RFC 3402 : Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 2 février 2006

Date de publication du RFC : Octobre 2002

https://www.bortzmeyer.org/3402.html

Le cœur du système DDDS, l'algorithme, est décrit dans ce RFC. C'est un système de réécriture de règles, qui mène à la localisation de la ressource souhaitée.

DDDS, décrit dans le RFC 3401 [1] est un système de correspondance entre une **clé**, un nom de domaine et la localisation d'une ressource, par un URI. Le DNS permet de trouver une adresse ou un URI à partir d'un nom de domaine mais la correspondance est rigide : le nom de domaine est utilisé littéralement, sans variables, sans expressions.

DDDS change cela en permettant l'utilisation de réécritures. Les règles de réécriture ont été popularisées par le fichier de configuration de sendmail. Aujourd'hui, peu de gens écrivent un tel fichier à la main mais des règles de réécriture sont aussi utilisées dans l'excellent module `mod_rewrite` d'Apache. Les règles de réécriture sont en général considérées comme un mécanisme très puissant, relativement simple à mettre en œuvre pour le programmeur mais souvent très difficile à utiliser et à déboguer.

Pour le plaisir, voici une toute petite partie des règles de réécriture d'un fichier de configuration de sendmail :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3401.txt`

```
# strip group: syntax (not inside angle brackets!) and trailing semicolon
R$*                     $: $1 <@>                       mark addresses
R$* < $* > $* <@>       $: $1 < $2 > $3                 unmark <addr>
R@ $* <@>               $: @ $1                         unmark @host:...
R$* [ IPv6 : $+ ] <@>   $: $1 [ IPv6 : $2 ]             unmark IPv6 addr
R$* :: $* <@>           $: $1 :: $2                     unmark node::addr
R:include: $* <@>       $: :include: $1                 unmark :include:...
R$* : $* [ $* ]         $: $1 : $2 [ $3 ] <@>           remark if leading colon
R$* : $* <@>            $: $2                           strip colon if marked
R$* <@>                 $: $1                           unmark
R$* ;                      $1                           strip trailing semi
R$* < $+ :; > $*        $@ $2 :; <@>                    catch <list:;>
R$* < $* ; >               $1 < $2 >                    bogus bracketed semi
```

et les règles du module `mod_rewrite` d'Apache :

```
# Assume that you want to provide www.username.host.domain.com for the
# homepage of username via just DNS A records to the same machine and
# without any virtualhosts on this machine.  Solution:
# For HTTP/1.0 requests there is no solution, but for HTTP/1.1
# requests which contain a Host: HTTP header we can use the following
# ruleset to rewrite http://www.username.host.com/anypath internally to
# /home/username/anypath:
RewriteEngine on
RewriteCond   %{HTTP_HOST}                ^www\.[^.]+\.host\.com$
RewriteRule   ^(.+)                       %{HTTP_HOST}$1          [C]
RewriteRule   ^www\.([^.]+)\.host\.com(.*) /home/$1$2
```

DDDS met donc quelques limites à ces règles, notamment en imposant que les règles ne sont pas appliquées au résultat des règles précédentes (du même groupe) mais toujours à la clé de départ.

La première règle doit être **bien connue**, c'est-à-dire qu'elle est codée en dur dans l'application, elle n'est pas récupérée dans la base de données.

Les règles ont une priorité, des options (par exemple pour indiquer si la règle est terminale ou pas).

La règle elle-même a pour syntaxe les expressions rationnelles et un mécanisme de substitution analogue à celui de sed. Un exemple est décrit dans l'entrée sur le RFC 3403, où la base de données est tout simplement le DNS.