Annotations de mapping et gestion des exceptions par evenement (FR/EN)

==========

FR : Annotations principales

---

1. @Column :

- Mappe un champ Java a une colonne source/cible.
- Attributs : name, position, format (date, nombre, etc.)

2. @InputData :

- Definie les proprietes du fichier d'entree.
- Attributs : path, format (CSV, Parquet...), options (delimiter, header...)

3. @OutputData :

- Definie les proprietes du fichier/table de sortie.
- Attributs : path, format, mode, partitionBy

4. @Referential :

- Injecte automatiquement des referentiels : List<T>, Map<K,T>, Broadcast<List<T>>.
- Options : broadcast, lazyLoading, verifyUniqueKey
- Detection automatique de la cle (avec ou sans @Key/@ComplexKey)

FR : Gestion des exceptions

---

- Chaque erreur dans un Step est capturee et transformee en evenement.
- Evenements emis : StepExceptionEvent, ReferentialLoadExceptionEvent, etc.

- Les handlers ecoutent ces evenements pour logger, relancer ou declencher une logique secondaire.

EN: Main Annotations

---

1. @Column:

   - Maps a Java field to a source/target column.

   - Attributes: name, position, format (date, number...)

2. @InputData:

   - Declares input data source configuration.

   - Attributes: path, format (CSV, Parquet...), options (delimiter, header...)

3. @OutputData:

   - Declares output data target configuration.

   - Attributes: path, format, mode, partitionBy

4. @Referential:

   - Auto-injects referentials: List<T>, Map<K,T>, Broadcast<List<T>>.

   - Options: broadcast, lazyLoading, verifyUniqueKey

   - Auto-detects key field (even without @Key/@ComplexKey)

EN: Exception Handling

---

- Exceptions inside steps are captured and emitted as events.

- Events like StepExceptionEvent or ReferentialLoadExceptionEvent.

- Handlers can log, retry, or trigger fallback logic.

Conclusion

---

- Mapping by annotation improves readability.

- Referential loading is fully declarative and type-safe.

- Event-driven exception handling improves traceability and resilience.