

Modèle de données et cycle de vie d'un Team Module

Document central

Le document persistant central est `teamsModules`. Côté Unity, `BackEndBrainCloud.GetTeamModule(teamUid, actModuleUid, ...)` appelle `get_document` avec l'UID généré par `UtilsBase.GetTeamModuleUid(teamUid, actModuleUid)`.

Objets principaux

- `TeamModuleBase` : module d'une équipe;
- `TeamModuleTypeData` : données métier d'un type de module;
- `TeamModuleDecisionsData` : pages et dropdowns d'un jeu de décisions;
- `TeamModulePageData` : données persistées d'une page;
- `DataDrivenTableData`, `DataDrivenSectionData`, `DataDrivenInputData` : structure éditale;
- `ResultsByModule` : résultat/correction récupéré par le client.

```
classDiagram
class TeamModuleBase
class TeamModuleTypeData
class TeamModuleDecisionsData
class TeamModulePageData
class DataDrivenTableData
class DataDrivenSectionData
class DataDrivenInputData
TeamModuleBase --> TeamModuleTypeData
TeamModuleTypeData --> TeamModuleDecisionsData
TeamModuleDecisionsData --> TeamModulePageData
TeamModulePageData --> DataDrivenTableData
DataDrivenTableData --> DataDrivenSectionData
```

Ce que TeamModuleTypeData peut porter

Selon le module, la structure contient notamment seed, bills, finished bills, pagesData, dynamicDropdownsData, comptes initiaux, inventory, immobilizations, contracts, seeded products/services, taxes, ownership et snapshots de contextes.

Practice / Summative

Un même Team Module peut avoir plusieurs `moduleTypesData`, notamment `practice` et `summative`. La plupart des custom commands reçoivent donc systématiquement `moduleType` en plus de `teamUid` et `actModuleUid`.

Contexts

Le framework supporte des données héritées entre contextes. Les champs `savedPagesDataByContextUids`, `initialPagesData` et les équivalents de dropdowns permettent de transporter ou préremplir l'état d'un contexte antérieur.

Versioning

Tout changement structurel du document doit être accompagné d'une migration dans `_utils/version/version_teams_modules.js`. Ce fichier contient déjà une chaîne de migrations successives.

Ajouter un nouveau champ TeamModuleTypeData

1. Définir si le champ est **généré, édité par l'étudiant, dérivé** ou **hérité d'un contexte précédent**.
2. Ajouter le champ côté serveur dans l'objet produit par Create Module Type Data.
3. Ajouter le champ dans le modèle Unity `TeamModuleTypeData.cs` si le client doit le lire.
4. Ajouter la désérialisation avec une valeur par défaut pour les vieux documents.
5. Si l'étudiant peut modifier le champ, ajouter une custom command dédiée ou réutiliser le système data-driven approprié.
6. Ajouter RTT si les coéquipiers doivent voir le changement immédiatement.
7. Ajouter une migration dans `version_teams_modules.js` si le champ doit exister dans les documents persistés historiques.
8. Vérifier practice et summative séparément.
9. Vérifier les backups/reopen/restart module si ce champ doit être restauré.

Choisir où mettre une donnée

Donnée	Emplacement recommandé
Structure immuable du module	Static data / settings.
Donnée générée par seed pour une équipe	<code>TeamModuleTypeData</code> .
Valeurs saisies dans les pages	<code>TeamModuleDecisionsData.pagesData</code> .
Options dynamiques de dropdown	<code>dynamicDropdownsData</code> .
Résultat/correction	document teams / results structures, pas dans les décisions.
Donnée à partager entre contextes	saved/initial context data ou champ cumulatif explicitement géré.

Reopen / restart / backup

Une nouvelle donnée persistante doit être testée avec `restartStudentModule`, `restore_module_backup` et `reopen_module` si ces flows doivent la restaurer ou la réinitialiser. C'est un trou classique lorsqu'on ajoute seulement la génération initiale.

Revision #2

Created 2026-09-18 19:14:56 UTC by Mathis

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