RequestTracker

RequestTracker (RT) est un outil de tickets extrêmement puissant et flexible. Tellement flexible qu'on en vient à se prendre la tête pour faire des trucs de oufs.

Petit tour des trucs que j'ai mis en place sur un RT 4.4.0.

Utiliser le *plus addressing*

De façon simple, pour que RT traite les tickets qui arrivent sur l'adresse email dédiée, on la met dans le `/etc/aliases` de sa machine. Ça fait un truc comme ça :

```
rt:         "|/opt/rt4/bin/rt-mailgate --queue general --action correspond --url https://rt.example.org/"
rt-comment: "|/opt/rt4/bin/rt-mailgate --queue general --action comment --url https://rt.example.org/"
```

Vous noterez que cela met les mails à destination de cette adresse dans la *queue* (ou file) `general`. Or on utilise généralement plus d'une *queue* dans un système de ticket : cela permet de diriger automatiquement vers les personnes les plus à même d'y répondre.

Le problème avec ce système, c'est qu'il faudrait ajouter une adresse dédiée à chaque fois que l'on crée une nouvelle file. Ça peut vite devenir usant.

On va donc utiliser le *plus addressing*. Cette technique, toute bête, consiste à ajouter un discriminant à une adresse mail, précédé généralement d'un + (mais on peut configurer son serveur de mail pour utiliser n'importe quel caractère). `rt@example.org` aura alors pour alias (par exemple) `rt+file_bidule@example.org`.

Pour que RT puisse utiliser cette technique, il faut ajouter `--extension=queue` dans la commande du `/etc/aliases` :

```
rt:         "|/opt/rt4/bin/rt-mailgate --extension=queue --queue general --action correspond --url https://rt.example.org/"
rt-comment: "|/opt/rt4/bin/rt-mailgate --extension=queue --queue general --action comment --url https://rt.example.org/"
```

Voilà ! Il ne vous reste plus qu'à créer vos files via l'interface web. Attention, créez-les avec un nom qui passera dans une adresse mail. Pas d'espace par exemple.

RT récupérera le nom de la file kivabien dans la partie entre le + et le @ et placera automatiquement le ticket dans cette file, tout en gardant la file `general` par défaut.

Les articles

Quoi de plus casse-pieds que de se répéter encore et encore en donnant toujours la même réponse ? Heureusement il y a les articles qui peuvent vous servir de réponses pré-enregistrées :-)

Création des classes et des articles

Allez dans le menu `Administration > Articles > Classes > Ajouter`, créez vos classes (j'en ai créé une par file, n'ayant pas réussi à assigner automatiquement des articles aux files), cochez `Tous les articles de cette classe doivent être disponibles sous forme de liste sur la page de réponse d'un ticket`, décochez `Inclure le résumé de l'article` et `Inclure le nom de l'article` et cochez `Include le champs personnalisé 'Content' > Value` (oh la belle typo de traduction) qui apparaîtra après avoir enregistré la classe (pour ces trois derniers, vous faites comme vous le sentez hein).

Liez la classe à une file via le menu `S'applique à`.

Voilà, vous n'avez plus qu'à créer vos articles dans la classe que vous venez de créer via le menu `Articles > Ajouter`.

Et là, magie, lorsque vous répondez via l'interface web, vous pourrez choisir une réponse pré-enregistrée.

Placement des articles dans la réponse

Je suis un grand fan du *bottom-posting*, mais RT place l'article au-dessus de la citation du message précédent. Remédions à cela.

```
cd /opt/rt4
mkdir -p local/html/Elements/
cp share/html/Elements/MessageBox local/html/Elements/
vi local/html/Elements/MessageBox
```

Cherchez la ligne contenant

```
% $m->comp('/Articles/Elements/IncludeArticle', %ARGS) if $IncludeArticle;
```

et remplacez-la par

```
% if ($IncludeArticle) {
%    my $article = $m->scomp('/Articles/Elements/IncludeArticle', %ARGS);
%    $article    =~ s{\n}{<br />}g;
%    $article    = RT::Interface::Email::ConvertHTMLToText($article);
%    $Default   .= $article unless ($Default =~ s/(.*)(--
.*)/$1$article$2/m);
% }
```

Hop ! votre article se trouve maintenant entre la citation et votre signature :-)

(un redémarrage de RT est peut-être nécessaire pour que cela soit pris en compte)

Ajout des articles pertinents dans le mail de notification d'un nouveau message

Une des forces de RT est de permettre aux intervenants de répondre aux tickets par mail. Le problème est que cela empêche de piocher dans les réponses pré-enregistrées.

Qu'à cela ne tienne, ajoutons-les au mail de notification envoyé aux membres du support.

