

# SPRING BATCH



**UP ASI  
Bureau E204**

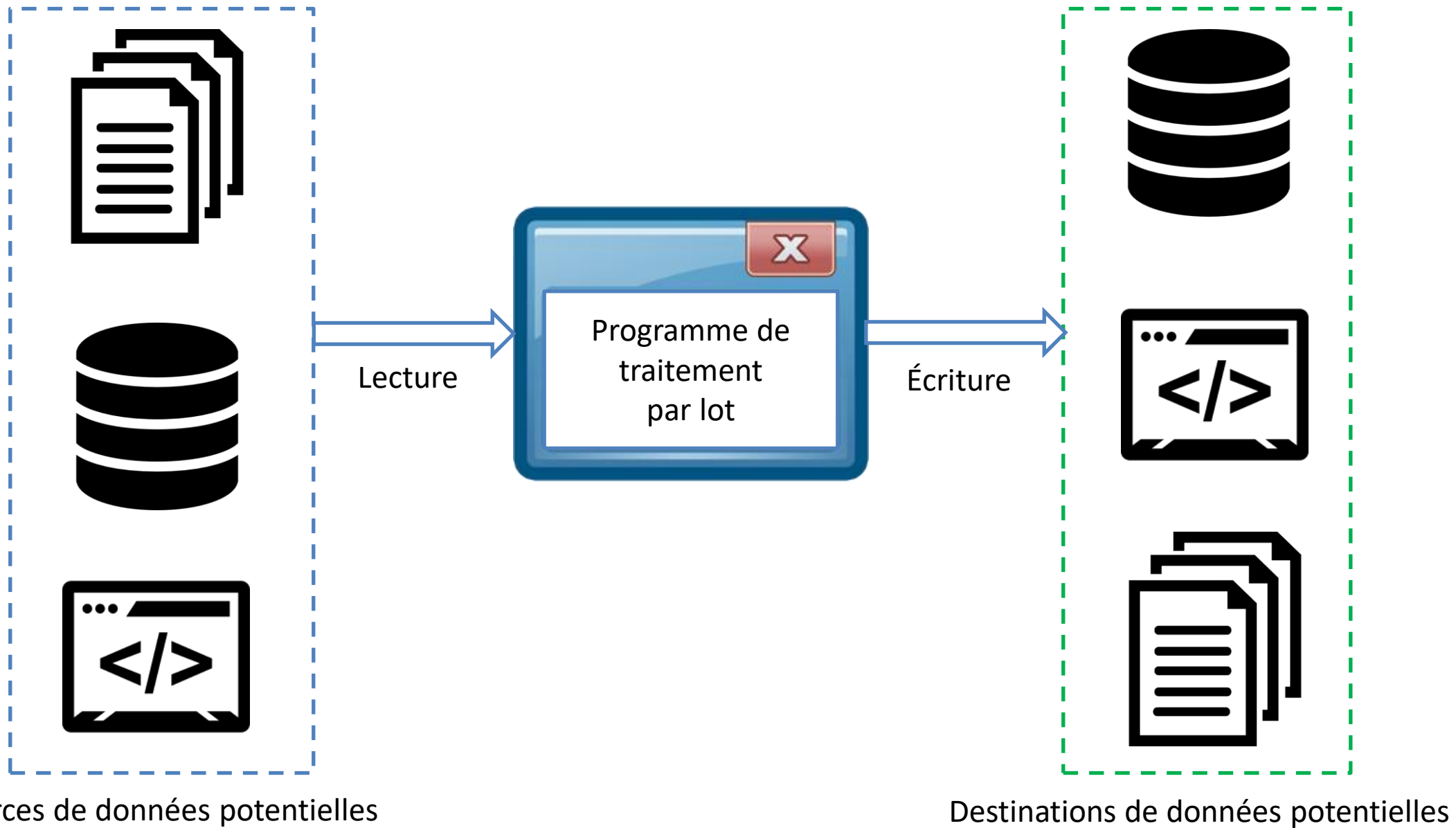
# PLAN DU COUS

- Introduction
- Spring Batch
- Pourquoi utiliser Spring Batch ?
- Architecture de base
- TP: Mettre en place un job
- Travail à faire

# Introduction

- Le mot Batch fait référence à un traitement sur un gros volume de données.
- Un Batch a pour objectif de :
  1. Lire des données provenant de plusieurs sources homogènes ou hétérogènes (fichiers, bases de données, etc).
  2. Faire les traitements nécessaires sur ces données.
  3. Stocker le résultat dans un ou plusieurs conteneurs de destination (fichiers, bases de données, queue, etc..) dans le but de les exploiter.

