Document global : Architecture, Annotations, Exceptions et Design Patterns (FR/EN)

==========

FR : Architecture Générale

- Modularité par étapes (steps) avec héritage abstrait.
- Injection dynamique via Spring (ApplicationContext).
- Plan d'exécution centralisé dans une Map.
- Point d'entrée unique via Launcher.

FR : Annotations utilisées

- @Column : mapping colonne/champ.
- @InputData : configuration des fichiers sources.
- @OutputData : export des résultats.
- @Referential : injection automatique des référentiels (List, Map, Broadcast).

FR : Gestion des Exceptions par Evénements

- Utilisation de StepExceptionEvent, ReferentialLoadExceptionEvent.
- Handlers pour logs, retry, fallback.

FR : Design Patterns Java utilisés

---

1. Strategy Pattern : chaque Step est interchangeable et suit une interface commune.

2. Factory Pattern : création dynamique des Steps depuis le plan.

3. Singleton Pattern : SparkSession, ApplicationContext.

4. Template Method Pattern : via AbstractStep définissant une méthode d'exécution standard.

5. Dependency Injection (Spring) : pour décorréler la logique et les dépendances.

6. Observer/Event Pattern : gestion des exceptions par événements.

EN : General Architecture

- Modular step execution via abstract inheritance.

- Dynamic injection through Spring ApplicationContext.

- Execution plan stored as a centralized Map.

- Single entry point via Launcher.

EN : Annotations used

- @Column: maps class fields to data columns.

- @InputData: input source configuration.

- @OutputData: output destination.

- @Referential: auto-inject referentials (List, Map, Broadcast).

EN : Event-based Exception Handling

- Using StepExceptionEvent, ReferentialLoadExceptionEvent.

- Handlers enable retry, fallback or logging logic.

EN : Java Design Patterns Used

---

1. Strategy Pattern: interchangeable steps implementing common interface.

2. Factory Pattern: dynamic step instantiation based on config.

3. Singleton Pattern: SparkSession, ApplicationContext.

4. Template Method Pattern: abstract steps define standard flow.

5. Dependency Injection (Spring): decoupling logic and dependencies.

6. Observer/Event Pattern: error events and listeners.