

Spécification de syntaxe

Moteur de script Moodlescript

Règles de base

La langage MoodleScript se veut un langage simple, facile à lire et à écrire, et qui évite les règles syntaxiques “techno” que l'on trouve dans d'autres principes d'écriture. La syntaxe évitera donc tout recours à des symboles et restera sur un principe de mots clefs explicites afin de garantir la lisibilité naturelle du code.

Une commande moodle sera habituellement composée d'un mot clef, d'arguments fixes, de variables et d'une liste de paramètre/attributs complémentaire. Tous les termes doivent être séparés par au moins un caractère. Une commande commencera toujours par une sémantique d'action (un verbe, ex. ADD, REMOVE, ENROL, BACKUP, etc.) et sera suivie par une alternance de mots-clefs et d'arguments, le tout formant une phrase “lisible” humainement. Certaines commandes pourront accepter une liste supplémentaire d'arguments, notamment pour apporter une liste de valeurs pour un objet à créer par exemple, ou donnant des clauses conditionnelles pour la destruction d'un objet.

La forme générale d'une commande MoodleScript est donc :

```
VERBE MOTCLEF1{1,n} [ [arg1]{0,n} MOTCLEF1{0,n} ]{0,n}
```

Une commande acceptant une liste complémentaire d'attributs sera de la forme:

```
VERBE MOTCLEF1{1,n} [ [arg1]{0,n} MOTCLEF1{0,n} ]{0,n} **HAVING**  
clef1: valeur1  
[clef2: valeur2]  
...  
[clefn: valeurn]
```

Une commande se termine à la première ligne vide rencontrée.

Une commande ne pourra accepter qu'une seule liste d'arguments au plus.

Mots-clefs

Les mots clefs sont des mots à valeur spéciale lorsqu'ils sont positionnés à certains endroits de la commande. Les mots-clefs DOIVENT toujours être écrits en MAJUSCULES.

Les mots clefs sont :

Des verbes (toujours le premier mot d'une commande)

Les verbes sont des mots-clefs à un seul “token” (pas d'espaces) et doivent désigner une action (par ex. ADD ou REMOVE)

Verbes acceptés:

ADD, REMOVE, ENROL, BACKUP, MOVE

Types d'objets administrables

Les types d'objet désignent des objets “administrables” de moodle. Ils peuvent être des mots simples ou des expressions à plusieurs mots (mais toujours dans un ensemble connu).

COURSE, CATEGORY, ENROL METHOD, USER, COHORT, BLOCK, MODULE, etc.

Articulations

Les articulations contextuelles sont des petits mots qui indiquent ce que l'on fait des arguments qui les entourent, afin de faciliter la lisibilité “naturelle” de la commande.

IN, FOR, TO, IF EXISTS, IF NOT EXISTS

Arguments, identifiants et variables

Les arguments sont le plus souvent des identifiants d'objets moodle, des valeurs littérales terminales (qui s'expriment telle qu'écrites) ou éventuellement des appels à des variables globales ^permettant d'obtenir des “valeurs”. La nature et la signification de la valeur peut varier au fil de l'expression, mais se référera toujours à la donnée la plus “logiquement” attendue à cet endroit de la “phase”.

Par exemple, pour une instruction d'inscription, l'expression :

ENROL id:33 IN id:3 AS shortname:student USING manual

montre 4 attributs qui se réfèrent (successivement) à un Utilisateur, un Cours, un Role et une méthode d'inscription.

Dans le cas où une commande viendrait à créer des ambiguïtés, la syntaxe devra comporter des “adverbes” propres à lever cette ambiguïté.

Identifiants

Lorsque l'expression doit identifier un objet déjà existant dans Moodle, et que plusieurs informations permettent d'identifier cet objet, un discriminateur explicite devra être utilisé pour exprimer la valeur, par exemple. pour un utilisateur, il existe actuellement quatre identifiants possibles: l'identifiant numérique primaire de base de données (id), l'identifiant de connexion (username), le numéro d'identification (idnumber) ou l'email.

Les expressions suivantes sont donc utilisables pour désigner un utilisateur dans moodle :

```
id:33
```

```
username:johndoe
```

```
idnumber:JD@35465
```

```
email:john.doe@gmail.com
```

Techniques spéciales d'identification : utiliser une fonction pour obtenir un identifiant

Dans certains usages, nous souhaiterions que l'identifiant soit fourni par une fonction existante (ou nouvelle) dans l'API moodle, qui s'exécutera dans le contexte courant du moteur. L'identifiant fera alors figurer le préfixe supplémentaire 'func', suivi d'un identifiant de fonction localisé dans son plugin comme suit :

```
idnumber:func:local_ent_installer@get_teacher_cat_idnumber
```

L'expression ci-dessus invoquera la fonction

`local_ent_installer_get_teacher_cat_idnumber()` du plugin `local`

`local_ent_installer`, en allant la chercher dans le fichier `locallib.php` (ou `lib.php` par défaut). La fonction devra retourner l'identifiant attendu. Dans l'exemple ci-dessus, la fonction devra retourner le numéro d'identification de l'objet administrable souhaité.