# Introduction



# Introduction

- Le batch permet d'automatiser une suite de commandes exécutées en série (lots) sur un ordinateur sans une intervention d'un employé pour réaliser cette opération.
- Il est utilisé pour automatiser certaines tâches comme :
  - La création de données facilitant l'aide à la décision ( base de données décisionnelle) **Exp : chiffre affaire par mois- année-catégorie Client-région**
  - L'arrêt planifié des ordinateurs
  - La sauvegarde de milliers de lignes dans une table de la BD
  - Exécuter des instructions DOS (Création, modification et suppression des fichiers / Formatage d'un disque /...)

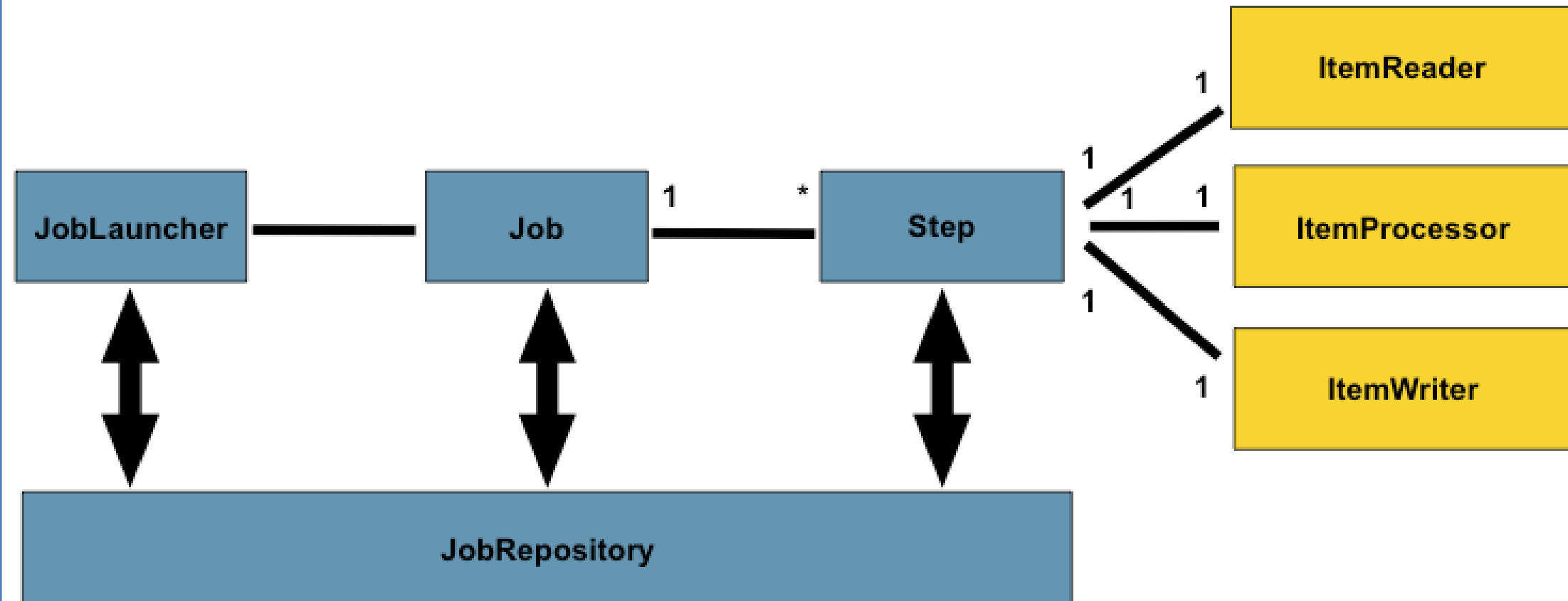
# Introduction

- Spring Batch est un framework sophistiqué open source capable **d'exécuter des tâches par lots** dans des environnements d'entreprise.
- Il permet de simplifier la création des jobs grâce à un code modulaire et facilement maintenable
- Il permet de corriger des problèmes récurrents :
  - ✓ Productivité ( gérer des volumes importants en un minimum de temps)
  - ✓ Gestion de gros volumes de données (temps de traitement très lent)
  - ✓ Fiabilité (problème de pertes d'informations)

