Manuel ChatScript pour plusieurs Bots

© Bruce Wilcox, mailto:gowilcox@gmail.com www.brilligunderstanding.com
Revision 6/25/2017 cs7.51
Traduction Mathieu Rigard 02/12/2017 pour Utopia : m.rigard@utopia-french.tech https://utopia-french.tech

A faire : corrections (en particuliers zones surlignées), ancrer les liens.

Le système peut faire cohabiter plusieurs bots sur le même moteur. On peut restreindre des Topics pour qu'ils ne soient accessibles qu'à certains bots. On peut également restreindre l'accès aux Faits et aux Fonctions, lorsque leurs variantes existe pour chaque bot.

Restreindre par les Règles

Le plus simple est de restreindre les Règles afin qu'elles ne soient accessibles qu'à certains bots en utilisant quelque chose comme :

```
?: ($bot=harry ...)

?: (!$bot=harry ...).

t: ($bot=harry) My name is harry.
```

On peut aussi préciser à quel bot vous allez accéder lors de votre login en ajoutant `:nomdubot` à votre nom de login, ex : `bruce:harry`. Le système de démo ne possède qu'un seul bot, Harry. S'il en avait deux, Harry et Georgia, on pourrait obtenir Georgia en se loggant comme ceci `votrenom:georgia`. Et on pourrait alors savoir à qui nous avons affaire en demandant `Quel est ton nom`.

Lorsqu'on ne spécifie pas le nom d'un bot nous avons le bot par défaut. Comme le système fait-il pour savoir duquel il s'agit ? Il cherche un Fait dans la base de donnée dont le sujet est le nom du bot par défaut et dont le verbe est defaultbot. S'il n'en trouve aucun, le bot par défaut est appelé anonymous, et il y a de grande chances que rien ne fonctionne du tout. Définir le bot par défaut est ce que fait la table lorsqu'on compile `simplecontrol.top`. Il contient :

```
table: defaultbot (^name)
^createfact(^name defaultbot defaultbot)
DATA:
harry
```

Typiquement, lorsque vous lancez un build level 1 sur vos Topic (ex : `:build ben` ou `:build 1` ou autre), vous appliquez une surcouche invoquant une fonction d'initialisation (`bot definition script`) pour votre ou vos bots sinon votre bot ne pourra pas fonctionner.

Cette fonction est appelée dès qu'un nouvel utilisateur tente de parler au bot, ce qu'il dit implique un Topic d'où commencer et quel code appliquer à l'entrée de l'utilisateur. Nous avons besoin de

l’une de ces fonctions pour chaque bot, même si cette fonction n’est qu’une simple copie (ou différente d’ailleurs). Dans le cas de Harry, la fonction est

```
^outputmacro: harry()
    ^addTopic(~introductions)
    $cs_control_pre = ~control
    $cs_control_main = ~control
    $cs_control_post = ~control
```

On peut changer le bot par défaut soit en invoquant différents fichiers de build nommant le bot par défaut, soit en éditant notre table.

PLUSIEURS BOTS

Pour rendre disponibles plusieurs bots nous pouvons commencer par créer plusieurs dossiers en copiant celui de Harry et en en faisant des bots spécifiques avec chacun leur `filesxxx.txt`. Cela permet de les construire chacun en temps que bots « stand-alone », c’est à dire existants séparément, mais si l’on veux les faire cohabiter on doit n’avoir qu’un seul fichier `filesxxx.txt` dans lequel seront nommés les dossiers distincts de chaque bot- ils doivent être builder ensembles.

Si vous ne faite seulement que créer plusieurs `bot definition scripts`, alors les bots partagerons également tous les définitions d’outputmacro, tous les Topics, et tous les Faits créés au ment du build. De toutes façons, les Faits créés durant la conversation avec un utilisateur seront tous uniques. Le résultat d’une conversation entre un bot spécifique et un utilisateur aboutit dans un fichier Topic spécifique, ainsi, ce qu’un bot apprend, l’autre ne l’apprend pas.

