépôt de document sera suivie par la méthode **CHECKIN**, mettant ainsi fin automatiquement à la transaction. Si au contraire, la ressource est verrouillée, le **CHECKIN** et donc le **PUT** vont échouer forçant ainsi le client à réessayer plus tard.

Le client n'a ainsi qu'à exécuter la méthode **PUT**, le serveur s'occupe de tous les autres *détails*. C'est d'ailleurs par le truchement de l'*autoversioning* que sur *documents.epfl.ch* une nouvelle version est créée à chaque dépôt de document. Ceci nous permet de profiter de la fonctionnalité de *versioning* même si on utilise les clients WebDAV simples tels que MS Web Folders ou Konqueror.

Voici, à titre d'illustration les autres méthodes d'*auto-versioning*:

checkout-checkin: quand on essaie d'effectuer une modification du fichier, à l'aide des méthodes **PUT** ou **PROPPATCH**, le serveur précédera la modification avec un **CHECKOUT** et la suivra avec un **CHECKIN**.

checkout: quand on essaie d'effectuer une modification du fichier, le serveur précédera la modification avec un **CHECKOUT**. C'est au client de faire le **CHECKIN**.

locked-checkout: quand on essaie d'effectuer une modification du fichier verrouillé, le serveur précédera la modification avec un **CHECKOUT**. C'est au client de verrouiller le fichier avant et d'effectuer le **CHECKIN** à la fin.

WEBDAV ACCESS CONTROL PROTOCOL

Il est difficile de considérer le travail collaboratif dans sa totalité sans aborder le sujet de partage de documents. Le partage des documents n'est quant à lui que difficilement mis en oeuvre sans un système de gestion des droits d'accès. Dans sa version la plus simple, un tel système peut proposer aux utilisateurs un ensemble de dossiers aux droits prédéfinis et statiques. L'administrateur du système réglera ainsi ces accès et s'occupera d'adapter les droits quand les utilisateurs le demandent. Dans les organismes avec beaucoup d'interactions transversales entre les utilisateurs, une école polytechnique vient à l'esprit comme exemple, cet administrateur aura beaucoup de travail, ce qui aura comme conséquence l'allongement de délais entre une demande de l'autorisation et sa mise en oeuvre. La décentralisation de la gestion des droits d'accès apparaît rapidement comme la seule solution au goulot d'étranglement ainsi crée. La possibilité donnée aux utilisateurs de gérer eux-mêmes les droits d'accès dans les espaces qui leurs sont réservés devient alors le meilleur moyen de décongestionnement des flux documentaires.

WebDAV Access Control Protocol est l'extension ajoutant la capacité de manipulation des droits d'accès au protocole WebDAV. Parlons peu, mais parlons bien: les droits d'accès sur un dossier ou un document sont représentés par une liste de binômes <Principal, Privileges>. **Principal** est l'entité ayant droit. Un **Principal** peut être un utilisateur, un groupe d'utilisateurs ou un des trois **Principal** prédéfinis:

Propriétaire (*owner*): le propriétaire du document ou du dossier;

Authentifié (*authenticated*): toute personne authentifiée (ayant fourni un nom d'utilisateur et un nom de passe corrects);

Non authentifié (*all*): toute personne accédant au document.

Un privilège peut être par exemple lire, écrire, supprimer, dépendant de l'implémentation côté serveur. Donc en résumant: dans une liste de droits d'accès, nous aurons pour chaque **Principal** une liste des privilèges de ce Principal. Cette liste de **Principal** sera appelée ACL ou *Access Control List* et un binôme <Principal, Privileges> sera appelé ACE ou *Access Control Entry*.

Comment, par exemple, connaître le propriétaire d'un document ?

```
PROPFIND /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Connection: close
Content-Type: text/xml; charset="utf-8"
Content-Length: 109

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop><D:owner/></D:prop>
</D:propfind>
```

```
<D:response>
<D:href>
  http://documents.epfl.ch/users/p/pv/
  pviceic/www/test/test.txt<
/D:href>
```

```
<D:propstat>
  <D:prop>
    <D:owner>
      <D:href>
        http://documents.epfl.ch/xythoswfs/
        principals/ldap/EPFL-default/pviceic
      </D:href>
    </D:owner>
  </D:prop>
  <D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>
```

Dans cet exemple, le serveur nous retourne le propriétaire sous forme d'une URL. Il n'y a rien d'étonnant à cela, très souvent, les systèmes de gestion d'utilisateurs présentent ceux-ci sous forme d'une arborescence (LDAP / Active Directory). Vous remarquerez également que pour interroger le serveur nous utilisons la méthode **PROPFIND**, décrite dans l'article précédent et qui permet de récupérer les *propriétés* ou *métadonnées* d'une ressource (dossier/document). Les droits d'accès pour un client WebDAV simple ne sont que des propriétés du fichier au même titre qu'auteur, nom ou taille. Il ne leur attribuera pas une sémantique particulière. Un client WebDAV plus avancé, pourra lui interpréter cette information et proposer les fonctionnalités plus avancées à l'utilisateur. Même un client de complexité moyenne qui ne ferait que d'interpréter les droits d'accès sans permettre leur modification est déjà d'une aide considérable. Il peut, par exemple, indiquer à l'utilisateur que celui-ci ne pourra pas modifier un fichier et cela, avant qu'il tente de le faire. Un client WebDAV basique au contraire échouera lors de l'écriture de fichier et obligera l'utilisateur d'aller sauver son document ailleurs.