# Pourquoi utiliser Spring Batch ?

- Spring Batch facilite le développement de batch à travers des outils dédiés. Il permet ainsi :
  - **Une division du code** bien défini permettant une maintenabilité facilitée et une logique commune à la notion des batchs (classes avec des rôles bien spécifiés à l'avance).
  - Le **traitement d'un gros volume de données** par lots tout en allégeant les charges des différentes instances sollicitées.

# Architecture de base





# Architecture de base - COMPOSANTS

- Pour gérer les données d'un batch, on utilise principalement les trois outils suivants:
  - ✓ **JobLauncher** : Il est chargé d'exécuter un **Job**. Le déclenchement peut être automatique (auto-déclenchement) ou manuel (script, web service, etc..).
  - ✓ **Job** : C'est le composant qui représente la tâche à qui on délègue la responsabilité du besoin métier traité dans le programme. Il permet de lancer un ou plusieurs step ordonnancés ( avec un ordre précis).

# Architecture de base - COMPOSANTS

- ✓ **Step** : Il représente une étape au sein du batch. Les Steps sont généralement stockées dans des **beans** pour pouvoir y accéder facilement dans différents Job.

Il est chargé de définir trois sous-composants :

- **ItemReader**
- **ItemProcessor**
- **ItemWriter**

# Architecture de base - COMPOSANTS

- Ces trois composants opèrent avec l'ordre suivant :
  1. **ItemReader** récupère les données d'entrées à traiter. Elles peuvent provenir de diverses sources (bases de données, csv, xml, xls, ...).

# Architecture de base - COMPOSANTS

- Ces trois composants opèrent avec l'ordre suivant :

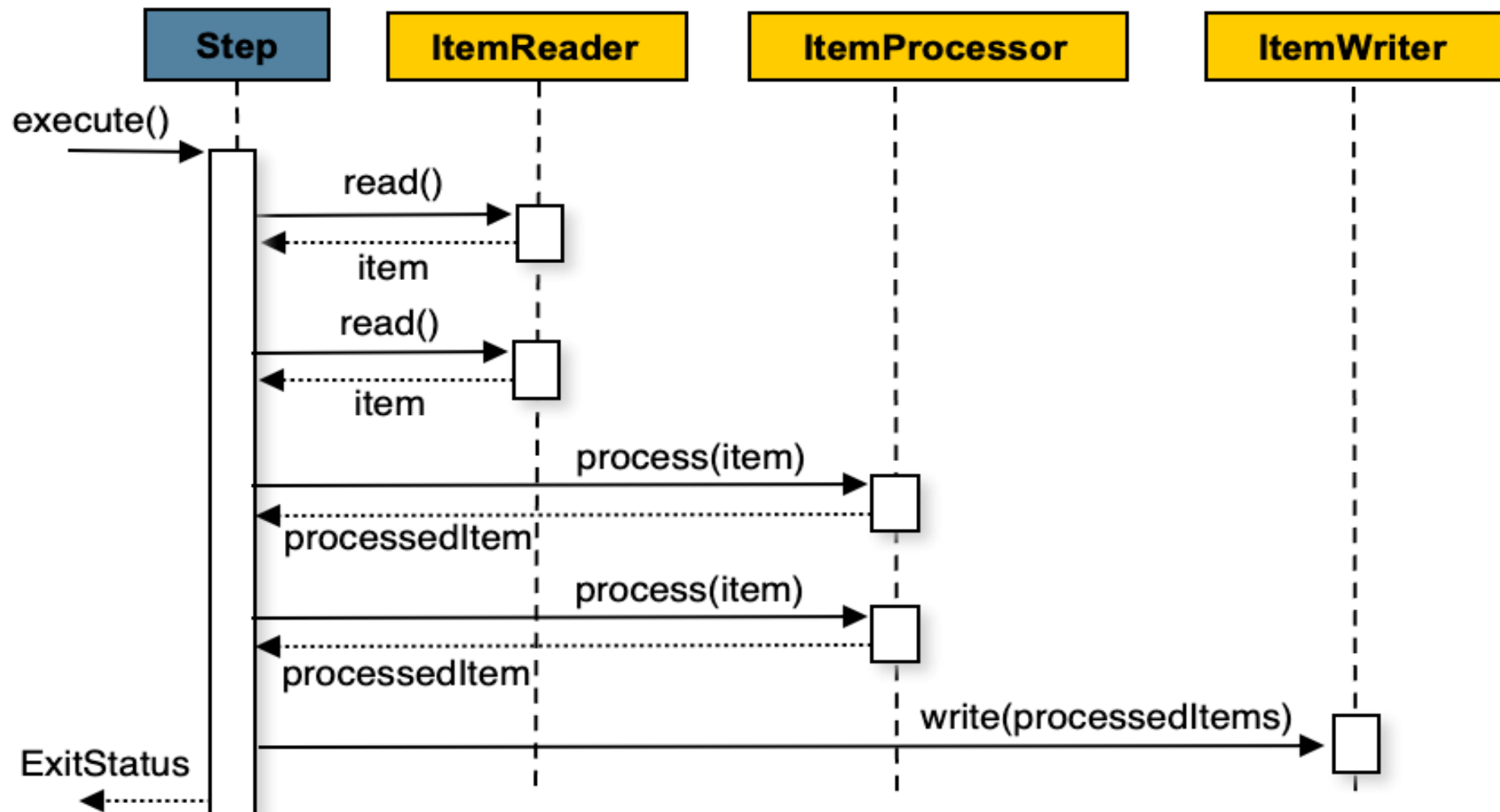
**2. ItemProcessor** contient la logique du traitement des données récupérées par l'ItemReader.