You cannot pass any parameters to this call, so the identifier must be fully determined using current environment globals such as `$USER`, `$COURSE`, etc. to compute the expected identifier.

Here is a sample of an application function that computes the current user's owned category to move a course in:

```
/**
 * Provides an uniform scheme for a teacher category identifier.
 * @param object $user a user object. If user is not given will return the
cat identifier of
 * the current user.
 * @return string
 */
function local_ent_installer_get_teacher_cat_idnumber($user = null) {
    global $USER;

    if (is_null($user)) {
        $user = $USER;
    }

    $teachercatidnum =
strtoupper($user->lastname).'_' . substr(strtoupper($user->firstname), 0, 1).
    $teachercatidnum .= '$'.$user->idnumber.'$CAT';
    return $teachercatidnum;
}
```

Called in a moodlescript stack context, it will compute the category idnumber of the current user, so we can write a moodlescript move instruction as follows, moving the current course to the adequate destination:

```
MOVE COURSE current TO
idnumber:func:local_ent_installer@get_teacher_cat_idnumber
```

Literal Argument

Literal arguments are simple words or strings. There is at the moment a restriction on syntax as strings are not specifically delimited (or only delimited by keywords). So strings should not contain keywords expressions. this is likely why we chose keywords in strict uppercase, to minimize syntactic collision with literal usual strings.

Variables

We may need to inject some environmental values in the script to replace some non terminal placeholders. An execution stack can be fed at launch time with a global context data stub that will be merged with each instruction local context (adding or overriding values). Global context variables can be placed wherever in script statements or attribute lists using the Moodle common SQL named variable form:

```
:varname
```

To be valid, the placeholder expression **MUST** have at least one space character before it.

You may obtain a list of the available variables in the stack logger using the following instruction:

```
LIST GLOBALS
```

this will output, e.g. in the admin tool console, giving the console environment preset variables:

```
> GLOBAL CONTEXT
> wwwroot: http://dev.moodle31.fr
> currentuserid: 2
> currentusername: admin
> sitedshortname: DEV31
```

Any local invocation of a MoodleScript stack may run the stack with his own global environment variable set, to serve some specific component scoped scripting needs.

Special keywords (metas)

'current'

'current' is a special keyword in place of an expected identifier that will resolve into the nearest current object in the executing environment. F.e, if the expected object is a user identifier, current will resolve to \$USER→id. If 'current' addresses a course identifier, it will usually resolve as \$COURSE→id, unless another course id is given to the execution stack by the global context (for plugin developers).

The use of current will simplify scripts run within a known context, by using shorten expressions:

```
ADD ENROL METHOD guest TO current
```

For adding an enrolment method to the current course.

```
ENROL current INTO current AS student
```

For enrolling the current user (the \$USER being executing the script) into the current course.

'last' or 'first'

this usually addresses the first available or last available item in the current syntax context. this is used f.e. for blocks location in a region, but might also address any object location being sorted with a sortorder attribute.

'runtime'

Usually identifiers and variable can be evaluated at parse time or at check time, because they are literals in the script, or they come from some input or global context. But this is not true in all cases. Lets take an example:

In the following scriptlet:

```
ADD CATEGORY "New category" TO idnumber:EXISTINGCAT HAVING  
idnumber: NEWCAT  
  
MOVE COURSE idnumber:SOMECOURSE TO idnumber:NEWCAT
```

We run into an issue because at parse time or at check time, NEWCAT category is not yet created. Thus we must tell the engine that in the second statement, we need the engine waiting the latest moment to evaluate the identifier to move the course in.

This can be done by the special keyword runtime: and we'll rewrite the scriptlet as follows:

```
ADD CATEGORY "New category" TO idnumber:EXISTINGCAT HAVING  
idnumber: NEWCAT  
  
MOVE COURSE idnumber:SOMECOURSE TO runtime:idnumber:NEWCAT
```

Using the runtime: special keyword will prevent the parser to evaluate and resolve the identifier, and will store it's initial definition in the handler class. The handler will also NOT try to resolve it at check time, as check time only checks the conditions of execution of all the statements without executing them. At real execution time, the identifier will be resolved to get it's definitive actual value.

Note that 'runtime' variables may raise a real error situation that cannot be recovered or anticipated by the engine and may terminate in a technical failure of the script and the calling process.

[Revenir à l'index du composant](#)

From:
<https://docs.activeprolearn.com/> - **Documentation Moodle ActiveProLearn**

Permanent link:
<https://docs.activeprolearn.com/doku.php?id=local:moodlescript:syntaxspecification&rev=1553602875>

Last update: **2024/04/04 15:52**