Allez dans `Administration > Global > Modèles > Choisir`. Il faut modifier le modèle `Notification de modification HTML` (oui, j'ai traduit le nom de mes modèles, mais il est simple à repérer, il est utilisé par les *scrips* 8 et 11).

Ajoutez ceci en bas du modèle :

```
{ my $hotlist = RT::Articles->new( RT->SystemUser );
  $hotlist->LimitHotlistClasses;
  $hotlist->LimitAppliedClasses( Queue => $Ticket->QueueObj );
  my $content   = "-- \n<p><b>Réponses pré-enregistrées pour cette catégorie
de tickets:</b></p>";

  if ($hotlist->Count) {
    while (my $article = $hotlist->Next) {
      $content .= '<p><b>'.$article->Name.'</b><br/>';
      my $class   = $article->ClassObj;
      my $cfs     = $class->ArticleCustomFields;
      my %include = (Name => 1, Summary => 1);
      $include{"CF-Title-".$_->Id} = $include{"CF-Value-".$_->Id} = 1 while
$_ = $cfs->Next;
      $include{$_} = not $class->FirstAttribute("Skip-$_") for keys
%include;

      while (my $cf = $cfs->Next) {
        next unless $include{"CF-Title-".$cf->Id} or $include{"CF-Value-
".$cf->Id};
        my $values = $article->CustomFieldValues($cf->Id);
        if ($values->Count == 1) {
          my $value = $values->First;
          if ($value && $include{"CF-Value-".$cf->Id}) {
            $content .= '<br/>';
            my $c     = $value->Content || $value->LargeContent;
            $c =~ s/\r?\n/<br\/>/g;
            $content .= $c;
          }
        } else {
          my $val = $values->Next;
          if ($val && $include{"CF-Value-".$cf->Id}) {
            $content .= '<br/>';
            my $c     = $value->Content || $value->LargeContent;
            $c =~ s/\r?\n/<br\/>/g;
            $content .= $c;
          }
          while ($val = $values->Next) {
            if ($include{"CF-Value-".$cf->Id}) {
              $content .= '<br/>';
              my $c     = $value->Content || $value->LargeContent;
              $c =~ s/\r?\n/<br\/>/g;
              $content .= $c;
```

```
            }
          }
        }
      }
      $content .= "<br/>----------</p>\n";
    }
  }
  $content;
}
{$content}
```

C'est moche et long, je sais. Dites-vous que j'ai passé plus d'une après-midi pour trouver ça, la documentation est inexistante pour faire ça.

Les intervenants n'auront plus qu'à copier-coller l'article qui se trouve au bas de leur mail de notification dans leur réponse :-)

Commandes par mail

C'est beau de répondre par mail, mais il faut encore se connecter à l'interface web pour effectuer certaines actions. Comme je suis fier d'être fainéant, j'ai créé un *scrip* pour autoriser certaines actions par mail.

Mais avant ça, précisions :

- RT permet aux intervenants de discuter du ticket *sans que cela soit vu par le créateur du ticket* : c'est le but de l'adresse `rt-comment@example.org` du début de l'article. On va utiliser cette adresse pour piloter RT par mail
- un *scrip* est une action effectuée par RT en réponse à un évènement, en utilisant de façon optionnelle un modèle. Typiquement, il y a un *scrip* qui envoie (action) un mail (d'après un modèle) aux membres du support lorsqu'un ticket est créé (évènement).

Créons donc un *scrip*. Menu `Administration > Scrips > Ajouter`.

- Condition (évènement) => Lors d'un commentaire
- Action => définie par l'utilisateur
- Modèle => Modèle vide

Dans le `Programme de préparation d'action personnalisé:` :

```
if ($self->TransactionObj->Attachments->First->Content =~
m/#(JePrends|Fermeture|Spam|Move:.*)/i) {
    return 1;
} else {
    return 0;
}
```

Oui, j'aurais pu faire un *one-liner*, mais il faut que ça reste lisible facilement, et quand on passe des heures à faire des bidouilles comme ça, on apprécie les codes lisibles en un coup d'œil.