# Architecture de base - COMPOSANTS

- Ces trois composants opèrent avec l'ordre suivant :

**3. ItemWriter** : Il se charge de sauvegarder les données en sortie de l'ItemProcessor dans une ou plusieurs destinations désirés (bases de données, csv, xml, xls, cloud, ...).

# Architecture de base - COMPOSANTS

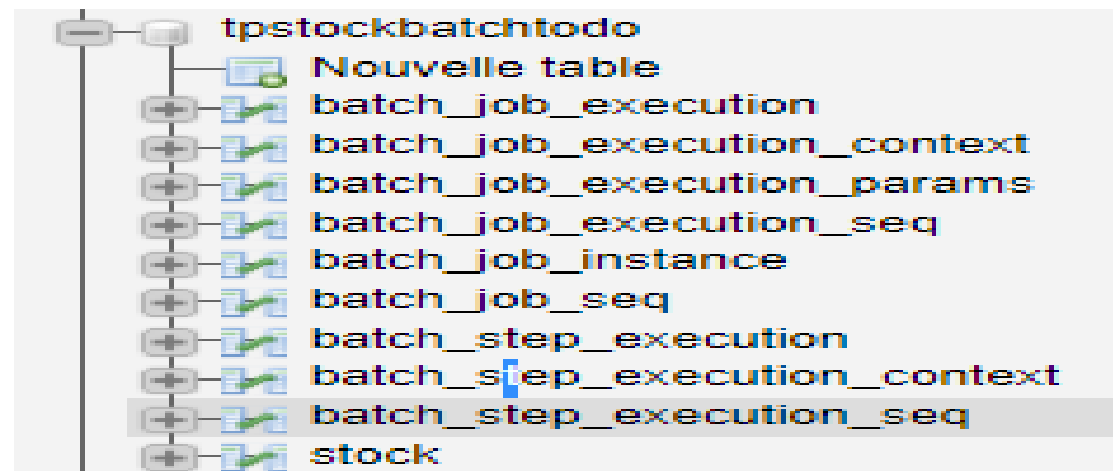


# Architecture de base - COMPOSANTS

**JobRepository** : Le JobRepository est la classe qui permet le stockage d'un nombre important de données autour du Job.

Il permet ainsi de récupérer **un historique des différents jobs** qui ont été lancés.

La sauvegarde de données faite par cette classe permettra donc de **redémarrer des Jobs plus efficacement** ou bien effectuer des pauses dans le traitement.