Si vous essayez de créer deux bots ayant chacun leur dossier séparé, et que vous clonez des copies depuis un dossier originel des fichiers .top `~control` et `~introductions` et de `~childhood` vous aurez un message d’erreur au moment de la compilation. Vous ne pouvez en fait qu’avoir une copie de `~control` que vos bots utilisent ensemble. De même, ils devront avoir des noms de Topic différents pour `~introductions` et `~childhood` OU ALORS vous pouvez mettre des restrictions pour les bots à quelques endroits. Voir plus loin.

On peut invoquer le nom de plusieurs bots en séparant leurs noms par des virgules non suivies d’espace. Ex : `Topic: ~myTopic bot=harry,roman [monMotClé]`

Restriction par Topic

Un Topic peut être restreint à un bot spécifique. Cela se fait lors de la définition du Topic en nommant un ou plusieurs bots (noms séparés par des virgules non suivies d’espace) autorisés à utiliser ce Topic.

```
Topic: ~monTopic bot=Harry,Georgia REPEAT (motClé)
```

Maintenant que nous avons découvert la possibilité de créer plusieurs copies d’un même Topic, nous allons voir que différent bots peuvent accéder à différentes copies. C’est ainsi que l’on fait une famille de Topics portant le même nom. Les règles a appliquer sont :

- Les Topics partagent les mêmes mots-clés

- `:verify` peut uniquement vérifier la première copie d'un sujet auquel il a accès

On peut créer plusieurs copies d'un Topic ayant le même nom, qui varieront suivant les restrictions de bot qui y seront attachées et suivant leur contenu. L'ensemble des mots-clés attribués à un Topic Topic est alors la somme des mots-clés présent dans l'ensemble des copies possédant le même nom de Topic. Vous ne devez pas attribuer à un Topic un nom se terminant par un point ou un nombre (cela est utilisé en interne par le système pour représenter plusieurs Topic ayant le même nom).

Lorsque le système tente d'accéder à un Topic, il vérifie les restrictions liés aux bots. Si un Topic échoue, il essaiera la version dupliquée suivante et ainsi de suite jusqu'à ce qu'il trouve une copie que le bot est autorisé à utiliser.

Cela signifie que si le Topic 1 est restreint à Ben et que le Topic 2 n'a aucune restriction, Ben va utiliser Topic 1 et tous les autres bots le Topic2. Si l'ordre des déclarations était interverti, alors tous les bots utiliseront le Topic 2 y compris Ben (puisque maintenant il précède le Topic 1 dans la déclaration).

On peut également utiliser les commandes `:disable` et `:enable` pour éteindre un ou des Topic à l'attention d'une personnalité.

Heureusement vous n'êtes pas obligé d'utiliser une restriction pour un bot sur chacun des Topics. On peut les mettre dans un fichier et tous les Topics qui seront compilés ainsi hériteront de la restriction (mais cela sera supplanté par une valeurs spécifiques qui pourraient toujours être insérée dans un Topic).

```
Topic: ~Topic1 ()
...
bot: Ben
Topic: ~Topic2 ()   # This Topic is restricted to Ben
...
```

Heureusement vous n'êtes pas obligé de faire ça pour tous les fichiers. Vous pouvez aussi le mettre dans le fichier de build filesxxx.txt. Mais vous aurez besoin de compiler chaque « bot definitions » ne possédant pas de restriction de bot applicable. Ex :

```
RAWDATA/BOTDEFINTIONS
bot: Harry
RAWDATA/HARRY/
bot: Georgia
RAWDATA/HARRY/
bot: 0
RAWDATA/QUIBBLE/
```

Note : Comment avons-nous pu réutiliser tous les Topics de Harry sans les changer ? C'était inutile. Mais vous auriez pu cloner le dossier HARRY pour faire Georgia et ensuite faire des modifications puis compiler le tout. Les Topics dupliqués peuvent coexister. Le `bot: 0` efface les restrictions de bots.

Si vous avez des restrictions de bot dans un fichier, ils l'emportent temporairement sur le fichier de construction un (:build 1), qui reprend son effet lorsque le fichier ou le Topic se termine.