En utilisant la méthode **PROPFIND**, nous pouvons également interroger le serveur afin d'obtenir l'ensemble des privilèges pouvant être attribué à un **Principal**:

```
PROPFIND /users/p/pv/pviceic/www/
test/test.txt HTTP/1.1
Host: documents.epfl.ch
Content-type: text/xml; charset="utf-8"
Content-Length: 126

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:supported-privilege-set/>
  </D:prop>
</D:propfind>
```

```
[..]
<D:supported-privilege-set xmlns:XA="http://
www.xythos.com/namespaces/StorageServer/
acl/">
  <D:supported-privilege>
    [..]
    <D:privilege>
      <D:all/>
    </D:privilege>
    [..]
    <D:privilege>
      <D:write/>
    </D:privilege>
    [..]
    <D:privilege>
      <XA:delete/>
    </D:privilege>
```

```
[..]
<D:privilege>
  <D:write-properties/>
</D:privilege>
[..]
<D:privilege>
  <D:write-content/>
</D:privilege>
[..]
<D:privilege>
  <D:bind/>
</D:privilege>
[..]
<D:privilege>
  <D:read/>
</D:privilege>
[..]
<D:privilege>
  <D:read-current-user-privilege-set/>
</D:privilege>
[..]
<D:privilege>
  <D:unlock/>
</D:privilege>
[..]
<D:privilege>
  <D:read-acl/>
</D:privilege>
[..]
<D:privilege>
  <XA:permissions/>
</D:privilege>
[..]
<D:privilege>
  <D:write-acl/>
</D:privilege>
[..]
```

Si nous voulons obtenir la liste des privilèges que l'utilisateur courant a sur une ressource donnée, nous le ferions à l'aide de mot-clé **current-user-privilege-set**:

```
PROPFIND /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Content-type: text/xml; charset="utf-8"
Connection: close
Content-Length: 129
<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:current-user-privilege-set/>
  </D:prop>
</D:propfind>
```

```
[..]
<D:response>
[..]
<D:current-user-privilege-set [..]>
  <D:privilege>
    <D:read/>
  </D:privilege>
  <D:privilege>
    <XA:write/>
  </D:privilege>
</D:current-user-privilege-set>
[..]
</D:response>
[..]
```

Dans l'exemple précédent, la personne qui interroge le serveur (non authentifié) a les privilèges de lecture (*read*) et d'écriture (*write*) sur le document **test.txt**. Si nous voulons

connaître la liste de tous les privilèges de tous les **Principals** pour une ressource donnée, nous interrogeons le serveur sur la propriété **ACL** (*Access Control List*) d'une ressource:

```
PROPFIND /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Content-type: text/xml; charset="utf-8"
Connection: close
Content-Length: 106
<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:acl/>
  </D:prop>
</D:propfind>
```

```
[..]
<D:response>
[..]
<D:acl [..]>
<D:ace>
  <D:principal>
    <D:property><D:owner/></D:property>
  </D:principal>
  <D:grant>
    <D:privilege><D:read/></D:privilege>
    <D:privilege><XA:permissions/></D:privilege>
  </D:grant>
</D:ace>
<D:ace>
  <D:principal>
    <D:property><D:owner/></D:property>
  </D:principal>
  <D:grant>
    <D:privilege>
      <XA:write/>
    </D:privilege>
    <D:privilege>
      <XA:delete/>
    </D:privilege>
  </D:grant>
</D:ace>
<D:ace>
  <D:principal>
    <D:authenticated/>
  </D:principal>
  <D:grant>
    <D:privilege>
      <D:read/>
    </D:privilege>
    <D:privilege>
      <XA:write/>
    </D:privilege>
    <D:privilege>
      <XA:delete/>
    </D:privilege>
  </D:grant>
</D:ace>
<D:ace>
  <D:principal>
    <D:all/>
  </D:principal>
  <D:grant>
    <D:privilege>
      <D:read/>
    </D:privilege>
    <D:privilege>
      <XA:write/>
    </D:privilege>
  </D:grant>
</D:ace>
```

```

<D:ace>
  <D:principal>
    <D:href>
      http://documents.epfl.ch/xythoswfs/
      principals/ldap/EPFL-default/kermit
    </D:href>
  </D:principal>
  <D:grant>
    <D:privilege>
      <D:all/>
    </D:privilege>
  </D:grant>
</D:ace>
</D:acl>
[..]
</D:response>
[..]