# Architecture de base - COMPOSANTS

## JobRepository

```
SELECT * FROM `batch_job_execution`
```

☐ Profilage [ [En ligne](#) ] [ [Modifier](#) ] [ [Expliquer SQL](#) ] [ [Créer source PHP](#) ] [ [Actualiser](#) ]

Nombre de lignes : 25 ▼

Trier sur l'index: Aucune ▼

+ Options

←T→ ▼	JOB_EXECUTION_ID	VERSION	JOB_INSTANCE_ID	CREATE_TIME	START_TIME	END_TIME	STATUS	EXIT_CODE	EXIT_MESSAGE
<input type="checkbox"/> Modifier  Copier  Effacer	3	2	3	2021-11-23 19:07:48	2021-11-23 19:07:48	2021-11-23 19:07:49	COMPLETED	COMPLETED	
<input type="checkbox"/> Modifier  Copier  Effacer	4	2	4	2021-11-23 19:07:48	2021-11-23 19:07:48	2021-11-23 19:07:49	COMPLETED	COMPLETED	
<input type="checkbox"/> Modifier  Copier  Effacer	5	2	4	2021-11-23 19:14:51	2021-11-23 19:14:51	2021-11-23 19:14:52	COMPLETED	NOOP	All steps already completed or no steps configured...
<input type="checkbox"/> Modifier  Copier  Effacer	6	2	5	2021-11-23 19:14:51	2021-11-23 19:14:51	2021-11-23 19:14:52	COMPLETED	COMPLETED	



# Architecture de base - COMPOSANTS











## JobRepository

```
SELECT * FROM `batch_job_instance`
```

Nombre de lignes : 25 ▼

Trier sur l'index: Aucune ▼

+ Options

				JOB_INSTANCE_ID	VERSION	JOB_NAME	JOB_KEY
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	3	0	listStocksJob	14c259a26f9a9012ed0e22a2db931b95
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	4	0	listStocksJob	d41d8cd98f00b204e9800998ecf8427e
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	5	0	listStocksJob	27f255df6f678fb6e277c0682de25c5e

# Architecture de base - COMPOSANTS

## JobRepository










```
SELECT * FROM `batch_step_execution`
```

☐ Profilage [ En ligne ] [ Modifier ] [ Expliquer SQL ] [ Créer source PHP ] [ Actualiser ]

Nombre de lignes : 25 ▼

Trier sur l'index: Aucune ▼

+ Options

←T→ ▼	STEP_EXECUTION_ID	VERSION	STEP_NAME	JOB_EXECUTION_ID	START_TIME	END_TIME	STATUS	COMMIT_COUNT	READ_COUNT
<input type="checkbox"/>  Modifier  Copier  Effacer	1	3	processingStep	4	2021-11-23 19:07:48	2021-11-23 19:07:48	COMPLETED	1	2
<input type="checkbox"/>  Modifier  Copier  Effacer	2	3	processingStep	3	2021-11-23 19:07:48	2021-11-23 19:07:48	COMPLETED	1	3
<input type="checkbox"/>  Modifier  Copier  Effacer	3	7	processingStep	6	2021-11-23 19:14:52	2021-11-23 19:14:52	COMPLETED	5	20

# Travail à faire

- Compléter les 8 toDo au niveau du code pour que le projet soit fonctionnel et que le batch puisse insérer les données correctement.

