Spécification de syntaxe

Expressions évaluables MoodleScript

Les expressions évaluables MoodleScript, définies à partir de la version 2022081700 du moteur MoodleScript, permettent d'écrire des tests sur des données de Moodle, indépendamment, ou au contraire en relation avec la situation courante de l'utilisateur. Une syntaxe générique a été définie pour permettre d'exprimer de nombreux cas de figure portant sur des données différentes. Cette syntaxe définit :

- Un jeu d'opérateur logiques pris en charge pour la combinaison “d'éléments évaluables”
- La forme générale d'un élément évaluable
- Le format des valeurs “littérales”

Les opérateurs logiques

Ils sont au nombre de 4 et correspondent à des combinatoires logiques classiques :

- NOT (opérateur unaire, négation)
- AND (binaire, ET logique)
- OR (binaire, OU logique)
- XOR (binaire, OU exclusif)

Le traitement de la combinatoire logique ne supporte pas actuellement l'imbrication (nesting), et donc le marquage par parenthèses. Les règles de précédence des opérateurs est imposé : NOT > AND > OR ou XOR.

forme générale d'un élément évaluable

Un élément évaluable est une proposition à au plus deux opérandes :

- [Opérande 1] <operateur> [Opérande 2]
- [Opérande 1] <operateur> (cette forme est donc admise pour un opérateur à une opérande).

Deux types d'opérateurs sont définis : un type portant sur les “données”, comme par exemple les opérateurs d'évaluation '=' ou '<'. Les opérandes sont alors des “valeurs” finales d'attributs, comparables arithmétiquement ou textuellement.

Exemple :

- `user:username:“harry”:department = “Physics”` (“harry” est il lié au département de Physique)

Le deuxième type porte sur des instances d'objets, comme par exemple l'opérateur 'isloggedin'. Dans ce cas, les opérandes sont des références à des instances d'objets Moodle et non des valeurs d'attributs.

Exemple :

- `user:username:“harry” isenrolled course:current` (“harry” a-t-il une inscription active dans le cours courant)

La liste complète des opérateurs “fonction” supportés est donnée plus loin.

D'après les définitions différentes ci-avant, on voit que les opérandes peuvent être de trois types :

- Une valeur scalaire littérale (numérique ou textuelle)
- Une référence à un objet
- Une référence à un attribut scalaire d'un objet (ce cas est une extension du précédent)

Valeurs scalaires

Doivent être notées entre guillemet si elles sont “seules”, c'est à dire que l'opérande est à lui tout seul une valeur scalaire.

Référence à un objet

Elle se décompose en deux ou trois segments qui doivent définir une instance unique d'un certain type d'objet métier de Moodle. On utilise un “type d'objet” en premier segment, les segments suivants doivent nommer un attribut identifiant et fixer une valeur de l'identifiant.

Formes générales :

```
  [optype]:[attribut identifiant]:[valeur identifiante]
```

ou

```
  [optype]:[deictic]
```

Exemples :

- `user:id:364`
- `user:idnumber:FC12654` (ou `user:idnumber:“FC12654”` aussi admis)
- `course:shortname:“TRANSFORM 2020-2021”`
- `category:idnumber:“MAGICS”`

Cas particulier, utilisation de déictiques :

Les déictiques sont des expressions dont la valeur change en fonction du contexte ou du moment de l'invocation. Nous utiliserons le plus souvent le mot clef “current” qui désignera l'objet correspondant au contexte courant dans lequel le script est exécuté.

Exemple :

- `user:current` désigne l'utilisateur correspondant à `$USER`
- `course:current` désigne le cours correspondant à `$COURSE`

Référence à un attribut final

Une référence à un attribut final est construite comme une référence “objet” suivi d'une mention d'attribut. Ces attributs sont le plus souvent les attributs “naturels” de l'objet en base de données. Dans certains cas cependant, et pour aider à l'unification du langage, certains noms d'attributs peuvent être utilisés pour accéder à des données indirectes liées (par exemple, les champs personnalisés de profil d'un compte utilisateur).

Forme générale :

```
[Ref objet]:[attribut]
```

Exemple :

- `user:current:department`
- `course:idnumber:“TR2020”:visible`

Détail des implémentations

Types d'objets autorisés

- user : Un utilisateur
- user_profile_field : un champ de données personnalisé de l'utilisateur
- course : Un cours
- category : Une catégorie de cours
- cohort : Une cohorte

Opérateurs supportés

- = : est égal
- != : est différent
- < ou ⇐ : est inférieur ou inférieur ou égal
- > ou >= : est supérieur ou supérieur ou égal
- ~ : contient (regex)
- !~ : ne contient pas (regex)

Opérateurs "fonctions" supportés

- hasrolein : a un rôle dans le contexte | <userref> hasrolein <courseref|categoryref>
- isenrolledin : a une inscription active dans le contexte | <userref> isenrolledin <courseref|categoryref>
- hasloggedin : s'est connecté au moins une fois | <userref> hasloggedin
- hascompleted : a achevé un contexte | <userref> hascompleted <courseref>
- hasstarted : a commencé à travailler (au sens de l'achèvement) | <userref> hasstarted <courseref|categoryref>
- isincategory : est enfant de la catégorie | <courseref|categoryref> isincategory <categoryref>
- isincattree : est dans l'arborescence | <courseref|categoryref> isincattree <categoryref>

- isinsubs : est dans une des sous-catégories | <courseref|categoryref> isinsubs <categoryref>
- isempty : est vide | <categoryref|cohortref> isempty

Objets à venir (prospective)

- group : Groupe de cours
- module : Module de cours (activité ou ressource)
- badge : Badge

Opérateurs "fonctions" à venir (prospective)

- isin : est dans la cohorte ou le groupe <userref> isin <cohortref|groupref>
- hasbadge : a acquis le badge
- hascompleted : extension au context <moduleref> en plus du contexte <courseref>

Fonctionnalités à venir

- Expressions nommées : Des expressions évaluables associées à un nom pour pouvoir être utilisées à plusieurs endroits et définies de manière unique et centralisée. Par anticipation, on note la nécessité d'une syntaxe facilement discriminable pour aller chercher une expression dans un catalogue. Une proposition peut être :

```
::<evaluablerulename>
```

qui discriminera nettement l'appel de référence des syntaxe “par membres” des appels aux attributs ou aux références d'objet. Le nom de règle doit être un token unique acceptant les caractères alphanumériques et le '_' sans espaces. L'usage de caractères étendus ou accentués est déconseillé.

Ainsi la simple mention de

```
::myrule1
```

ira rechercher la définition complète de la règle dans le catalogue de règles nommées. La localisation du catalogue est dans l'accessoire tool_moodlescript, le plugin habilité à proposer des interfaces administrateur.

A titre de prospective, et afin de favoriser l'usage des règles nommées, on peut envisager un passage de paramètres à une règle nommée, par exemple :

```
::myrule1 <param1> "<string param2>"
```

qui proposera des valeur à substituer dans l'expression nommée sous les formes classiques de variables shell : $1, $2, etc... L'exemple suivant montre ce que pourrait être l'écriture d'une telle expression :

```
user:idnumber:$1:username ~ ^a
```

Pour une règle disant “est ce que le username de l'utilisateur donné en paramètre par $1 commence par 'a' ?”

Revenir à l'index du composant