```

Ici, nous voyons que le propriétaire du document a les privilèges de lecture et celui de modification de privilèges. Nous voyons de plus que ces deux privilèges sont non modifiables (*protected*). Le propriétaire a également les privilèges d'écriture et de suppression. En suite, le principal *authentifié* a les privilèges de lecture, écriture et suppression, quant au principal non authentifié (*all*), lui peut uniquement lire et modifier le fichier. Enfin, nous voyons que l'utilisateur **kermit** a tous les privilèges sur ce document.

Si nous voulons modifier les droits d'accès, pour donner par exemple uniquement les droits de lecture et d'écriture à l'utilisateur **kermit**, nous utilisons la méthode ACL:

```

ACL /users/p/pv/pviceic/www/test/
test.txt HTTP/1.1
Host: documents.epfl.ch
Content-type: text/xml; charset="utf-8"
Connection: close
Content-Length: 357
<?xml version="1.0" encoding="utf-8" ?>
<D:acl xmlns:D="DAV:"
xmlns:XA="http://www.xythos.com/namespaces/
StorageServer/acl/">
  <D:ace>
    <D:principal>
      <D:href>
        http://documents.epfl.ch/xythoswfs/
        principals/ldap/EPFL-default/kermit
      </D:href>
    </D:principal>
    <D:grant>
      <D:privilege><D:read/></D:privilege>
      <D:privilege><XA:write/></D:privilege>
    </D:grant>
  </D:ace>
</D:acl>

```

Je vous laisse à présent imaginer toutes les combinaisons possibles d'accès que ce système d'*access control list* permet. Afin de ne pas embrouiller le lecteur plus que de raison, j'omets tout le chapitre de gestion d'héritages des droits d'accès. Pour les personnes intéressées à approfondir ce sujet, les références se trouvent à la fin de l'article.

Nous voilà arrivés à la fin de ce deuxième volet sur le protocole WebDAV. J'espère avoir montré comment ce caractère facultatif et néanmoins extrêmement complet des extensions au protocole de base donne une souplesse extraordinaire à celui qui veut l'utiliser ou l'implémenter. Le but avoué de

ces articles est également de vous donner des idées des applications que vous pourriez développer dans vos laboratoires, sections ou unités et qui utiliseraient le serveur *documents.epfl.ch* comme repository. Avec les bases de php, python ou perl, il devient dès lors trivial de rajouter une gestion de fichiers avancée à vos applications Web, tout en sauvegardant ces fichiers sur une infrastructure robuste et sûre.

Dans le troisième et le dernier article sur le sujet, je parlerai de CalDAV, un protocole de *calendar* basé sur WebDAV.

RÉFÉRENCES

- greenbytes.de/tech/webdav/draft-reschke-webdav-search-17.html
- www.webdav.org/deltav/WWW10/deltav-intro.htm
- webdav.org/specs/rfc3744.html
- FI6/08 – Webdav, du HTTP au partage de fichiers: ditwww.epfl.ch/SIC/SA/SPIP/Publications/spip.php?article1498
- FI4/08 – La gestion électronique de documents: ditwww.epfl.ch/SIC/SA/SPIP/Publications/spip.php?article1443

IL Y A 20 ANS DANS LE FI

Dans le numéro du FI6/88 paru le 12 juillet 1988, Michel Dysli écrivait dans l'article PC ou Terminal:

Aujourd'hui, lors de l'acquisition ou du remplacement d'un terminal d'ordinateur se pose la question du choix entre un nouveau terminal et un ordinateur personnel sur lequel un programme émule les fonctions d'un ou de plusieurs terminaux. Le **terminal** conventionnel peut être un modèle simple et bon marché (environ 2000.- frs avec un bon écran) avec, au minimum, les fonctions ANSI procurées par le déjà ancien mais bien connu VT1000 et le support des caractères à 8 bits. [...]

et si vous avez vraiment besoin d'un terminal graphique

[...]on trouve déjà des modèles très élaborés pour plus de 50'000 frs.

mais si votre choix se porte sur un ordinateur personnel, pas de panique car

[...]**L'ordinateur personnel**, que ce soit un compatible IBM ou un Macintosh, émule les fonctions d'un ou de plusieurs terminaux par un programme géré par le système d'exploitation du PC.

[...]Il existe un troisième choix, qui est encore un peu coûteux, aujourd'hui, mais qui probablement sera la solution standard dans quelques années; c'est une station graphique, soit un ordinateur personnel avec des mots de 32 bits, une mémoire de plus de 4 Mbytes, un disque dur de plus de 40 Mb et un écran graphique de haute résolution.

[...]Le meilleur choix est donc, aujourd'hui, un bon ordinateur personnel (environ 9'000 frs avec disque dur de 20 Mb)[...]