

UI data-driven, pages, tables et prefabs By Module

Rôle

L'UI By Module est générée à partir de `module_pages.json`, des `pages_settings` et des données runtime du `TeamModuleTypeData`. La structure visuelle n'est donc pas entièrement codée en dur dans les prefabs.

Hiérarchie UI

```
flowchart TD
    M[TeamModuleControl] --> PAGE[ModulePageControlBase]
    PAGE --> TABLE[ModulePageTableControlBase]
    TABLE --> SEC[Section / EditableSection]
    SEC --> ROW[Row / EditableRow]
    ROW --> CELL[CellControlBase]
    CELL --> INPUT[InputControlBase]
    INPUT --> TEXT[TextInput]
    INPUT --> DROP[Dropdown / AutoComplete]
    INPUT --> CHECK[Checkbox / Toggle]
```

Classes structurantes

Responsabilité	Classes
Liste/modules	<code>TeamModuleListControl</code> , <code>TeamModuleItemControl</code> , <code>TeamModuleControl</code> , <code>TeamModuleMenuControl</code>
Pages	<code>ModulePageControlBase</code> , <code>ModulePageInputsControl</code> , <code>ModulePageTableControl*</code>

Responsabilité	Classes
Tables data-driven	<code>DataDrivenTableControl</code> , <code>DataDrivenSectionControl</code> , <code>DataDrivenRowControl</code> , <code>DataDrivenInputsControl</code>
Rows/sections dynamiques	<code>EditableRowControlBase</code> , <code>EditableSectionControlBase</code> , <code>CustomRowControl</code>
Inputs	<code>InputControlBase</code> , <code>InputCellControl</code> , <code>TextInputControl</code> , <code>DropdownInputControl</code> , autocomplete, checkbox
Correction	<code>CorrectedTeamModuleWindowControl</code> , <code>ModuleErrorsPopupControl</code> , <code>CellErrorButtonControl</code>

Types de pages

Le modèle Unity expose plusieurs classes de données : `ModulePageDataInputs`, `ModulePageDataTable`, `ModulePageDataTableWithInputs`, `ModulePageDataTableWithDropdown`, etc. Le type configuré dans la static data détermine quel contrôle Unity est utilisé.

Pages Accounting spécialisées

En plus du moteur générique, le dossier `Assets/Src/UI/SimByModule/UI/` contient les vues spécialisées : trial balance, balance sheet, income statement, statement of equity, bank reconciliation, formulaires de taxes, reports d'employés, immobilisations, contrats, bills et autres documents.

Frontend JSON → runtime

```
flowchart LR
    MP[module_pages.json] --> SD[StaticDataByModule]
    PS[pages_settings.json] --> SD
    SD --> PC[ModulePageControl]
    TM[TeamModuleTypeData.pagesData] --> PC
    PC --> INPUTS[Rows / Cells / Inputs]
    INPUTS --> CTRL[TeamByModuleController]
    CTRL --> SERVER[Custom Command]
```

Quand modifier quoi

- Structure d'une page : `module_pages.json`.
- Comportement dynamique/validation : `pages_settings.json` et définitions associées.
- Nouveau rendu visuel réutilisable : classe/prefab sous `SimByModule/UI`.
- Donnée persistée : modèle `DataDriven*` + custom command serveur.
- Nouvelle page Accounting réellement spécialisée : ajouter le contrôle dédié et le brancher dans le système de page correspondant.

Ajouter un nouveau type de page ou de contenu

Avant de créer un nouveau type, vérifier si la page peut être exprimée avec les types existants :

`table`, `table_with_dropdown`, `table_with_dynamic_dropdown`, `table_with_inputs`, `inputs` ou `table_inputs_with_dropdown`.

Si un type existant suffit

1. Déclarer la page dans `module_pages.json`.
2. Configurer structure/inputs/columns et `pages_settings`.
3. Ajouter un prefab custom seulement si le type `inputs` en a besoin via les références de prefab.
4. Vérifier rendu normal + rendu corrected.

Si un vrai nouveau ModulePageType est nécessaire

1. Ajouter la string dans `ModulePageTypeConstants`.
2. Ajouter la valeur dans `ModulePageBase.ModulePageType`.
3. Mettre à jour `GetModulePageType()` et `GetModulePageTypeName()`.
4. Ajouter le modèle `ModulePageData*` si la structure JSON diffère.
5. Mettre à jour `Factory.CreateModulePageData(type)`.
6. Créer/adapter le `ModulePageControlBase` correspondant.
7. Ajouter le prefab template aux listes utilisées par `ReferencedModuleWindowControl` et `CorrectedTeamModuleWindowControl`.

8. Mettre à jour `UtilsByModule.GetModulePageControlPoolType()` si le nouveau type peut partager un pool.
9. Ajouter le rendu de contenu dans `AllocModulePageContentTemplate()`.
10. Ajouter Answer Key + Grading si la page est évaluée.

Ajouter un nouvel input data-driven

Un nouvel input implique généralement quatre zones : modèle de définition static data, modèle runtime de valeur/options, contrôle Unity, et custom command de sauvegarde. Si l'input possède des options de correction, vérifier aussi le grading options.

Rendu corrected à ne pas oublier

Une page qui fonctionne pendant la prise de décision mais plante/est vide en correction est souvent un nouveau type branché seulement dans la fenêtre normale. Toujours tester

`CorrectedTeamModuleWindowControl` et les `CellErrorButtonControl / expectedValues`.

Checklist UI By Module

- page normale;
- page corrected;
- practice;
- summative;
- resize/window;
- export Excel si applicable;
- dynamic rows/sections;
- formules si l'input les supporte;
- RTT avec deux clients.

Revision #2

Created 2026-09-18 19:16:28 UTC by Mathis

Updated 2026-09-18 20:12:22 UTC by Mathis