

Inventaire et stocks

Modèle serveur

`_utils/by_module/classes/InventoryProduct.js` représente un produit d'inventaire. Il gère les entrées de stock, achats, ventes, méthode FIFO/non-FIFO et post-traitement selon le type de stock.

Fonctions structurantes

- `deserialize()` / `serialize()`;
- `buyItem(quantity, price, billUid)`;
- `sellItem(quantity, deleteEntry)`;
- `addInventoryEntry(...)`;
- `initializeInventoryProduct(...)`;
- `postProcessInventoryEntries(...)`;
- `getSellReport(quantity)`;
- helpers de recherche/déduction par bill ou balance.

Modèle Unity

`Assets/Src/UI/SimByModule/Data/Runtime/InventoryProduct.cs` désérialise `isFIFO`, `sellPrice`, `numDecimals` et la liste `InventoryEntries`. Il expose notamment le prix de vente et le coût d'achat courant.

Flux

flowchart LR

STATIC[products / module settings] --> CREATE[Create Module Type Data]

CREATE --> INV[InventoryProduct]

BILL[Purchase/Sale Bills] --> INV

```
INV --> ENTRIES[Inventory entries]
ENTRIES --> COST[Cost of goods / stock]
COST --> REPORT[End Stock of Goods Report]
COST --> AK[Answer Key / accounts]
```

Permanent vs periodic stock

Les constantes By Module distinguent `permanent_stock` et `periodic_stock`. Les préférences de module et les traitements de fin de période déterminent comment l'inventaire influence les comptes et rapports.

Report de fin de stock

Le serveur possède `EndStockOfGoodsReportBill`, qui conserve les variations de stock. Côté Unity, `BillDataEndStockOfGoodsReport` représente ces données.

Tests

`tests/unit/by_module_tests/classes_tests/inventory.test.js` couvre la classe d'inventaire. Les changements qui touchent l'inventaire via les bills doivent aussi passer les tests bills + answer key.

Modifier une règle d'inventaire ou ajouter un comportement stock

Une règle de stock peut impacter génération, bills, account state, reports, Answer Key et Grading. Toujours tracer le cycle complet avant de modifier `InventoryProduct`.

Flux à respecter

```
flowchart LR
CFG[stock_type / settings] --> CREATE[createInventory]
CREATE --> INV[InventoryProduct]
```

```
BILLS[Purchase/Sale Bills] --> INV
INV --> COGS[Cost / balances]
COGS --> REPORT[EndStockOfGoodsReport]
COGS --> ACC[Account State]
ACC --> AK[Answer Key]
AK --> G[Grading]
```

Permanent vs periodic

`stock_type` distingue `permanent_stock` et `periodic_stock`. Ne pas ajouter une branche métier qui ignore ce setting : les comptes et la régularisation de fin de période ne se comportent pas de la même façon.

Ajouter/modifier une méthode de consommation

1. Modifier `InventoryProduct.sellItem()` / `postProcessInventoryEntries()` selon le besoin.
2. Vérifier FIFO/non-FIFO.
3. Vérifier quantités partielles et stock insuffisant.
4. Vérifier que `getSellReport()` retourne encore les coûts attendus.
5. Mettre à jour les bills/reports qui consomment ces données.
6. Vérifier l'account state et les écritures de stock.
7. Ajouter des tests dans `classes_tests/inventory.test.js`.

Cas limites à tester

- stock initial = 0;
- plusieurs achats à prix différents;
- vente traversant plusieurs lots FIFO;
- quantités décimales selon `numDecimals`;
- periodic stock avec régularisation de fin;
- contextes suivants qui reprennent l'inventaire précédent;
- reopen/restart module.

Revision #2

Created 2026-09-18 19:16:35 UTC by Mathis

Updated 2026-09-18 20:13:35 UTC by Mathis