

acc_base, acc1 et acc2 — séparation des responsabilités

Principe

La vraie logique Accounting partagée côté serveur vit dans `acc_base`. `acc1` et `acc2` sont des wrappers quiinstancient la même base avec un `GameUid` différent et exposent les commandes sous le namespace propre au jeu.

Utils

```
classDiagram
class GameSpecificUtilsAccBase
class GameSpecificUtilsAcc1
class GameSpecificUtilsAcc2
GameSpecificUtilsAccBase <|-- GameSpecificUtilsAcc1
GameSpecificUtilsAccBase <|-- GameSpecificUtilsAcc2
```

`GameSpecificUtilsAcc1()` appelle `GameSpecificUtilsAccBase.call(this, "acc1")`; `Acc2` fait la même chose avec `"acc2"`. Les deux enregistrent ensuite leur instance dans `utilsByGameUid` et exposent `utilsAccGameSpecific`.

Ce qui appartient à acc_base

- constantes et préférences Accounting;
- mapping bill → buy/sell;
- partenaires et comptes impactés;

- seed de Team Module;
- amortissement d'immobilisations;
- calculs contrats/régularisations;
- settings régionaux;
- résolution act/module/page settings;
- comptes utilisés et structure de pages;
- conditions/visibilité;
- prix et helpers partagés.

Command wrappers

Les dossiers `acc_base/command`, `acc1/command` et `acc2/command` contiennent des scripts de mêmes familles : answer key, generate results, custom command, reopen module, correction mode, etc. Les wrappers Acc1/Acc2 assurent surtout le namespace et la sélection des static data game-specific.

Unity

La même règle s'applique : le framework se trouve dans `Assets/Src/Game/Data/ByModule` et `Assets/Src/UI/SimByModule`; seules les vraies différences de contenu/assets vont sous `Assets/GameSpecific/acc1` ou `acc2`.

Règle de maintenance

Si une modification doit fonctionner de la même façon dans Acc1 et Acc2, elle ne doit pas être dupliquée dans deux fichiers game-specific. La placer dans le framework By Module ou `acc_base` selon qu'elle est générique au framework ou spécifique au domaine Accounting.

Où placer une nouvelle règle Accounting

Règle	Emplacement
Valable pour tout By Module, même un futur jeu non Accounting	<code>_utils/by_module</code> .

Règle	Emplacement
Spécifique à la comptabilité mais commune Acc1/Acc2	<code>acc_base</code> .
Différence réelle Acc1 seulement	<code>acc1</code> .
Différence réelle Acc2 seulement	<code>acc2</code> .

Ajouter un helper Accounting partagé

1. Ajouter la fonction à `GameSpecificUtilsAccBase`.
2. Utiliser `this.*` lorsque la fonction fait partie de l'API de l'instance.
3. Réutiliser les constantes/preferences déjà exposées par la base.
4. Vérifier que `GameSpecificUtilsAcc1` et Acc2 héritent automatiquement du comportement.
5. Tester les deux jeux, car leurs static data peuvent différer même si le code est commun.

Éviter la duplication acc1/acc2

Si deux fichiers game-specific contiennent la même fonction à quelques valeurs près, déplacer la logique dans `acc_base` et passer la différence par static data/settings. Les wrappers game-specific doivent rester minces.

Debug d'un “utilsAccGameSpecific is not defined”

1. Vérifier que le wrapper game-specific inclut son fichier utils.
2. Vérifier que l'instance `GameSpecificUtilsAcc1/Acc2` a été construite.
3. Vérifier l'ordre des `bridge.include()`.
4. Dans le debugger local, inclure le même game-specific que le script réel.
5. Ne pas supposer que le test runner est neutre : il inclut actuellement Acc2 dans son bootstrap.

Revision #2

Created 2026-09-18 19:17:36 UTC by Mathis

Updated 2026-09-18 20:13:33 UTC by Mathis