Dans le `Code d'action personnalisée (commit):` :

```
if ($self->TransactionObj->Attachments->First->Content =~ m/#JePrends/i) {
    if ( $self->TicketObj->OwnerAsString eq '' ) {
        my $id = $self->TransactionObj->Creator;
        $RT::Logger->info("Setting owner to ".$id);
        $self->TicketObj->SetOwner($id, 'SET');
    }
} elsif ($self->TransactionObj->Attachments->First->Content =~
m/#Fermeture/i) {
    $RT::Logger->info("Closing ticket");
    $self->TicketObj->SetStatus('resolved');
} elsif ($self->TransactionObj->Attachments->First->Content =~ m/#Spam/i) {
    my $ticket = $self->TicketObj;
    my ($status, $msg) = $ticket->SetStatus('rejected');
    $RT::Logger->error("Couldn't delete ticket: $msg") unless $status;

    my $requestors = $ticket->Requestor->UserMembersObj;
    while (my $requestor = $requestors->Next) {
        $requestor->SetDisabled(1);
        $RT::Logger->info("Disabling user ".$requestor->Format." because
he's likely a spammer");
    }
} elsif ($self->TransactionObj->Attachments->First->Content =~
m/#Move:(.*)/i) {
    my $new_queue = $1;
    my $ticket    = $self->TicketObj;

    my ($result, $msg) = $ticket->SetQueue($new_queue);
    if ($result) {
        $RT::Logger->info("Moving ticket to queue ".$new_queue);
    } else {
        $RT::Logger->error("Error while moving ticket to queue
".$new_queue.": ".$msg);
    }
} elsif ($self->TransactionObj->Attachments->First->Content =~
m/#Merge:(.*)/i) {
    my $target = $1;
    my $ticket = $self->TicketObj;

    $ticket->MergeInto($target);
    $RT::Logger->info("Merging ticket into ticket ".$target);
}

return 1;
```

Voilà, enregistrez et c'est bon.

Lorsqu'un commentaire contiendra une commande, elle sera exécutée :

- #JePrends => l'intervenant s'assigne le ticket
- #Fermeture => le ticket est marqué comme résolu
- #Spam => le ticket est supprimé et son auteur ne pourra plus ouvrir de tickets, son adresse

mail sera blacklistée
- #Move:file_de_tickets => le ticket est basculé vers la file de tickets spécifiée
- #Merge:no_ticket => fusionne le ticket avec le ticket dont on donne le n°

Et le spam alors ?

Pour le spam, préparez d'abord un spamassassin pour votre serveur de mails. Ce n'est pas l'objet de cet article, il n'y a qu'à fouiller un peu le web pour trouver des tutos.

On va recréer un *scrip*, mais avant cela on va créer une nouvelle file nommée spam (menu Administration > Files > Ajouter).

Pour notre nouveau *scrip* :

- Condition (évènement) => Lors d'une création
- Action => définie par l'utilisateur
- Modèle => Modèle vide

Dans le Programme de préparation d'action personnalisé: :

```
if ( $self->TicketObj->Subject !~ /\[ .* \]/i ) {
  my $inMessage = $self->TransactionObj->Attachments->First;

  # if no message attachment - assume web UI
  return 0 if (!$inMessage);

  # exit if not email message
  return 0 if (!$inMessage->GetHeader('Received'));

  return ($inMessage->GetHeader('X-Spam-Level') =~ m/\*+/) ? 1 : 0;
} else {
  return 1;
}
```

Dans le Code d'action personnalisée (commit): :

```
my $spamlevel = $self->TransactionObj->Attachments->First->GetHeader('X-
Spam-Level');
if ($spamlevel =~ m/\*\*\*+/) {
  if ($spamlevel =~ m/\*\*\*\*\*/) {
    $RT::Logger->info("This mail seems to be a spam => deleting");
    $self->TicketObj->Delete();
  } else {
    $RT::Logger->info("This mail seems to be a spam => queue spam");
    $self->TicketObj->SetQueue('spam');
  }
}
return 1;
```

Avec cela, les mails ayant un score de 5 ou plus au *spamLevel* seront supprimés, et ceux qui ont entre 3 et 5 vont au purgatoire, dans la file spam.

Prenez soin de déplacer ce *scrip* tout en haut de la liste pour qu'il soit le premier exécuté.

Plugins

En vrac, les plugins que j'utilise :

- RT::Authen::ExternalAuth::LDAP
- RT::Extension::LDAPImport
- RT::Extension::ReportSpam

Les deux premiers sont maintenant intégrés à RT, il n'y a pas besoin de les installer, juste de les configurer. Ils servent respectivement à assurer l'authentification LDAP à l'interface web, et à importer en masse les comptes du LDAP pour permettre à l'administrateur de mettre les collaborateurs dans les bons groupes sans attendre qu'ils se soient logués une première fois.

Le dernier plugin ajoute un **S** dans le menu des tickets, permettant de les déclarer comme spam d'un simple clic.

Conclusion

On peut faire de merveilleuses choses avec RT, pour peu que l'on ait le temps de fouiller dans la documentation (officielle ou non)... et dans le code !

Une fois bien configuré, il devrait permettre d'alléger la charge de travail du groupe de support et je peux vous dire que pour en faire depuis plus de deux ans pour Framasoft et bien plus pour mon ancien boulot, ce n'est pas quelque chose à négliger :-)

NB : bien évidemment, ce superbe logiciel est en Perl :D