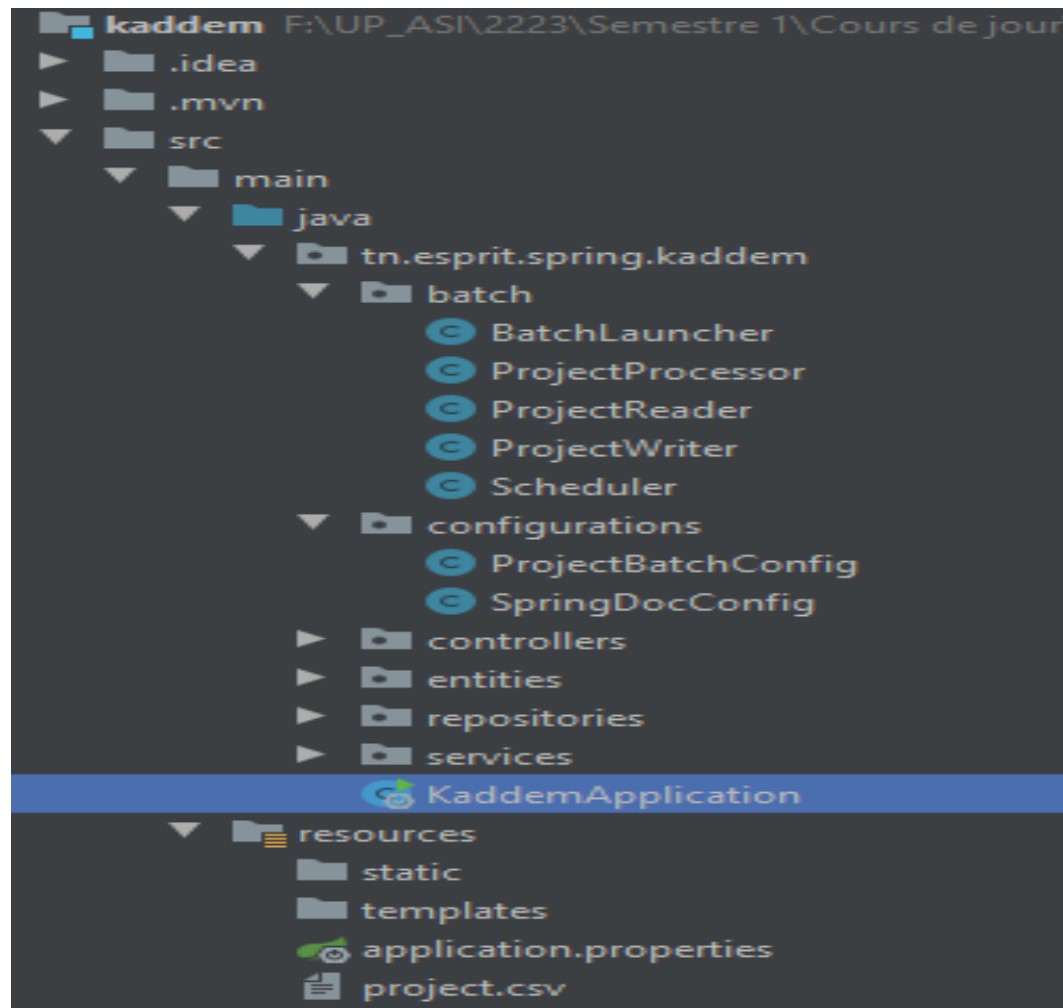
# Travail à faire

Créer un batch qui permet de faire les étapes suivantes :

- Lire les id des équipes qui ont des projets terminés
- Modifier le montant de contrats des membres de l'équipe (augmentation de 10% du montant initial) si le projet auquel ils sont attachés est fini (`terminated = true`)
- Sauvegarder le nom de l'étudiant, la date de l'augmentation du montant du contrat et le nouveau montant du contrat dans la table log

# TP: METTRE EN PLACE UN JOB

- Importer le projet Spring boot TODO ayant l'arborescence suivante et créer la base donnée kaddembatch :



# TP: METTRE EN PLACE UN JOB - Dépendances

- Pour utiliser Spring Batch avec Spring Boot, il faut ajouter les dépendances suivantes (déjà existantes dans le projet toDo):

```
<dependency>
```

```
    <groupId>org.springframework.batch</groupId>
```

```
    <artifactId>spring-batch-test</artifactId>
```

```
    <scope>test</scope>
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-batch</artifactId>
```

```
</dependency>
```

# TP: METTRE EN PLACE UN JOB - Configuration

- **Configuration du Job:**

- Pour activer le traitement par lots, nous devons annoter la classe de configuration **StockBatchConfig** avec **@EnableBatchProcessing**.
- Nous devons par la suite créer:
  - ✓ Un reader pour lire notre fichier CSV.
  - ✓ Un processor pour traiter les données d'entrée avant d'écrire.
  - ✓ Un writer pour écrire dans la base de données.

