

Création du Module Type Data

Point d'entrée

Le moteur est `_utils/by_module/byModule_utils_create_module_type_data.js`. Il fabrique les données déterministes d'un module à partir du seed, des act settings, module settings et page settings.

```
flowchart TD
  IN[seed + settings] --> INIT[initialisation RNG et contexte]
  INIT --> GEN[génération données métier]
  GEN --> BILL[bills / partenaires / produits]
  GEN --> INV[inventory]
  GEN --> IMM[immobilizations]
  GEN --> CON[contracts]
  INIT --> PREV[contextes précédents]
  PREV --> FIX[fixed pages]
  PREV --> INITIAL[initial pages]
  GEN --> PRE[precomputed pages]
  FIX --> OUT[TeamModuleTypeData]
  INITIAL --> OUT
  PRE --> OUT
```

Initial pages

`getInitialPagesData()` lit `current_module.initial_pages_data`. Une référence préfixée par `previous_` permet de réutiliser une page d'un contexte antérieur sous l'UID du contexte courant.

Fixed et precomputed data

`utilsByModule.getFixedData()` extrait des données fixes depuis l'answer key précédente.
`getPrecomputedData()` peut calculer des données attendues à l'avance à partir de l'answer key courante.

Données cumulatives

Le moteur peut transporter entre contextes l'inventaire, les immobilisations/dépréciations, les contrats et des snapshots de pages/dropdowns. C'est ici qu'il faut chercher un bug où un nouveau module démarre avec de mauvaises valeurs héritées.

Fichiers liés

- `byModule_utils_create_module_type_data.js` ;
- `functions/byModule_utils_create_module_type_data_functions.js` ;
- `byModule_utils.js` ;
- `classes/InventoryProduct.js` ;
- `classes/bills/*` et `BillFactory.js` ;
- `acc_base/utils/acc_base_utils.js`.

Validation

Utiliser `tests/unit/by_module_tests/create_module_type_data_tests` et, pour une reproduction complète, `testGeneralFunctionalities(moduleUid)` dans le debugger local des Server Scripts.

Carte interne du moteur de création

Le fichier principal est `byModule_utils_create_module_type_data.js`, avec les helpers dans `functions/byModule_utils_create_module_type_data_functions.js`. Pour debugger un module qui “naît mal”, il faut identifier à quelle étape la mauvaise valeur apparaît.

Fonction	Responsabilité
<code>byModuleCreateModuleTypeData()</code>	Entrée publique appelée avec seed + settings.

Fonction	Responsabilité
<code>createModuleTypeData()</code>	Orchestration principale.
<code>initCreateModuleTypeData()</code>	Initialise RNG, settings, dates, static data et contexte de calcul.
<code>getContextData()</code>	Résout les données nécessaires au contexte courant.
<code>createInitialModuleTypeData()</code>	Construit les structures initiales métier.
<code>createInventory()</code>	Initialise l'inventaire lorsque le module utilise du stock.
<code>createBills()</code>	Génère les bills du contexte via settings/factories.
<code>postProcessContextAccounts()</code>	Finalise les comptes après génération du contexte.
<code>getInitialPagesData()</code>	Construit les pages initiales, y compris les références <code>previous_*</code> .
<code>initPagesData()</code>	Crée la structure runtime des pages/inputs.
<code>getCurrentDepreciations()</code>	Récupère les amortissements à propager.
<code>getCumulativeDepreciationFromYear()</code>	Calcule la dépréciation cumulée à transporter.

Ajouter une nouvelle donnée générée de A à Z

1. **Définir sa source** : static data, preference, previous context ou RNG.
2. **Définir où elle vit** dans `TeamModuleTypeData`.
3. **Initialiser la valeur** dans le moteur de création au bon moment.
4. **Si elle dépend d'un contexte précédent**, ajouter la logique de copie/translation dans `getContextData()` / `getInitialPagesData()` ou l'helper spécialisé.
5. **Si elle dépend du seed**, utiliser le RNG existant; ne pas appeler `Math.random()`.
6. **Si elle alimente les bills**, la créer avant `createBills()`.
7. **Si elle alimente l'Answer Key**, vérifier que le moteur de correction retrouve exactement la même structure.
8. **Ajouter la désérialisation Unity** si le client doit la lire.
9. **Ajouter une migration teamsModules** si des documents existants doivent recevoir le nouveau champ.
10. **Ajouter un test** dans `create_module_type_data_tests`.

Déterminisme : règle absolue

Le seed sert à pouvoir recréer les mêmes données, Answer Key et résultats. Une nouvelle branche de génération ne doit pas introduire un appel aléatoire hors du RNG contrôlé ni changer silencieusement l'ordre des tirages d'un module existant sans mesurer l'impact.

Contextes précédents

Les champs `previous_*`, `savedPagesDataByContextUids`, `savedDynamicDropdownsDataByContextUids`, `inventory`, `immobilizations` et `contracts` peuvent traverser les contextes. Avant de créer une nouvelle donnée “cumulative”, définir explicitement :

- est-elle copiée telle quelle ?
- recalculée ?
- transformée vers l'UID du nouveau contexte ?
- ignorée en practice/summative ?

Debug pas à pas

1. Fixer un seed reproductible.
2. Logger/inspecter les settings résolus.
3. Comparer les données seedées avant bills.
4. Comparer après `createBills()`.
5. Inspecter `inventory/immobilizations/contracts`.
6. Inspecter `initialPagesData` et dynamic dropdowns.
7. Seulement ensuite lancer Answer Key.

Le helper `testGeneralFunctionalities(moduleUid)` du debugger local est idéal pour rejouer cette chaîne sans passer par Unity.

Revision #3

Created 2026-09-18 19:14:59 UTC by Mathis

Updated 2026-09-18 20:11:11 UTC by Mathis