MoodleScript : Architecture du moteur de script

Le moteur de script est composé d'un plugin local où réside le moteur d'interprétation et d'exécution des scripts et syntaxes associées, et d'un outil d'administration permettant d'écrire, vérifier et exécuter des scripts à la volée.

Concepts d'architecture

Commmande

Une commande est un mot clef apparaissant en tête de l'instruction. Une commande peut se diviser en sous-actions par le deuxième mot de l'instruction si la syntaxe le permet. Chaque sous action donne lieu à l'écriture d'un parser pour vérifier sa syntaxe et d'un handler pour l'exécuter dans une pile d'exécution.

Parser

Le parser est un objet qui consomme un script et exécute le parsing syntaxique des instructions qui y sont identifiées. Le parsing est confié à un parseur spécialisé pour chaque “mot de commande” qui est constitué par le premier “mot” de l'instruction.

La phase de parsing confronte chaque instruction aux modèles syntaxiques autorisés et valide la syntaxe. Au moment de cette interprétation des variables peuvent être créées dans le contexte du parser.

L'exécution du script ne peut être démarrée qu'une fois que le parser a terminé l'examen de TOUTES les instructions du script. Le script est alors déclaré VALIDE.

Le résultat du parsing est une pile de handlers (voir ci-après) associée à un contexte de données préparé pour chaque handler à partir du décodage de la syntaxe de l'instruction.

Handler

Un handler est un objet chargé de procéder à l'exécution de l'instruction qu'il implémente. Il reçoit les données de contexte comme une combinaison :

- Des données de contexte fournie à l'exécuteur de script à son initialisation
- Des données accumulées dans la pile d'exécution
- Des données découvertes par l'examen syntaxique de l'instruction

Contexte

Un contexte est un jeu de données que le parser constitue pour l'instruction, alternativement un jeu de données disponible pour l'exécution.

Runtime

Le Runtime est un contexte particulier pendant l'exécution de la pile. Certaines données ne peuvent être disponibles qu'à la suite d'une exécution particulière (et jamais au moment de la vérification syntaxique). Bien que la construction du moteur soit prévue pour vérifier le plus tôt possible les inconsistances ou incohérences d'écriture des instructions pour faciliter la mise au point des scripts, certaines vérifications ne peuvent se faire qu'au tout dernier moment.

Principe d'anticipation des erreurs : check() et dynamic_check()

Le moteur d'exécution Moodlescript interprète une séquence d'instructions paramétriques dans une pile d'exécution séquentielle. Ces instructions vont modifier les données de Moodle successivement pour accomplir le scénario souhaité. L'exécution de chaque instruction se fait en rapport avec un "contexte" qui apporte la valeur finale des paramètres que consomme l'instruction. Certaines de ces valeurs auront pu être produites par une instruction précédentes et ne peuvent être connues qu'au moment de l'exécution (runtime). Dans des scripts complexes, il est donc difficile d'anticiper certaines erreurs introduite dans la logique du script.

Le moteur Moodlescript tente de répondre à ce problème en proposant une architecture dont le but est précisément de détecter les erreurs potentielles le plus tôt possible. Les situations à couvrir sont les suivantes :

- Les valeurs des paramètres sont écrites "en dur" dans le code, et décrivent des objets ou références existantes : Elles peuvent donc être vérifiées immédiatement dès le "parsing" du code.
- Les valeurs des paramètres sont des expressions "non terminales", mais qui font appel à des symboles statiques (disponibles dès le parsing). Elles peuvent être vérifiées après résolution pendant cette phase.
- Les valeurs des paramètres sont "variables", mais, bien que dépendant des instructions précédentes, ne dépendent que de valeurs statiques ou "solubles" de ces instruction. Là encore, il est possible de vérifier ces valeurs dès le parsing, en simulant les variations du contexte induites par chaque instruction
- Les valeurs sont variables, et ne peuvent être connues que après l'exécution effective de l'instruction. Dans ce dernier cas, il n'est pas possible d'anticiper une erreur sur de telles variables et le script lancera une exception d'exécution.

Plus tôt les erreurs potentielles sont détectées, et moins le risque d'altérer partiellement et incomplètement les données traitées sera grand. Une altération incomplète des données pourrait laisser le système dans un état instable et difficile à récupérer.

La fonction check()

La fonction dynamic_check()

Retour à l'index du plugin - Retour à l'index des ensembles - Retour au catalogue

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

Permanent link:
https://docs.activeprolearn.com/doku.php?id=local:moodlescript:enginearchitecture

Last update: **2025/10/15 10:32**