# TP: METTRE EN PLACE UN JOB – Les classes

- Inspecter en suivant l'ordre les classes suivantes (inspecter le code source des classes en respectant l'ordre) :
  - **BatchLauncher** : Lancer les jobs
  - **Scheduler** : Lancer le batch Launcher à intervalle de temps régulier pour démarrer les différents jobs (un dans notre cas ou plusieurs).
  - **ProjectProcessor**
  - **ProjectReader**
  - **ProjectWriter**
  - **ProjectBatchConfig**

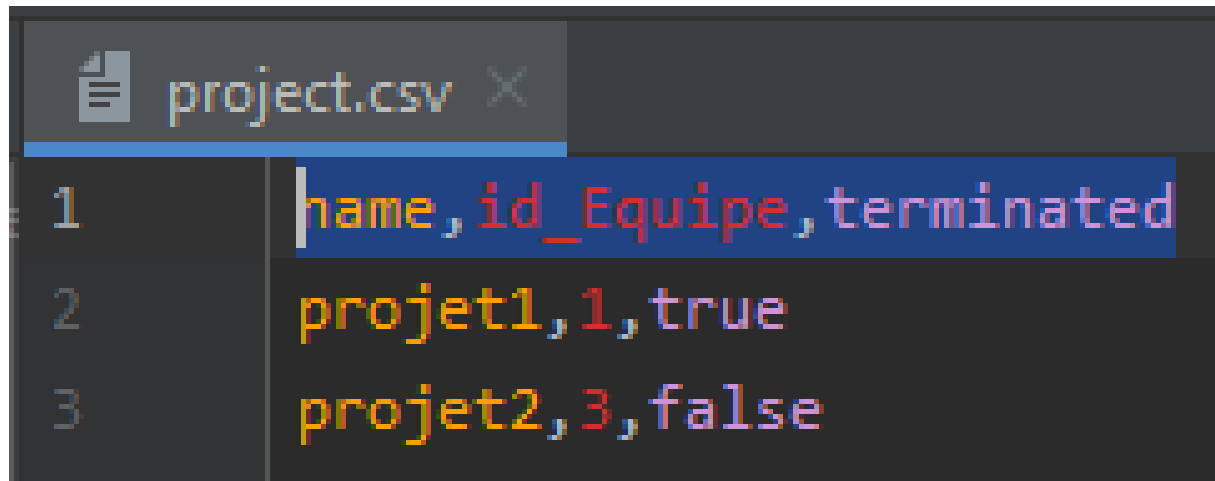


# TP: METTRE EN PLACE UN JOB - Configuration

- Ajouter la ligne suivante dans le fichier de configuration **application.properties**:

```
spring.batch.initialize-schema=ALWAYS
```

- Inspecter le fichier excel **project.csv** qui contient les lignes ajoutées dans le répertoire **src/main/resources**:

A screenshot of a code editor window titled 'project.csv'. The editor shows three lines of CSV data. The first line is a header: 'name,id\_Equipe,terminated'. The second line is 'projet1,1,true'. The third line is 'projet2,3,false'. The text is color-coded: 'name' is yellow, 'id\_Equipe' is red, 'terminated' is blue, 'projet1' is yellow, '1' is red, 'true' is blue, 'projet2' is yellow, '3' is red, and 'false' is blue.

1	name,id_Equipe,terminated
2	projet1,1,true
3	projet2,3,false

# TP: METTRE EN PLACE UN JOB

- Lancer le projet Spring Boot
- En cas d'erreur lors du lancement indiquant qu'une table n'existe pas (batch\_job\_execution par exemple), le lancement du script **batch.sql** va résoudre le problème (comportement liée au type du SGBD choisi)

# TP: METTRE EN PLACE UN JOB – Les tables

- Lancer le script **batch.sql** pour créer les tables de Spring Batch ( grâce auxquelles SpringBatch assure le monitoring des données manipulés dans nos traitements).

```
CREATE TABLE BATCH_JOB_INSTANCE (
    JOB_INSTANCE_ID BIGINT NOT NULL PRIMARY KEY ,
    VERSION BIGINT ,
    JOB_NAME VARCHAR(100) NOT NULL,
    JOB_KEY VARCHAR(32) NOT NULL,
    constraint JOB_INST_UN unique (JOB_NAME, JOB_KEY)
) ENGINE=InnoDB;
```

```
CREATE TABLE BATCH_JOB_EXECUTION (
    JOB_EXECUTION_ID BIGINT NOT NULL PRIMARY KEY ,
    VERSION BIGINT ,
    JOB_INSTANCE_ID BIGINT NOT NULL,
    CREATE TIME DATETIME NOT NULL,
```

# SPRING BATCH

Si vous avez des questions, n'hésitez pas à nous contacter :

**Département Informatique**  
**UP ASI**

Bureau E204

# SPRING BATCH

