

Custom Commands et édition des pages By Module

Routeur serveur

La majorité des actions étudiantes passent par `_utils/command/utils_custom_command_by_module.js`. La fonction `executeCommandByModule()` route `commandUid` vers la fonction métier appropriée.

Commandes majeures

Fonction	Responsabilité
<code>startStudentModule</code>	Initialise/démarre un module pour l'équipe.
<code>saveStudentModuleRowInputValue</code>	Sauvegarde une valeur d'input dans une row.
<code>saveStudentModuleRowInputOptions</code>	Sauvegarde les options d'un input.
<code>saveStudentModuleInputFormula</code>	Sauvegarde une formule; voir page dédiée Formules.
<code>add/remove/edit ... Row</code>	Gestion des rows dynamiques et de leur ordre.
<code>add/remove/edit ... Section</code>	Gestion des sections dynamiques.
<code>saveDynamicDropdownData</code>	Persistance des données de dropdown dynamique.
<code>getModuleGrade</code>	Calcule/récupère la correction.
<code>getTeamModuleTypeResults</code>	Retourne les résultats déjà générés.
<code>restartStudentModule</code>	Réinitialise un essai de module selon les règles.
<code>ignoreErrorFromTeamModuleResults</code>	Permet l'override/ignorance d'une erreur de correction.

Rows et sections dynamiques

Lorsqu'une row/section temporaire doit être persistée, le serveur peut lui attribuer un nouvel ObjectId. Il renvoie alors les alias et index à l'UI afin que le client remplace l'UID local. Le serveur maintient aussi les références de formules associées.

Synchronisation du ledger

Après certaines modifications de pages, le code vérifie

`utilsByModule.shouldSyncLedger(moduleSettings, pageUid)`. Si nécessaire, il appelle `rebuildLedgerFromJournals()` puis envoie `sendLedgerPageChangedMessage()`. Une édition dans un journal peut donc modifier indirectement la page ledger.

RTT

Chaque commande émet un message précis, par exemple `team_module_row_input_options_changed`, `team_module_row_removed`, `dynamic_dropdown_data_changed` ou `team_module_input_formula_changed`. Les gros objets ne sont pas toujours envoyés directement : le message `get_data` peut demander au client de relire seulement un chemin d'entité.

Côté Unity

`TeamByModuleController` envoie ces commandes, reçoit les messages et met à jour `TeamModuleTypeData`. Les classes `DataDrivenTableControl`, `DataDrivenRowControl`, `DataDrivenSectionControl` et les contrôles d'input construisent l'interface à partir des données statiques + runtime.

Ajouter une Custom Command By Module de A à Z

1. Définir commandUid et payload

Le commandUid doit être stable et décrire l'action, pas le contrôle UI qui la déclenche. La payload contient généralement `teamUid`, `actModuleUid`, `moduleType` et les données minimales de l'opération.

2. Ajouter le wrapper game-specific si nécessaire

Acc1/Acc2 passent par leur `<game>_custom_command`, qui délègue au routeur By Module commun.

3. Brancher executeCommandByModule()

Ajouter le nouveau cas dans `utils_custom_command_by_module.js`. Extraire la logique complexe dans un helper séparé si elle dépasse une opération simple de page.

4. Modifier le Team Module

Lire le bon `teamsModules`, sélectionner `moduleTypesData[moduleType]`, valider que la page/row/section visée existe puis appliquer la mutation.

5. Recalculer les dépendances

Si l'action touche un journal/formule/dropdown/ledger, exécuter les rebuilds nécessaires avant la sauvegarde. Ne pas compter sur l'UI pour recomposer un état dérivé.

6. Sauvegarder et envoyer RTT

Choisir un message précis dans `MessageTypeByModule`. Pour une grosse structure, préférer un refresh ciblé par chemin plutôt qu'une payload massive.

7. Ajouter le handler Unity

`TeamByModuleController` doit mettre à jour la structure locale exactement comme le serveur. Tester avec deux étudiants dans la même équipe.

Concurrence rows/sections

Les rows/sections dynamiques peuvent être créées avec un UID temporaire côté client puis recevoir un ObjectId serveur. Toute nouvelle commande structurelle doit gérer ce remplacement et les références de formules associées.

Checklist custom command

- mauvais moduleType rejeté proprement;
- UID row/section inexistant géré;
- document sauvegardé avant RTT;
- ledger/formules recalculés si dépendants;
- second client synchronisé;
- relog recharge exactement le même état;
- practice et summative vérifiés.

Revision #2

Created 2026-09-18 19:14:58 UTC by Mathis

Updated 2026-09-18 20:12:24 UTC by Mathis