Les Faits partagés (Shared Facts)

`Share` – Normalement, si vous avez plusieurs bots, ils dialoguent tous indépendamment avec l'utilisateur. Ce qu'un bot apprend ou dit n'est pas accessible par les autres bots. Il est possible de créer un groupe de bots qui entendent tous ce qui a été dit et partagent les informations. Partager pour un Topic signifie que tous les bots peuvent avoir un accès normal en lecture et modification à l'état d'un Topic. Ainsi si l'un des bots utilise un gambit, un autre ne pourra pas le réutiliser après pour lui-même.

```
Topic: ~monTopic SHARE REPEAT ()
```

Tous les Faits créés seront visible pour les autres bots. Et si vous créez une variable utilisateur permanente dont le nom commence par `$share_`, alors tous les bots pourront la voir et la modifier. Alors `$share_mavar` deviendra une variable commune. Lorsque le partage est effectif, l'état attaché à l'utilisateur (ce qu'il dit, ce que le bot dit, à quelle volley il en est, où sont les marqueurs de répliques) est partagé communément à l'ensemble des bots- ils progressent dans une conversation conjointe.

Les Bots entièrement indépendants - Restriction sur les Faits et les Fonctions

Normalement, tous les Faits générés lors de la compilation sont disponibles pour tous les Bots. Mais vous pouvez créer des Faits limités à des Bots spécifiques. Vous pouvez avoir jusqu'à 64 Bots différents contrôlant différents Faits. Cela s'effectue en réglant $cs_botid à une certaine valeur avant de créer des Faits. Et vous mettez `$cs_botid = un_chiffre dans votre `bot definition script.` Les chiffres déclarés forment une sorte de masque qui autorise la propriété et l'accès aux Faits. Par exemple dans le script de définition de bot pour Harry, vous pourriez faire `$cs_botid = 1` et pour Georgia : `$cs_botid = 2`. Chaque valeur représente un bit et devrait donc être une puissance de 2.

Lorsque vous voulez créer des Faits dans des tables, quelque part au début des Faits d'un bot à créer, vous pouvez mettre le `$cs_botid` depuis la table. Le plus pratique est d'utiliser la commande `bot:` pour le faire dans un fichier ou pour le build.

```
bot: 1 Harry
```

Lorsque vous utilisez la commande bot: dans une ligne de votre fichier de build `filesxxx.txt`, il s'applique toujours jusqu'à ce qu'une autre commande bot: soit passée. Si c'est une commande passée dans un fichier local, alors cela affecte ledit fichier mais ensuite la compilation ne réverbère pas jusqu'à ce qu'une nouvelle valeur bot: soit trouvée (soit globalement ou dans un fichier).

Ceci définit à la fois le botid et le nom du bot, contrôle les restrictions que les Faits obtiennent (botid) et les restrictions que les Topics obtiennent (nom du bot). Le botid ne fait pas que contrôler des Faits. Il contrôle également la propriété des définitions outputmacro. Vous pouvez définir différentes copies de fonctions avec le même nom, différents arguments, différents codes, en faisant en sorte que le botid soit différent.

Vous pouvez passer à un propriétaire de bot sans nommer le(s) bot(s), auquel cas les Topic créés seront utilisables par n'importe quel bot mais les Faits et les Fonctions seront limités au propriétaire du bot.

```
bot: 2
```

Vous désactivez les règles de propriété avec

```
bot: 0
```

Dans le fichier de build, vous devez compiler toutes les macros de définition de bot et la macro de bot par défaut sous la propriété bot: 0 . Tous les autres Bots peuvent être construits indépendamment en utilisant des identifiants de bot qui sont des puissances de deux et des noms de bot uniques-. ``` bot: 0 private/Mine/CONTROL/botmacro.top private/Mine1/CONTROL/botmacro.top private/Mine2/CONTROL/botmacro.top bot: 1 John private/John/ bot: 2 Harry private/Harry/ bot: 4 Martha private/Martha/